In diesem File stellen alle Quelle-Codes, auf die im Handskript verwiesen wird.

Berechnungen, Teil 1:

03_Erste_varC-Schwingung

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
```

```
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_andere_Ausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion:   }
      Str(x[lv]{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion:   }
      Str(Q[lv]*1E6{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
```

```
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=1;                              {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=10000;                          {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                        {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                     {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=2e-6{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=10e-6{m};        {Dicke der Kunststoff-Folie: 10_Mü}
  D:=1.0{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Start der elektrischen Schwingung: }
  Q[0]:=2E-10{C}; Qp[0]:=0; Qpp[0]:=0;             {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};     {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=1E-4{sec.};                                                        {Zeitschritte}
  N:=20000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);            {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);          {Anziehung nach dem Coulomb-Gesetz}
                                          {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                    {Festhalten der Platten bis zum Zeitpunkt t=0}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
```

```
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln;     Wait;

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');     R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
  Writeln('  t/[sec.]  |           x/[m]          |        Q[i]');
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[0]*Q[0]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);     {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Write(i*dt{sec.}:11:9,' | ',x[i]{Volt},' |');
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Writeln(Q[i]); {Wait;}
  end;
  Excel_andere_Ausgabe('Teil_03.dat'); Writeln;

{-----------------------------------------------------------------}


  Wait;   Wait;
End.
```

04_Echte_Energie-Konversion

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;
```

```
Procedure Excel_andere_Ausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion:   }
      Str(x[lv]{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion:   }
      Str(Q[lv]*1E6{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
```

```
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Amplituden_anzeigen;
Var i             : Integer;
    schreibe      : Boolean;
    SteigX,SteigQ : Boolean;
    BildX,BildQ   : Array[0..200] of Double;
    zvx,zvQ       : Integer;
    eq,lq,ex,lx   : Double;
begin
{ Zuerst die x-Amplituden:}
  SteigX:=false; If x[1]>x[0] then SteigX:=true;
  schreibe:=false;  zvx:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigX then
    begin
      If x[i]<x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Max:'); end;
    end;
    If Not(SteigX) then
    begin
      If x[i]>x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildX[zvx]:=x[i]; zvx:=zvx+1;
    end;
    schreibe:=false;
  end;
  zvx:=zvx-1;
{ Danach die Q-Amplituden:}
  SteigQ:=false; If Q[1]>Q[0] then SteigQ:=true;
  schreibe:=false;  zvQ:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigQ then
    begin
      If Q[i]<Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Max:'); end;
    end;
    If Not(SteigQ) then
    begin
      If Q[i]>Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildQ[zvQ]:=Q[i]; zvQ:=zvQ+1;
    end;
    schreibe:=false;
  end;
  zvQ:=zvQ-1;
{ Jetzt der Überblick über "Spitze-Spitze":}
  Writeln('Orte, Spitze-Spitze:');
  i:=2;  ex:=BildX[i]-BildX[i-1];
  Repeat
    Writeln(i,': ',BildX[i]-BildX[i-1]);
```

```
    lx:=BildX[i]-BildX[i-1];
    i:=i+2;
  Until (i>=zvx);
  Writeln('Ladungen, Spitze-Spitze:');
  i:=2;  eq:=BildQ[i]-BildQ[i-1];
  Repeat
    Writeln(i,': ',BildQ[i]-BildQ[i-1]);
    lq:=BildQ[i]-BildQ[i-1];
    i:=i+2;
  Until (i>=zvQ);
  Write('Gesamtaenderung, Orts-Amplitude: ');
  If Abs(lx)>Abs(ex) then Write('+');
  If Abs(lx)<Abs(ex) then Write('-');
  Writeln(Abs(lx-ex));
  Write('Gesamtaenderung, Ladg-Amplitude: ');
  If Abs(lq)>Abs(eq) then Write('+');
  If Abs(lq)<Abs(eq) then Write('-');
  Writeln(Abs(lq-eq));
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=12650; {12264;}               {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                        {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                     {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                    {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=2e-6{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=10e-6{m};        {Dicke der Kunststoff-Folie: 10_Mü}
  D:=1.0{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Start der elektrischen Schwingung: }
  Q[0]:=2E-10{C}; Qp[0]:=0; Qpp[0]:=0;             {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};     {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=1E-4{sec.};                                                        {Zeitschritte}
  N:=30000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                        {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    Wait;
```

```
{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');     R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[0]*Q[0]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);     {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  If MacheFiles then Excel_andere_Ausgabe('Teil_03.dat'); Writeln;
  Amplituden_anzeigen;

{-----------------------------------------------------------------}

  Wait;   Wait;
End.
```

05_EnBer

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;
```

```
Procedure Excel_andere_Ausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion:   }
      Str(x[lv]{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion:   }
      Str(Q[lv]*1E6{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
```

```
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Amplituden_anzeigen;
Var i             : Integer;
    schreibe      : Boolean;
    SteigX,SteigQ : Boolean;
    BildX,BildQ   : Array[0..200] of Double;
    zvx,zvQ       : Integer;
    eq,lq,ex,lx   : Double;
    Wmech1,Wmech2,Wel1,Wel2:Double;
begin
{ Zuerst die x-Amplituden:}
  SteigX:=false; If x[1]>x[0] then SteigX:=true;
  schreibe:=false;  zvx:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigX then
    begin
      If x[i]<x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Max:'); end;
    end;
    If Not(SteigX) then
    begin
      If x[i]>x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildX[zvx]:=x[i]; zvx:=zvx+1;
    end;
    schreibe:=false;
  end;
  zvx:=zvx-1;
{ Danach die Q-Amplituden:}
  SteigQ:=false; If Q[1]>Q[0] then SteigQ:=true;
  schreibe:=false;  zvQ:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigQ then
    begin
      If Q[i]<Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Max:'); end;
    end;
    If Not(SteigQ) then
    begin
      If Q[i]>Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildQ[zvQ]:=Q[i]; zvQ:=zvQ+1;
    end;
    schreibe:=false;
  end;
  zvQ:=zvQ-1;
{ Jetzt der Überblick über "Spitze-Spitze":}
  Writeln('Orte, Spitze-Spitze:');
  i:=2;  ex:=BildX[i]-BildX[i-1];
```

```
  Repeat
    Writeln(i,': ',BildX[i]-BildX[i-1]);
    lx:=BildX[i]-BildX[i-1];
    i:=i+2;
  Until (i>=zvx);
  Writeln('Ladungen, Spitze-Spitze:');
  i:=2;  eq:=BildQ[i]-BildQ[i-1];
  Repeat
    Writeln(i,': ',BildQ[i]-BildQ[i-1]);
    lq:=BildQ[i]-BildQ[i-1];
    i:=i+2;
  Until (i>=zvQ);
  Write('Gesamtaenderung, Orts-Amplitude: ');
  If Abs(lx)>Abs(ex) then Write('+');
  If Abs(lx)<Abs(ex) then Write('-');
  om:=pi*zvx/N/dt; Writeln('Kreisfrequenz omega= ',om);
  Writeln(Abs(lx-ex));
  Wmech1:=m/2*(ex*ex)*om*om; Wmech2:=m/2*(lx*lx)*om*om;
  Writeln('Mechanische Energie zu Beginn: ',Wmech1,' Joule');
  Writeln('Mechanische Energie am Ende:   ',Wmech2,' Joule');
  Writeln('Energie-Veränderung: ',Wmech2-Wmech1,' Joule');
  Write('Gesamtaenderung, Ladg-Amplitude: ');
  If Abs(lq)>Abs(eq) then Write('+');
  If Abs(lq)<Abs(eq) then Write('-');
  Writeln(Abs(lq-eq));
  Wel1:=L/2*(eq*eq)*om*om; Wel2:=L/2*(lq*lq)*om*om;
  Writeln('Elektrische Energie zu Beginn: ',Wel1,' Joule');
  Writeln('Elektrische Energie am Ende:   ',Wel2,' Joule');
  Writeln('Energie-Veränderung: ',Wel2-Wel1,' Joule');
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=12530; {12264;}               {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                        {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                     {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=2e-6{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=10e-6{m};        {Dicke der Kunststoff-Folie: 10_Mü}
  D:=1.{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Start der elektrischen Schwingung: }
  Q[0]:=2E-10{C}; Qp[0]:=0; Qpp[0]:=0;             {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};     {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=3.5E-4{sec.};                                                      {Zeitschritte}
  N:=30000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                              {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                        {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=false;      {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
```

```
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    {Wait;}

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');     R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[0]*Q[0]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);     {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  If MacheFiles then Excel_andere_Ausgabe('Teil_03.dat'); Writeln;
  Amplituden_anzeigen;

{-----------------------------------------------------------------}

  Wait;   Wait;
End.
```

05_Mit_Energie-Berechnung

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;
```

```
Procedure Excel_andere_Ausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion:   }
      Str(x[lv]{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion:   }
      Str(Q[lv]*1E6{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
```

```
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Amplituden_anzeigen;
Var i             : Integer;
    schreibe      : Boolean;
    SteigX,SteigQ : Boolean;
    BildX,BildQ   : Array[0..200] of Double;
    zvx,zvQ       : Integer;
    eq,lq,ex,lx   : Double;
    Wmech1,Wmech2,Wel1,Wel2:Double;
begin
{ Zuerst die x-Amplituden:}
  SteigX:=false; If x[1]>x[0] then SteigX:=true;
  schreibe:=false;  zvx:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigX then
    begin
      If x[i]<x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Max:'); end;
    end;
    If Not(SteigX) then
    begin
      If x[i]>x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildX[zvx]:=x[i]; zvx:=zvx+1;
    end;
    schreibe:=false;
  end;
  zvx:=zvx-1;
{ Danach die Q-Amplituden:}
  SteigQ:=false; If Q[1]>Q[0] then SteigQ:=true;
  schreibe:=false;  zvQ:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigQ then
    begin
      If Q[i]<Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Max:'); end;
    end;
    If Not(SteigQ) then
    begin
      If Q[i]>Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildQ[zvQ]:=Q[i]; zvQ:=zvQ+1;
    end;
    schreibe:=false;
  end;
  zvQ:=zvQ-1;
{ Jetzt der Überblick über "Spitze-Spitze":}
  Writeln('Orte, Spitze-Spitze:');
  i:=2;  ex:=BildX[i]-BildX[i-1];
```

```
  Repeat
    Writeln(i,': ',BildX[i]-BildX[i-1]);
    lx:=BildX[i]-BildX[i-1];
    i:=i+2;
  Until (i>=zvx);
  Writeln('Ladungen, Spitze-Spitze:');
  i:=2;  eq:=BildQ[i]-BildQ[i-1];
  Repeat
    Writeln(i,': ',BildQ[i]-BildQ[i-1]);
    lq:=BildQ[i]-BildQ[i-1];
    i:=i+2;
  Until (i>=zvQ);
  Write('Gesamtaenderung, Orts-Amplitude: ');
  If Abs(lx)>Abs(ex) then Write('+');
  If Abs(lx)<Abs(ex) then Write('-');
  om:=pi*zvx/N/dt; Writeln('Kreisfrequenz omega= ',om);
  Writeln(Abs(lx-ex));
  Wmech1:=m/2*(ex*ex)*om*om; Wmech2:=m/2*(lx*lx)*om*om;
  Writeln('Mechanische Energie zu Beginn: ',Wmech1,' Joule');
  Writeln('Mechanische Energie am Ende:   ',Wmech2,' Joule');
  Writeln('Energie-Veränderung: ',Wmech2-Wmech1,' Joule');
  Write('Gesamtaenderung, Ladg-Amplitude: ');
  If Abs(lq)>Abs(eq) then Write('+');
  If Abs(lq)<Abs(eq) then Write('-');
  Writeln(Abs(lq-eq));
  Wel1:=L/2*(eq*eq)*om*om; Wel2:=L/2*(lq*lq)*om*om;
  Writeln('Elektrische Energie zu Beginn: ',Wel1,' Joule');
  Writeln('Elektrische Energie am Ende:   ',Wel2,' Joule');
  Writeln('Energie-Veränderung: ',Wel2-Wel1,' Joule');
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=12530; {12264;}               {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                        {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                     {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=2e-6{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=10e-6{m};        {Dicke der Kunststoff-Folie: 10_Mü}
  D:=1.{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Start der elektrischen Schwingung: }
  Q[0]:=2E-10{C}; Qp[0]:=0; Qpp[0]:=0;             {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};     {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=3.5E-4{sec.};                                                      {Zeitschritte}
  N:=30000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                              {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                        {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=false;      {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
```

```
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    {Wait;}

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');     R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[0]*Q[0]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);     {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  If MacheFiles then Excel_andere_Ausgabe('Teil_03.dat'); Writeln;
  Amplituden_anzeigen;

{-----------------------------------------------------------------}

  Wait;   Wait;
End.
```

06_mech_und_el_nimmt_zu

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
```

```
  Close(fout);
end;

Procedure Excel_andere_Ausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion:   }
      Str(x[lv]{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion:   }
      Str(Q[lv]*1E6{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Amplituden_anzeigen;
Var i             : Integer;
    schreibe      : Boolean;
    SteigX,SteigQ : Boolean;
    BildX,BildQ   : Array[0..200] of Double;
    zvx,zvQ       : Integer;
    eq,lq,ex,lx   : Double;
    Wmech1,Wmech2,Wel1,Wel2:Double;
begin
{ Zuerst die x-Amplituden:}
  SteigX:=false; If x[1]>x[0] then SteigX:=true;
  schreibe:=false;  zvx:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigX then
    begin
      If x[i]<x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Max:'); end;
    end;
    If Not(SteigX) then
    begin
      If x[i]>x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildX[zvx]:=x[i]; zvx:=zvx+1;
    end;
    schreibe:=false;
  end;
  zvx:=zvx-1;
{ Danach die Q-Amplituden:}
  SteigQ:=false; If Q[1]>Q[0] then SteigQ:=true;
  schreibe:=false;  zvQ:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigQ then
    begin
      If Q[i]<Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Max:'); end;
    end;
    If Not(SteigQ) then
    begin
      If Q[i]>Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildQ[zvQ]:=Q[i]; zvQ:=zvQ+1;
    end;
    schreibe:=false;
  end;
  zvQ:=zvQ-1;
{ Jetzt der Überblick über "Spitze-Spitze":}
```

```
  Writeln('Orte, Spitze-Spitze:');
  i:=2;  ex:=BildX[i]-BildX[i-1];
  Repeat
    Writeln(i,': ',BildX[i]-BildX[i-1]);
    lx:=BildX[i]-BildX[i-1];
    i:=i+2;
  Until (i>=zvx);
  Writeln('Ladungen, Spitze-Spitze:');
  i:=2;  eq:=BildQ[i]-BildQ[i-1];
  Repeat
    Writeln(i,': ',BildQ[i]-BildQ[i-1]);
    lq:=BildQ[i]-BildQ[i-1];
    i:=i+2;
  Until (i>=zvQ);
  Write('Gesamtaenderung, Orts-Amplitude: ');
  If Abs(lx)>Abs(ex) then Write('+');
  If Abs(lx)<Abs(ex) then Write('-');
  om:=pi*zvx/N/dt; Writeln('Kreisfrequenz omega= ',om);
  Writeln(Abs(lx-ex));
  Wmech1:=m/2*(ex*ex)*om*om; Wmech2:=m/2*(lx*lx)*om*om;
  Writeln('Mechanische Energie zu Beginn: ',Wmech1,' Joule');
  Writeln('Mechanische Energie am Ende:   ',Wmech2,' Joule');
  Writeln('Mechan. Energie-Veränderung: ',Wmech2-Wmech1,' Joule');
  Write('Gesamtaenderung, Ladg-Amplitude: ');
  If Abs(lq)>Abs(eq) then Write('+');
  If Abs(lq)<Abs(eq) then Write('-');
  Writeln(Abs(lq-eq));
  Wel1:=L/2*(eq*eq)*om*om; Wel2:=L/2*(lq*lq)*om*om;
  Writeln('Elektrische Energie zu Beginn: ',Wel1,' Joule');
  Writeln('Elektrische Energie am Ende:   ',Wel2,' Joule');
  Writeln('Elektr. Energie-Veränderung: ',Wel2-Wel1,' Joule'); Writeln;
  Writeln('Summe: Gesamt-Energiegewinn: ',Wmech2-Wmech1+Wel2-Wel1,' Joule'); Writeln;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=12534; {12264;}               {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                        {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                      {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                    {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=2e-6{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=10e-6{m};        {Dicke der Kunststoff-Folie: 10_Mü}
  D:=1.0{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Start der elektrischen Schwingung: }
  Q[0]:=2E-10{C}; Qp[0]:=0; Qpp[0]:=0;             {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};     {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=3.53E-4{sec.};                                                        {Zeitschritte}
  N:=30000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                        {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;      {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
```

```
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    {Wait;}

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');     R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[0]*Q[0]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);     {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  If MacheFiles then Excel_andere_Ausgabe('Teil_03.dat'); Writeln;
  Amplituden_anzeigen;

{-----------------------------------------------------------------}

  Wait;   Wait;
End.
```

Harmon_mechan_Schwingg

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
```

```
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(QP[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    Fc,Fd : Double; {Coulombkraft und Federkraft}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=1;                              {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=5;                                                  {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
```

```
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=200000;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                       {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                    {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                    {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=2e-6{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=10e-6{m};        {Dicke der Kunststoff-Folie: 10_Mü}
  D:=1.0{N/m};            {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Start der elektrischen Schwingung: }
  Q[0]:=2E-10{C}; Qp[0]:=0; Qpp[0]:=0;            {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};     {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=1E-4{sec.};                                                        {Zeitschritte}
  N:=2000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  X[0]:=Plapos(0);  {Iterative Ermittlung der Position der Kondensatorplatten.}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);        {Anziehung nach dem Coulomb-Gesetz}
                                              {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                  {Festhalten der Platten bis zum Zeitpunkt t=0}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln;     Wait;

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');     R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
  Writeln('  t/[sec.]  |  x/[m] | ');
  For i:=1 to N do
  begin
    xpp[i]:=-1/m*D*(x[i-1]-CD/2)+1/m*Q[0]*Q[0]/4/pi/epo/(2*x[i-1])*(2*x[i-1]);
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    C:=epo*epr*CA/(2*x[i]);    Qp[i]:=C;
    Writeln(i*dt{sec.}:11:9,' | ',x[i]{Volt},' |');
```

```
  end;
  Excel_eine_Kolumne('Teil_03.dat'); Writeln;

{-----------------------------------------------------------------}


  Wait;   Wait;
End.
```

Klassischer_Schwingkreis

```
Program Klassischer_Schwingkreis_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..1000000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..1000000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;


Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
```

```
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=1;                              {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.001{m};                                  {Kondensator-Geometrie}
  epr:=1;                                              {Keine Materialien im Kondensator}
  C:=epo*epr*CA/CD;                                                          {Kapazität}
{ Spule: }
  SN:=2000; SL:=0.05{m}; SR:=0.02{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=1;                                                {Keine Materialien in der Spule}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                      {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                    {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                  {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=1e-5{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=1e-5{m};         {Dicke der Kunststoff-Folie: 10_Mü}
  D:=500{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Start der elektrischen Schwingung: }
  Q[0]:=3E-8{C}; Qp[0]:=0; Qpp[0]:=0;            {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};                               {Spannung über dem Kondensator zu Beginn}
  dt:=100E-9{sec.};                                                          {Zeitschritte}
  N:=3000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  F:=1/4/pi/epo*Q[0]*Q[0]/CD*CD;                    {Anziehung nach dem Coulomb-Gesetz}
  x[0]:=-F/D;                                  {Verformung der beiden Kondensatorplatten}
                                               {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                        {Festhalten der Platten bis zum Zeitpunkt t=0}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:10:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln;     Wait;

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Writeln(i*dt{sec.}:11:9,' | ',Q[i]/C{Volt}:7:2,' |');
  end;
  Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');     R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }    {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;   {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Writeln(i*dt{sec.}:11:9,' | ',Q[i]/C{Volt}:7:2,' |');
  end;
  Excel_Datenausgabe('Teil_02.dat'); Writeln;

  Wait;   Wait;
End.
```

Raumenergie_Schwingkreis

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;
```

```
Procedure Excel_andere_Ausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion:   }
      Str(x[lv]{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion:   }
      Str(Q[lv]*1E6{Volt}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
```

```
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Amplituden_anzeigen;
Var i             : Integer;
    schreibe      : Boolean;
    SteigX,SteigQ : Boolean;
    BildX,BildQ   : Array[0..200] of Double;
    zvx,zvQ       : Integer;
    eq,lq,ex,lx   : Double;
    Wmech1,Wmech2,Wel1,Wel2:Double;
begin
{ Zuerst die x-Amplituden:}
  SteigX:=false; If x[1]>x[0] then SteigX:=true;
  schreibe:=false;  zvx:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigX then
    begin
      If x[i]<x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Max:'); end;
    end;
    If Not(SteigX) then
    begin
      If x[i]>x[i-1] then begin schreibe:=true; SteigX:=Not(SteigX); Write('X-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildX[zvx]:=x[i]; zvx:=zvx+1;
    end;
    schreibe:=false;
  end;
  zvx:=zvx-1;
{ Danach die Q-Amplituden:}
  SteigQ:=false; If Q[1]>Q[0] then SteigQ:=true;
  schreibe:=false;  zvQ:=0;
  Writeln('     I:   t/[sec.]  |           x/[m]         |        Q[i]');
  For i:=1 to N do
  begin
    If SteigQ then
    begin
      If Q[i]<Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Max:'); end;
    end;
    If Not(SteigQ) then
    begin
      If Q[i]>Q[i-1] then begin schreibe:=true; SteigQ:=Not(SteigQ); Write('Q-Min:'); end;
    end;
    If schreibe then
    begin
      Writeln(i:6,': ',i*dt:7:5,' | ',x[i],' |',Q[i]); {Wait;}
      BildQ[zvQ]:=Q[i]; zvQ:=zvQ+1;
    end;
    schreibe:=false;
  end;
  zvQ:=zvQ-1;
{ Jetzt der Überblick über "Spitze-Spitze":}
  Writeln('Orte, Spitze-Spitze:');
  i:=2;  ex:=BildX[i]-BildX[i-1];
```

```
  Repeat
    Writeln(i,': ',BildX[i]-BildX[i-1]);
    lx:=BildX[i]-BildX[i-1];
    i:=i+2;
  Until (i>=zvx);
  Writeln('Ladungen, Spitze-Spitze:');
  i:=2;  eq:=BildQ[i]-BildQ[i-1];
  Repeat
    Writeln(i,': ',BildQ[i]-BildQ[i-1]);
    lq:=BildQ[i]-BildQ[i-1];
    i:=i+2;
  Until (i>=zvQ);
  Write('Gesamtaenderung, Orts-Amplitude: ');
  If Abs(lx)>Abs(ex) then Write('+');
  If Abs(lx)<Abs(ex) then Write('-');
  om:=pi*zvx/N/dt; Writeln('Kreisfrequenz omega= ',om);
  Writeln(Abs(lx-ex));
  Wmech1:=m/2*(ex*ex)*om*om; Wmech2:=m/2*(lx*lx)*om*om;
  Writeln('Mechanische Energie zu Beginn: ',Wmech1,' Joule');
  Writeln('Mechanische Energie am Ende:   ',Wmech2,' Joule');
  Writeln('Mechan. Energie-Veränderung: ',Wmech2-Wmech1,' Joule');
  Write('Gesamtaenderung, Ladg-Amplitude: ');
  If Abs(lq)>Abs(eq) then Write('+');
  If Abs(lq)<Abs(eq) then Write('-');
  Writeln(Abs(lq-eq));
  Wel1:=L/2*(eq*eq)*om*om; Wel2:=L/2*(lq*lq)*om*om;
  Writeln('Elektrische Energie zu Beginn: ',Wel1,' Joule');
  Writeln('Elektrische Energie am Ende:   ',Wel2,' Joule');
  Writeln('Elektr. Energie-Veränderung: ',Wel2-Wel1,' Joule'); Writeln;
  Writeln('Summe: Gesamt-Energiegewinn: ',Wmech2-Wmech1+Wel2-Wel1,' Joule'); Writeln;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                  {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=12534; {12264;}               {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                       {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                     {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=2e-6{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=10e-6{m};        {Dicke der Kunststoff-Folie: 10_Mü}
  D:=1.0{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Start der elektrischen Schwingung: }
  Q[0]:=2E-10{C}; Qp[0]:=0; Qpp[0]:=0;             {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};     {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=3.53E-4{sec.};                                                          {Zeitschritte}
  N:=30000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                                 {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                          {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;      {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
```

```
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    {Wait;}

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');     R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                                {Federkraft gegenüber CD}
    Fc:=-Q[0]*Q[0]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);     {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                   {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  If MacheFiles then Excel_andere_Ausgabe('Teil_03.dat'); Writeln;
  Amplituden_anzeigen;

{-----------------------------------------------------------------}

  Wait;   Wait;
End.
```

Berechnungen, Teil 2:

Erzw_Schwing_10_30muWatt

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
```

```
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
```

```
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
```

```
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
```

```
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=34.50;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.4; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.1;{Volt}    Pulsdauer:=830;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
```

```
      begin
        Umerk:=Uo4;                        {Spannung anlegen}
      end;
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=1 to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/N;
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=220;                          {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                        {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                      {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                    {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                 {Dichte der Kunststoff-Folie}
```

```
  dAL:=2570e-6{m};                                    {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                        {Dicke der Kunststoff-Folie}
  D:=6400{N/m};            {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;              {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=20E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                        {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                          {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
```

```
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];     {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
```

```
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then Reibung:=true;
    Flast:=0;
    If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                            {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                    {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;             {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];            {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;
{-------------------------------------------------------------}
  Wait;   Wait;
End.
```

Erzw_Schwing_10_143muWatt

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
```

```
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
```

```
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
```

```
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
```

```
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=12.0;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.4; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.3;{Volt}    Pulsdauer:=700;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
      end;
```

```
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=1 to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/N;
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=500;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                          {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                       {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                    {Dichte der Kunststoff-Folie}
  dAL:=3350e-6{m};                                  {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                           {Dicke der Kunststoff-Folie}
  D:=50000{N/m};            {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
```

```
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;              {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=20E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                      {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
```

```
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];     {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
```

```
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                      {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                           {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then Reibung:=true;
    Flast:=0;
    If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                          {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                     {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];             {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Erzw_Schwing_10_229muWatt

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
```

```
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
```

```
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
```

```
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
```

```
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=12.0;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.4; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.3;{Volt}    Pulsdauer:=700;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
      end;
```

```
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=1 to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/N;
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=502;                       {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                         {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                      {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                    {Dichte der Kunststoff-Folie}
  dAL:=3348e-6{m};                                  {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                           {Dicke der Kunststoff-Folie}
  D:=50010{N/m};            {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
```

```
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;              {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=20E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                      {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
```

```
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];     {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
```

```
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                      {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                           {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then Reibung:=true;
    Flast:=0;
    If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                          {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                     {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];             {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Erzw_Schwing_10_2800muWatt

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}
    Ianf        : Integer; {Anfang der Leistungsentnahme}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
```

```
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
```

```
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
```

```
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
```

```
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=8;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.25; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.100;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
```

```
      end;
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=Ianf to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/(N-Ianf);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=502;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                         {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                      {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                  {Dichte der Kunststoff-Folie}
  dAL:=3800e-6{m};                                {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                          {Dicke der Kunststoff-Folie}
```

```
  D:=86500{N/m};            {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;             {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=20E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
```

```
{   Writeln('  t/[sec.]  |           x/[m]         |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];     {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
```

```
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                           {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then begin Reibung:=true; Ianf:=i-1; end;
    Flast:=0;
    If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                           {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                      {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];             {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-------------------------------------------------------------}
  Wait;   Wait;
End.
```

Erzw_Schwing_10_4500muWatt

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}
    Ianf        : Integer; {Anfang der Leistungsentnahme}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
```

```
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
```

```
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
```

```
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
```

```
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=8;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.00; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.10;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
```

```
      end;
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=Ianf to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/(N-Ianf);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=502;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                         {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                      {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                  {Dichte der Kunststoff-Folie}
  dAL:=3800e-6{m};                                {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                          {Dicke der Kunststoff-Folie}
```

```
  D:=86487{N/m};            {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;             {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=20E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                        {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
```

```
{   Writeln('  t/[sec.]  |           x/[m]         |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];     {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
```

```
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                           {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then begin Reibung:=true; Ianf:=i-1; end;
    Flast:=0;
    If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                           {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                      {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];             {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-------------------------------------------------------------}
  Wait;   Wait;
End.
```

Erzw_Schwing_10_6379nW

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
```

```
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
```

```
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
```

```
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
```

```
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=34.50;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.4; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.1;{Volt}    Pulsdauer:=830;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
      end;
```

```
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=1 to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/N;
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=400;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                          {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                        {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                    {Dichte der Kunststoff-Folie}
  dAL:=1000e-6{m};                                  {Dicke der Metall-Kondensatorplatten}
  dFol:=200e-6{m};                                    {Dicke der Kunststoff-Folie}
  D:=400{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
```

```
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;              {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=500E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                      {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
```

```
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];     {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
```

```
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                      {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                           {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then Reibung:=true;
    Flast:=0;
    If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                          {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                     {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];             {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Erzw_Schwing_10_gpr

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
```

```
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
```

```
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/1000{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
```

```
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
```

```
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=34.50;      Phasenshift:=Phasenshift/(100*dt);
  If Pulsform='Trigger' then
  begin
    Uo4:=0.1;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
      end;
    end;
  end;
```

```
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
  If i>N/2 then Umerk:=0; {Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=1 to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/N;
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=400;                          {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                        {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                      {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                    {Dichte der Kunststoff-Folie}
  dAL:=1000e-6{m};                                  {Dicke der Metall-Kondensatorplatten}
  dFol:=200e-6{m};                                    {Dicke der Kunststoff-Folie}
  D:=400{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
```

```
  fFol:=omFol/2/pi;              {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=500E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                          {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]         |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
```

```
      Fd:=-D*(x[i-1]-CD/2);                                {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                  {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
```

```
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    Flast:=2.50*xp[i-1];   {Eine klassische Geschwindigkeits-proportionale Reibung geht immer.}
{## Flast:=0;              {Bei Bedarf kann die Lastkraft ausgeschaltet werden.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                             {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                     {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];            {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-------------------------------------------------------------}
  Wait;   Wait;
End.
```

Erzw_Schwing_10_last_einschalten

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
```

```
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
```

```
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
```

```
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
```

```
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=34.50;      Phasenshift:=Phasenshift/(100*dt);
  If Pulsform='Trigger' then
  begin
    Uo4:=0.1;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
      end;
    end;
```

```
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=1 to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/N;
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=400;                          {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                          {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                        {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                      {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                    {Dichte der Kunststoff-Folie}
  dAL:=1000e-6{m};                                  {Dicke der Metall-Kondensatorplatten}
  dFol:=200e-6{m};                                    {Dicke der Kunststoff-Folie}
  D:=400{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
```

```
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;             {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=500E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                          {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                    {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
```

```
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
```

```
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                      {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then Reibung:=true;      {ES>1/32*D*CD*CD}
    Flast:=0;
    If Reibung then Flast:=0.85*Fges;     {begin Flast:=2.50*xp[i-1]; end;  {Eine klassische Geschwindigkeits-
proportionale Reibung geht immer.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-System: }
    xpp[i]:=Fges/m;                            {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                     {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];             {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Erzw_Schwing_10_last_einschalten_phasenkorr

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
```

```
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
```

```
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
```

```
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
```

```
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=34.50;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.4; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.1;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
      end;
    end;
```

```
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=1 to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/N;
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=400;                          {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                          {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                        {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                      {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                    {Dichte der Kunststoff-Folie}
  dAL:=1000e-6{m};                                  {Dicke der Metall-Kondensatorplatten}
  dFol:=200e-6{m};                                    {Dicke der Kunststoff-Folie}
  D:=400{N/m};             {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
```

```
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;             {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=500E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                          {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                    {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
```

```
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                 {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;           {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
```

```
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                      {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then Reibung:=true;      {ES>1/32*D*CD*CD}
    Flast:=0;
    If Reibung then Flast:=0.85*Fges;     {begin Flast:=2.50*xp[i-1]; end;  {Eine klassische Geschwindigkeits-
proportionale Reibung geht immer.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-System: }
    xpp[i]:=Fges/m;                            {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                     {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];             {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Mit_Sekundärspule_59mW

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Pse         : Array[0..200000] of Double;  {Leistungs-Entnahme über die Sekundär-Spule}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}
    Ianf        : Integer; {Anfang der Leistungsentnahme}
    Rv          : Double;  {Verbraucher-Widerstand zur Leistungsentnahme aus der Spule}
    Psekmittel  : Double;  {Mittlere Leistungs-Entnahme über die Sekundär-Spule}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
```

```
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
```

```
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=8;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.00; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.10;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
```

```
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                        {Spannung anlegen}
      end;
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=Ianf to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/(N-Ianf);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Procedure Leistungsmittel_ueber_Sekundaerspule;
Var lv : LongInt;
begin
  Psekmittel:=0;
  For lv:=Ianf to N do Psekmittel:=Psekmittel+Pse[lv];
  Psekmittel:=Psekmittel/(N-Ianf);
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
```

```
  mur:=502;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                    {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                  {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};              {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                {Dichte der Kunststoff-Folie}
  dAL:=3800e-6{m};                              {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                  {Dicke der Kunststoff-Folie}
  D:=86487{N/m};          {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);       {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;            {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=620;  {früher 10 kiloOhm}                    {Elektrischer Lastwiderstand}
  Rv:=10E8;                {Verbraucher-Widerstand zur Leistungsentnahme aus der Spule}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                      {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                    {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                          {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                    {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
```

```
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                        {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}
```

```
  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;    Pse[0]:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                                {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                          {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then begin Reibung:=true; Ianf:=i-1; end;
    Flast:=0;  If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
{   Flast:=0; {Wir schalten hier die mechanische Reibung im Kondensator aus.}
    {Die Leistungsentnahme soll später elektrisch als der Spule über einen Trafo geschehen.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                           {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
{   Hier beginnt die Berechnung der elektrischen Schwingung}
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);  {Diese Spannung soll die Schwingung anregen.}
{##}Utrig[i]:=Uin;
{   Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;   {Früher, ohne Leistungsentnahme aus der Spule}
    Qpp[i]:=-1/C/(R+Rlast+Rv)*Qp[i-1]-1/L/C*Rv/(R+Rlast+Rv)*Q[i-1]-(R+Rlast)/L/2*Rv/(R+Rlast+Rv)*Qp[i-
1]+1/L*Rv/(R+Rlast+Rv)*Uin;
                                                                          {Jetzt mit Leistungsentnahme aus
der Spule}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {uralter Test: alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                                  {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;                           {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];                        {elektrische Energie der Spule}
    Pse[i]:=Abs(L*Qp[i]*Qpp[i]);              {Über Joch und Sekundärspule entnommene Leistung}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
```

```
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;     Writeln;
  Writeln('Mittlere mechanische Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;
  Ianf:=1; Leistungsmittel_ueber_Sekundaerspule;
  Writeln('Mittlere Leistungs-Entnahme ueber Sekundaerspule : ',Psekmittel,' Watt');

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Mit_Sekundärspule_63mW

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Pse         : Array[0..200000] of Double;  {Leistungs-Entnahme über die Sekundär-Spule}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}
    Ianf        : Integer; {Anfang der Leistungsentnahme}
    Rv          : Double;  {Verbraucher-Widerstand zur Leistungsentnahme aus der Spule}
    Psekmittel  : Double;  {Mittlere Leistungs-Entnahme über die Sekundär-Spule}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
```

```
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
```

```
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=8;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.00; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.10;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
```

```
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                        {Spannung anlegen}
      end;
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=Ianf to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/(N-Ianf);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Procedure Leistungsmittel_ueber_Sekundaerspule;
Var lv : LongInt;
begin
  Psekmittel:=0;
  For lv:=Ianf to N do Psekmittel:=Psekmittel+Pse[lv];
  Psekmittel:=Psekmittel/(N-Ianf);
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
```

```
  mur:=501;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                    {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                  {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19280{kg/m³};              {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                {Dichte der Kunststoff-Folie}
  dAL:=3800e-6{m};                              {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                  {Dicke der Kunststoff-Folie}
  D:=86487{N/m};          {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);       {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;            {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=640;  {früher 10 kiloOhm}                    {Elektrischer Lastwiderstand}
  Rv:=10E8;                {Verbraucher-Widerstand zur Leistungsentnahme aus der Spule}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                      {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                    {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                          {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                    {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
```

```
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                        {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}
```

```
  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;    Pse[0]:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                                {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                          {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then begin Reibung:=true; Ianf:=i-1; end;
    Flast:=0;  If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
{   Flast:=0; {Wir schalten hier die mechanische Reibung im Kondensator aus.}
    {Die Leistungsentnahme soll später elektrisch als der Spule über einen Trafo geschehen.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                           {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
{   Hier beginnt die Berechnung der elektrischen Schwingung}
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);  {Diese Spannung soll die Schwingung anregen.}
{##}Utrig[i]:=Uin;
{   Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;   {Früher, ohne Leistungsentnahme aus der Spule}
    Qpp[i]:=-1/C/(R+Rlast+Rv)*Qp[i-1]-1/L/C*Rv/(R+Rlast+Rv)*Q[i-1]-(R+Rlast)/L/2*Rv/(R+Rlast+Rv)*Qp[i-
1]+1/L*Rv/(R+Rlast+Rv)*Uin;
                                                                          {Jetzt mit Leistungsentnahme aus
der Spule}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {uralter Test: alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                                  {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;                           {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];                        {elektrische Energie der Spule}
    Pse[i]:=Abs(L*Qp[i]*Qpp[i]);              {Über Joch und Sekundärspule entnommene Leistung}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
```

```
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;     Writeln;
  Writeln('Mittlere mechanische Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;
  Ianf:=1; Leistungsmittel_ueber_Sekundaerspule;
  Writeln('Mittlere Leistungs-Entnahme ueber Sekundaerspule : ',Psekmittel,' Watt');

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Mit_Sekundärspule_180nanoWatt

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Pse         : Array[0..200000] of Double;  {Leistungs-Entnahme über die Sekundär-Spule}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}
    Ianf        : Integer; {Anfang der Leistungsentnahme}
    Rv          : Double;  {Verbraucher-Widerstand zur Leistungsentnahme aus der Spule}
    Psekmittel  : Double;  {Mittlere Leistungs-Entnahme über die Sekundär-Spule}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
```

```
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
```

```
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=8;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.00; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.1;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
```

```
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                        {Spannung anlegen}
      end;
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=Ianf to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/(N-Ianf);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Procedure Leistungsmittel_ueber_Sekundaerspule;
Var lv : LongInt;
begin
  Psekmittel:=0;
  For lv:=Ianf to N do Psekmittel:=Psekmittel+Pse[lv];
  Psekmittel:=Psekmittel/(N-Ianf);
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=1.0; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
```

```
  mur:=1200;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                              {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                    {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                 {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};               {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                  {Dichte der Kunststoff-Folie}
  dAL:=5000e-6{m};                                {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                   {Dicke der Kunststoff-Folie}
  D:=12000{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;             {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=1E-10;  {früher 10 kiloOhm}                      {Elektrischer Lastwiderstand}
  Rv:=300E6;                {Verbraucher-Widerstand zur Leistungsentnahme aus der Spule}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                    {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                    {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
```

```
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                        {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}
```

```
  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;    Pse[0]:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                                {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                          {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then begin Reibung:=true; Ianf:=i-1; end;
    Flast:=0;  If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    {Die Leistungsentnahme soll später elektrisch als der Spule über einen Trafo geschehen.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                           {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
{   Hier beginnt die Berechnung der elektrischen Schwingung}
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);  {Diese Spannung soll die Schwingung anregen.}
{##}Utrig[i]:=Uin;
{   Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;   {Früher, ohne Leistungsentnahme aus der Spule}
    Qpp[i]:=-1/C/(R+Rlast+Rv)*Qp[i-1]-1/L/C*Rv/(R+Rlast+Rv)*Q[i-1]-(R+Rlast)/L/2*Rv/(R+Rlast+Rv)*Qp[i-
1]+1/L*Rv/(R+Rlast+Rv)*Uin;
                                                                          {Jetzt mit Leistungsentnahme aus
der Spule}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {uralter Test: alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                          {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;                   {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];                 {elektrische Energie der Spule}
    Pse[i]:=Abs(L*Qp[i]*Qpp[i]);             {Über Joch und Sekundärspule entnommene Leistung}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
```

```
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;     Writeln;
  Writeln('Mittlere mechanische Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;
  Ianf:=1; Leistungsmittel_ueber_Sekundaerspule;
  Writeln('Mittlere Leistungs-Entnahme ueber Sekundaerspule : ',Psekmittel,' Watt');

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Param_optimieren_01

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
```

```
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion: Ort}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion: Ladung}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dritte Funktion: Energie mechanisch}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Vierte Funktion: Energie elektrisch}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
```

```
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt');
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=2;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=5200; {12534;}              {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
```

```
  AD:=pi*0.0002*0.0002{m²};                      {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                    {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                    {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};    {Dichte von Aluminium}
  rhoFol:=1500{kg/m³};   {Dichte der Kunststoff-Folie}
  dAL:=2e-6{m};          {Dicke der Aluminium-Kondensatorplatten: 10_Mü}
  dFol:=10e-6{m};        {Dicke der Kunststoff-Folie: 10_Mü}
  D:=2.0{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=200E3;    {Ohm}                                  {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=2.00E-10{C}; Qp[0]:=0; Qpp[0]:=0;          {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=5E-5{sec.};                                                    {Zeitschritte}
  N:=50000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                                 {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    { Wait; }

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]         |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
```

```
    Fd:=-D*(x[i-1]-CD/2);                                {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                          {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
  Leistung_berechnen;

{-------------------------------------------------------------------}

  Wait;   Wait;
End.
```

Param_optimieren_02_4.2mW

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]/C{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
```

```
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion: Ort}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion: Ladung}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dritte Funktion: Energie mechanisch}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Vierte Funktion: Energie elektrisch}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
```

```
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
```

```
      Letzte:=EgesOO;
    end;
  end;
  Writeln('Gewonnene Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=2;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                  {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=10603;                          {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                       {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                     {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                  {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};    {Hier Wolfram eingesetzt} {ansonsten: 2700 für Aluminium}
  rhoFol:=2500{kg/m³};    {Dichte der Kunststoff-Folie}
  dAL:=5000e-6{m};        {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};          {Dicke der Kunststoff-Folie}
  D:=4850{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=50E3;    {Ohm}                                    {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=160.2E-10{C}; Qp[0]:=0; Qpp[0]:=0;           {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=1.7E-5{sec.};                                                      {Zeitschritte}
  N:=60000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                          {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    { Wait; }

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
```

```
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}

  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]         |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                          {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
  EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
  Leistung_berechnen;

{-----------------------------------------------------------------}

  Wait;   Wait;
End.
```

Param_optimieren_03

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Nur bei Teil 4 ist durch "C4" zu teilen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
```

```
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion: Ort}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion: Ladung}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dritte Funktion: Energie mechanisch}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Vierte Funktion: Energie elektrisch}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
```

```
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
```

```
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln('Gewonnene Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=2;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                  {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=10603;                          {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                       {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                     {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                  {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};    {Hier Wolfram eingesetzt} {ansonsten: 2700 für Aluminium}
  rhoFol:=2500{kg/m³};    {Dichte der Kunststoff-Folie}
  dAL:=5000e-6{m};        {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};          {Dicke der Kunststoff-Folie}
  D:=4850{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=50E3;    {Ohm}                                    {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=160.2E-10{C}; Qp[0]:=0; Qpp[0]:=0;           {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=1.7E-5{sec.};                                                        {Zeitschritte}
  N:=60000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                        {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    { Wait; }

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
```

```
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                        {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
  EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
  Leistung_berechnen;

{-----------------------------------------------------------------}
  Writeln;
  Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
  R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
  L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Param_optimieren_04

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
```

```
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion: Ort}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion: Ladung}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dritte Funktion: Energie mechanisch}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Vierte Funktion: Energie elektrisch}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
```

```
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
```

```
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln('Gewonnene Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
```

```
  Abstd:=2;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=10603;                         {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                      {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                    {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                 {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};    {Hier Wolfram eingesetzt} {ansonsten: 2700 für Aluminium}
  rhoFol:=2500{kg/m³};    {Dichte der Kunststoff-Folie}
  dAL:=5000e-6{m};        {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};          {Dicke der Kunststoff-Folie}
  D:=4850{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=50E3;    {Ohm}                                  {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=160.2E-10{C}; Qp[0]:=0; Qpp[0]:=0;          {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=1.7E-5{sec.};                                                      {Zeitschritte}
  N:=60000;                                          {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                              {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    { Wait; }

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
```

```
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                          {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
  EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
  Leistung_berechnen;

{-----------------------------------------------------------------}
  Writeln;
  Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
  R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
  L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
  Pin[0]:=0;        {Initialisierung für Leistungsmessung}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
  Zugefuehrte_Leistung_mitteln;    Writeln;

{-----------------------------------------------------------------}

  Writeln('5.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}
  R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
  If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  For i:=1 to N do
  begin
    Uin:=U5(i*dt);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
  end;
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Param_optimieren_05

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
```

```
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion: Ort}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion: Ladung}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dritte Funktion: Energie mechanisch}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Vierte Funktion: Energie elektrisch}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
```

```
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
end;
```

```
Procedure EnergieMaxima;
Var i                     : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln('Gewonnene Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
```

```
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=2;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=10603;                          {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                        {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                      {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                    {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};    {Hier Wolfram eingesetzt} {ansonsten: 2700 für Aluminium}
  rhoFol:=2500{kg/m³};    {Dichte der Kunststoff-Folie}
  dAL:=5000e-6{m};        {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};          {Dicke der Kunststoff-Folie}
  D:=4850{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=50E3;    {Ohm}                                  {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=160.2E-10{C}; Qp[0]:=0; Qpp[0]:=0;           {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=1.7E-5{sec.};                                                        {Zeitschritte}
  N:=60000;                                        {Anzahl der Zeitschritte insgesamt}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                              {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                      {Festhalten der Platten bis zum Zeitpunkt t=0}
```

```
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    { Wait; }

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                        {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
  EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
  Leistung_berechnen;

{-----------------------------------------------------------------}
  Writeln;
```

```
  Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
  R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
  L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
  Pin[0]:=0;        {Initialisierung für Leistungsmessung}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
  Zugefuehrte_Leistung_mitteln;    Writeln;

{-----------------------------------------------------------------}

  Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}
  R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
  If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  For i:=1 to N do
  begin
    Uin:=U5(i*dt);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
  end;
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;

{-----------------------------------------------------------------}

  Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}
  R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
  If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  For i:=1 to N do
  begin
    Uin:=U6(i*dt);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
  end;
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;

{-----------------------------------------------------------------}

  Wait;   Wait;
End.
```

Param_optimieren_06

```
Program Harmonischer_Oszillator_im_DFEM;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i           : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
```

```
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Erste Funktion: Ort}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Zweite Funktion: Ladung}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dritte Funktion: Energie mechanisch}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Vierte Funktion: Energie elektrisch}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;
```

```
Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
```

```
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
end;

Procedure EnergieMaxima;
Var i                     : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte          : Double;
    Neu                   : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln('Gewonnene Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
```

```
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U7:=merk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U7(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
```

```
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=2;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=0.1*0.1{m²}; CD:=0.002{m};  {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                                {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=10603;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                          {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                        {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                      {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};    {Hier Wolfram eingesetzt} {ansonsten: 2700 für Aluminium}
  rhoFol:=2500{kg/m³};    {Dichte der Kunststoff-Folie}
  dAL:=5000e-6{m};        {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};          {Dicke der Kunststoff-Folie}
  D:=4850{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;        {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=50E3;    {Ohm}                                    {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=160.2E-10{C}; Qp[0]:=0; Qpp[0]:=0;           {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=1.7E-5{sec.};                                                        {Zeitschritte}
  N:=60000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                                 {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                        {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;    { Wait; }

{ Beginn des Rechenprogramms.}
  Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');
  For i:=1 to N do
```

```
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
  Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;

{-----------------------------------------------------------------}
  Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{ Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                        {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
  EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
  Leistung_berechnen;

{-----------------------------------------------------------------}
  Writeln;
  Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
  R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
  L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
  Pin[0]:=0;        {Initialisierung für Leistungsmessung}
  For i:=1 to N do
  begin
    Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
  end;
  If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
  Zugefuehrte_Leistung_mitteln;    Writeln;

{-----------------------------------------------------------------}

  Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}
  R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
  If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  For i:=1 to N do
  begin
    Uin:=U5(i*dt);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
```

```
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
  end;
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;

{-----------------------------------------------------------------}

  Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}
  R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
  If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  For i:=1 to N do
  begin
    Uin:=U6(i*dt);
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
  end;
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;

{-----------------------------------------------------------------}
  Writeln('7.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Impulsbetrieb:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}
  R:=rho*2*pi*SR*SN/AD{Ohm};        {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}    {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                 {Kapazität wieder nach den Vorgaben einstellen.}
  dt:=dtmerk;  N:=Nmerk;            {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}

  x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);           {Coulombkraft}
    xpp[i]:=(Fc+Fd)/m;                                  {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
      Wait; Wait; Halt;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];     {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
  Leistung_berechnen;
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_06:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Param_optimieren_10

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}
    Ianf        : Integer; {Anfang der Leistungsentnahme}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
```

```
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
```

```
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
```

```
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
```

```
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=8;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*0.85; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.10;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                       {Spannung anlegen}
```

```
      end;
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=Ianf to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/(N-Ianf);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=6; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
  mur:=502;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                                {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                         {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                      {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                          {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=19300{kg/m³};                {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                  {Dichte der Kunststoff-Folie}
  dAL:=3800e-6{m};                                {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                          {Dicke der Kunststoff-Folie}
```

```
  D:=86500{N/m};            {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;             {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=20E3;  {früher 10 kiloOhm}                          {Elektrischer Lastwiderstand}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                          {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                        {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);             {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);           {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     {Wait;}

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
```

```
{   Writeln('  t/[sec.]  |           x/[m]         |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                 {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;              {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];     {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
```

```
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                           {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                        {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then begin Reibung:=true; Ianf:=i-1; end;
    Flast:=0;
    If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                           {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);
{##}Utrig[i]:=Uin;
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                      {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];             {elektrische Energie der Spule}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;
  Writeln; Writeln('Mittlere mechan. Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;

{-------------------------------------------------------------}
  Wait;   Wait;
End.
```

Param_optimieren_11

```
Program Param_optimieren_08;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo     : Double;  {Naturkonstanten}
    v           : Double;  {Propagationsgeschwindigkeit der Ströme}
    CA,CD,C     : Double;  {Platten-Kondensator: Plattenfläche, Plattenabstand, Kapazität}
    GG3         : Double;  {Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
    SP3         : Double;  {Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
    UC,UL{,UR}  : Double;  {Spannung über Kondensator, Spule, Widerstand}
    SN,SL,SA,SR : Double;  {Luft-Spule: Windungszahl, Spulenlänge, Querschnittsfläche, Spulen-Radius}
    L           : Double;  {Induktivität der Luft-Spule}
    DL          : Double;  {Drahtlänge des Spulendrahtes}
    epr,mur     : Double;  {Epsilon_r und Mü_r für Kondensator und Spule}
    rho,R       : Double;  {spezifischer und Ohm`scher Widerstand des Spulendrahtes}
    AD          : Double;  {Querschnittsfläche des Spulendrahtes}
    Q,Qp,Qpp    : Array[0..200000] of Double;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Array[0..200000] of Double;  {Auslenkung jeder einzelnen Kondensatorplatte}
    Eel,Emech   : Array[0..200000] of Double;  {Elektrische und mechanische Energie in der Schwingung}
    Esum        : Array[0..200000] of Double;  {Insgesamt in die Schwingung zugeführte Energie als Fkt der Zeit}
    Pin         : Array[0..200000] of Double;  {Elektrische Leistung im Input der Anregung}
    Pent,Eent   : Array[0..200000] of Double;  {Mechanische Leistungs-Entnahme und Energie-Entnahme}
    Pse         : Array[0..200000] of Double;  {Leistungs-Entnahme über die Sekundär-Spule}
    Utrig       : Array[0..200000] of Double;  {Spannung bei Bewegungstriggerung}
    dt          : Double;  {Zeitschritte}
    N           : LongInt; {Anzahl der Zeitschritte insgesamt}
    i,j         : LongInt; {Laufvariable zum Durchzählen der Zeitschritte}
    Abstd       : Integer; {Jeder wievielte Punkte soll geplottet werden}
    rhoAL,rhoFol: Double;  {Dichte von Aluminium und Folie}
    dAL,dFol    : Double;  {Dicke der Aluminium-Kondensatorplatten und der Folie}
    D           : Double;  {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
    m           : Double;  {(mechanische) Masse der Aluminium-Kondensatorplatten}
    omfol,fFol  : Double;  {Eigenkreisfrequenz und Eigenfrequenz der Kondensatorplatten-Schwingung}
    F           : Double;  {Anziehungskraft zwischen den Kondensatorplatten}
    Stern1      : Double;  {Hilfsvariable}
    Fc,Fd       : Double;  {Kräfte: Coulombkraft und Federkraft}
    Flast,Fges  : Double;  {Kraft zur mechanischen Leistungsentnahme und Gesamtkraft}
    MacheFiles  : Boolean; {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
    om          : Double;  {Kreisfrequenz Omega}
    Rlast       : Double;  {Elektrischer Lastwiderstand}
    Uo4,OmE4    : Double;  {Spannungsamplitude und Anregungsfrequenz für erzwungene Schwingung, Teil4}
    R4,L4,C4    : Double;  {Werte spezielle für Teil4}
    Uin         : Double;  {Input-Spannung von Teil 5}
    dtmerk      : Double;  {Zum Merken der Parameter-Eingabe-Werte}
    Nmerk       : LongInt; {Zum Merken der Parameter-Eingabe-Werte}
    Teil7       : Boolean; {Sollen wir gleich Teil_7 rechnen ?}
    etamech,etael : Double; {Wirkungsgrade}
    Pentmittel  : Double;  {Mittlere mechanischen Leistungs-Entnahme}
    ES,xx       : Double;  {Hilfsvariablen für Energie im Schwinger und Mechanische Lastermittlung}
    obUm,unUm   : LongInt; {Umkehrpunkte oben und unten für die Triggerung der Input-Signale}
    Reibung     : Boolean; {Ist Reibung eingeschaltet ?}
    Ianf        : Integer; {Anfang der Leistungsentnahme}
    Rv          : Double;  {Verbraucher-Widerstand zur Leistungsentnahme aus der Spule}
    Psekmittel  : Double;  {Mittlere Leistungs-Entnahme über die Sekundär-Spule}

Label Raus;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure Excel_Datenausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
    A0    : Double;  {abklingende Amplitude der gedämpften Schwingung}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(Q[lv]:14:7,Zahl);
      If (Name='Teil_04.dat') then Str(Q[lv]/C4{Volt}:14:7,Zahl); {Bei Teil 4 ist durch "C4" zu teilen}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (zweite) Funktion die Einhüllende der abklingenden Schwingung:}
      A0:=Q[0]/C/sin(arctan(sqrt(1/L/C-R*R/4/L/L)/(R/2/L)));      {klassische}
      Str(A0*exp(-R/2/L*lv*dt){Volt}:20:10,Zahl);                 {Formeln}
      If (Name='Teil_04.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Ausgabe_Teil3(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Kolumne A: Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      If (Name='Teil_07.dat') then Str(Pent[lv]{Volt}:14:7,Zahl);  {Bei Teil 8: mechanische Entnahme.}
{##}  If (Name='Teil_07.dat') then Str(x[lv]:14:7,Zahl);         {Bei Teil 9: mechanische Auslenkung.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne B: Erste Funktion}
      Str(x[lv]-0.001{Volt}:20:14,Zahl);  {-0.001 Offset zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 5 ist durch "C" zu teilen.}
      If (Name='Teil_06.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 6 ist durch "C" zu teilen.}
      If (Name='Teil_07.dat') then Str(Q[lv]/C{Volt}:14:7,Zahl);  {Bei Teil 7 ist durch "C" zu teilen.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne C: Zweite Funktion}
      Str(Q[lv]*1E6*0.25{Volt}:20:14,Zahl);  {*0.25 Skalierung zur besseren Darstellung}
      If (Name='Teil_05.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Input-Leistung anschauen}
      If (Name='Teil_06.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Input-Leistung anschauen}
      If (Name='Teil_07.dat') then Str(Pin[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Input-Leistung anschauen}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne D: Dritte Funktion: mechanische Energie}
      Str(Emech[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne E: Vierte Funktion: elektrische Energie}
      Str(Eel[lv]{Joule}:20:14,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
```

```
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Kolumne F: Fünfte Funktion: Energiesumme}
      Str(Emech[lv]+Eel[lv]{Joule}:20:14,Zahl);
      If (Name='Teil_05.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 5: Zugeführte Energiesumme anschauen}
      If (Name='Teil_06.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 6: Zugeführte Energiesumme anschauen}
      If (Name='Teil_07.dat') then Str(Esum[lv]{Watt}:14:7,Zahl);  {Bei Teil 7: Zugeführte Energiesumme anschauen}
{##}  If (Name='Teil_07.dat') then Str(Utrig[lv]/100{Watt}:14:7,Zahl); {Bei Teil 9: Input-Spannung}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_Raumenergieausgabe(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt*1e6{nano_sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Spannung über dem Kondensator:}
      Str(x[lv]{Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Excel_eine_Kolumne(Name:String);
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,Name); Rewrite(fout);  {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      Str(x[lv]{Volt}:20:14,Zahl); {Hier trage ich das zu plottende Feld ein.}
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Plapos(z:LongInt):Double;  {Iterative Ermittlung der Position der Kondensatorplatten.}
Var xs    : Double; {Startwert}
    sw    : Double; {Schrittweite}
    an,ab : Boolean;
begin
  xs:=0;
  If z=0 then xs:=CD/2;   {Die Position der beiden Platten liegt bei +/-xs.}
  If z>0 then xs:=x[z-1]; {Dies kann ggf. vom vorigen Arbeitsschritt übernommen werden.}
  sw:=xs/20;
  Repeat
    sw:=sw/10;
    an:=false; ab:=false;
    Repeat
      Fc:=1/4/pi/epo*q[z]*q[z]/(2*xs)/(2*xs);
      Fd:=D*(xs-CD/2);  {Die Feder wird gegenüber CD ausgelenkt.}
      If Fc+Fd>0 then begin xs:=xs-sw; an:=true; end;
```

```
      If Fc+Fd<0 then begin xs:=xs+sw; ab:=true; end;
      If xs<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
    Until (an and ab);
  Until (sw<xs/1e14);
  Plapos:=xs;
end;

Procedure Leistung_berechnen;  {Über dem Lastwiderstand "Rlast", als Integralmittelwert}
Var i   : Integer;
    P   : Double;              {Leistung im Zeitintervall dt}
    Eges: Double;              {Gesamtenergie über den gesamten Zeitraum}
begin
  Eges:=0;
  For i:=0 to N do
  begin
    P:=+Rlast*Qp[i]*Qp[i];
    Eges:=Eges+P*dt;
  end;
  Writeln('Eges= ',Eges, ' Joule in ',N*dt,' sec.');
  Writeln('=> Leistung Pmittel= ',Eges/(N*dt),' Watt  am Lastwiderstand');
  etael:=Eges/(N*dt);
end;

Procedure EnergieMaxima;
Var i                    : Integer;
    EgesM1,EgesOO,EgesP1 : Double;
    Erste,Letzte         : Double;
    Neu                  : Boolean;
begin
  Writeln(' Amplituden-Zunahme: ');  Neu:=true; Erste:=0; Letzte:=0;
  For i:=1 to N-1 do
  begin
    EgesM1:=Emech[i-1]+Eel[i-1];
    EgesOO:=Emech[i+0]+Eel[i+0];
    EgesP1:=Emech[i+1]+Eel[i+1];
    If (EgesM1<=EgesOO)and(EgesOO>=EgesP1) then
    begin
      Writeln(i,':  ',x[i],' => ',EgesOO);
      If Neu then begin Erste:=EgesOO; Neu:=false; end;
      Letzte:=EgesOO;
    end;
  end;
  Writeln; Writeln('Gewonnene mechanische Schwingungsenergie: ');
  Writeln(Letzte-Erste,' Joule => Leistung: ',(Letzte-Erste)/(N*dt),' Watt.');
  etamech:=(Letzte-Erste)/(N*dt);
end;

Procedure Zugefuehrte_Leistung_mitteln;
Var lv   : LongInt;
    Psum : Double;
begin
  Psum:=0;
  For lv:=0 to N do Psum:=Psum+Pin[lv];
  Writeln('Leistungs-Mittel: ',Psum/(N+1),' Watt, Input aus  Spannungsquelle');
  etamech:=etamech/(Psum/(N+1));
  etael:=etael/(Psum/(N+1));
end;

Function U5(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Sinus';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U5:=merk;
end;

Procedure Spannung_auf_Oszi;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
```

```
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U5(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U6(t:Double):Double;
Var merk : Double;
    Pulsform : String;
begin
  Pulsform:='Rechteck';   merk:=0;
  If Pulsform='Sinus' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    merk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.1) then merk:=Uo4;
  end;
  U6:=merk;
end;

Procedure Spannung_auf_Oszi_6;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
{   Zuerst die Zeit als Argument:}
    Str(lv*dt{sec.}:14:10,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Write(fout,chr(9));  {Daten-Trennung}
{   Dann als (erste) Funktion die Input-Spannung:}
    Str(U6(lv*dt){Volt}:14:7,Zahl);
    For j:=1 to Length(Zahl) do
    begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
      If Zahl[j]<>'.' then write(fout,Zahl[j]);
      If Zahl[j]='.' then write(fout,',');
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function U7(t:Double):Double;
Var Umerk       : Double;  {Spannung merken}
    Pulsform    : String;
    Pulsdauer   : LongInt;
    Phasenshift : Double;  {Phasendifferenz zwischen obUm und Spannungs-Impuls}
begin
  Pulsform:='Trigger';   Umerk:=0;
  Phasenshift:=8;      Phasenshift:=Phasenshift/(100*dt);
  If Reibung then Phasenshift:=Phasenshift*1.00; {zur Anpassung der Phasenlage an die Bewegung mit Reibung}
  If Pulsform='Trigger' then
  begin
    Uo4:=0.1;{Volt}    Pulsdauer:=500;       {Wert zum Testen}
    If i<=Pulsdauer then Umerk:=Uo4/10;   {Start-Impuls geben}
    If i>=Pulsdauer then                  {Getriggerte Pulse im Betrieb geben}
    begin
      If xp[i-1]*xp[i]<0 then             {Umkehrpunkte der Bewegung bestimmen}
      begin
        If x[i]>SP3 then begin obUm:=i; Writeln(i,' obUM bei ',x[i]); end;
        If x[i]<SP3 then begin unUm:=i; Writeln(i,' unUM'); end;
      end;
```

```
      If (i>=obUm+Phasenshift)and(i<=(obUm+Pulsdauer+Phasenshift)) then
      begin
        Umerk:=Uo4;                        {Spannung anlegen}
      end;
    end;
  end;
  If Pulsform='Sinus' then
  begin
    Uo4:=0.05;{Volt}    OmE4:=4.6;{s^-1}     {Werte zum Testen}
    Umerk:=Uo4*sin(OmE4*t);
  end;
  If Pulsform='Rechteck' then
  begin
    Uo4:=12;{Volt}    OmE4:=34.6;{s^-1}     {Werte zum Testen}
    Umerk:=0;
    If (0<sin(OmE4*t))and(sin(OmE4*t)<0.25) then Umerk:=Uo4;
  end;
{ If i>N/2 then Umerk:=0; {Kann bei Bedarf zugeschaltet werden: Nur den Anfang der Bewegung anregen.}
  U7:=Umerk;
end;

Procedure Spannung_auf_Oszi_7;
Var fout  : Text;    {Daten-File zum Aufschreiben der Ergebnisse}
    Zahl  : String;
    lv,j  : Integer; {Laufvariable}
begin  {Daten für Excel aufbereiten und ausgeben:}
  Assign(fout,'U_von_t.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to N do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
{     Zuerst die Zeit als Argument:}
      Str(lv*dt{sec.}:14:10,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung}
{     Dann als (erste) Funktion die Input-Spannung:}
      Str(U7(lv*dt){Volt}:14:7,Zahl);
      For j:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[j]<>'.' then write(fout,Zahl[j]);
        If Zahl[j]='.' then write(fout,',');
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Mechanische_Energie_Entnahme;
Var lv : LongInt;
begin
  Pentmittel:=0;
  For lv:=Ianf to N do Pentmittel:=Pentmittel+Pent[lv];
  Pentmittel:=Pentmittel/(N-Ianf);
end;

Procedure Insgesamt_zugefuehrte_Energie_berechnen;
Var lv : LongInt;
begin
  Esum[0]:=Pin[0]*dt;
  For lv:=1 to N do Esum[lv]:=Esum[lv-1]+Pin[lv]*dt;
end;

Procedure Leistungsmittel_ueber_Sekundaerspule;
Var lv : LongInt;
begin
  Psekmittel:=0;
  For lv:=Ianf to N do Psekmittel:=Psekmittel+Pse[lv];
  Psekmittel:=Psekmittel/(N-Ianf);
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }
{ Allgemeine Werte: }
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  v:=Sqrt(1/muo/epo){m/s};     {Zunächst Lichtgeschw. als Bewegungsgeschw. der Ladungen}
  Abstd:=8;                    {Jeder wievielte Punkte soll geplottet werden}
{ Kondensator: }
  CA:=1E5; {0.1*0.1 m²}; CD:=0.002{m}; {Kondensator-Geometrie, Plattenfläche, Plattenabstand}
  epr:=3;                                              {Dielektrikum im Kondensator}
  C:=epo*epr*CA/CD;                    {Kapazität des unverformten Platten-Kondensators}
{ Spule: }
  SN:=34600; SL:=0.08{m}; SR:=0.05{m}; SA:=pi*SR*SR{m²};              {Spulen-Geometrie}
```

```
  mur:=1200;                        {Spulenkern ist nötig, zur Abstimmung der Frequenz}
  L:=muo*mur*SN*SN*SA/SL;                                              {Induktivität}
  rho:=1.7E-8{Ohm*m}; {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  AD:=pi*0.0002*0.0002{m²};                    {Querschnittsfläche des Spulendrahtes}
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Ohm`scher Widerstand des Spulendrahtes}
  DL:=SN*2*pi*SR;                                  {Drahtlänge des Spulendrahtes}
{ Mechanische Schwingung der Kondensatorplatten:}
  rhoAL:=2700{kg/m³};               {19300 für Wolfram, ansonsten: 2700 für Aluminium}
  rhoFol:=2000{kg/m³};                                  {Dichte der Kunststoff-Folie}
  dAL:=5000e-6{m};                                {Dicke der Metall-Kondensatorplatten}
  dFol:=1e-6{m};                                    {Dicke der Kunststoff-Folie}
  D:=12000{N/m};           {Federsteifigkeit der Federn zwischen den Kondensatorplatten}
  m:=CA*dAL*rhoAL+CA*dFol*rhoFol; {(mechanische) Masse der Aluminium-Kondensatorplatten}
  omFol:=Sqrt(D/m);        {Schwingungs-Eigenkreisfrequenz der Kondensatorplatten_Folie}
  fFol:=omFol/2/pi;             {Schwingungseigenfrequenz der Kondensatorplatten_Folie}
{ Bewußte Erzeugung von Leistung:}
  Rlast:=1E-10;  {früher 10 kiloOhm}                      {Elektrischer Lastwiderstand}
  Rv:=300E6;                {Verbraucher-Widerstand zur Leistungsentnahme aus der Spule}
  { Start der elektrischen Schwingung: }
  Q[0]:=1E-20; {160.2E-10{Coulomb};  Qp[0]:=0; Qpp[0]:=0;    {Ladung auf dem Kondensator zu Beginn}
  UC:=Q[0]/C{V};       {Spannung über dem Kondensator zu Beginn, das Dielektrikum isoliert}
  dt:=100E-6{sec.};                                                    {Zeitschritte}
  N:=200000;                                          {Anzahl der Zeitschritte insgesamt}
  dtmerk:=dt;  Nmerk:=N;                                    {Merken der Input-Werte}
{ Start der mechanischen Schwingung: }
  x[0]:=Plapos(0);              {Iterative Ermittlung der Position der Kondensatorplatten.}
  GG3:=x[0];{Gleichgewichtsposition der flexiblen Platten, Teil 3, Federkraft=Coulombkraft}
  SP3:=CD/2;{Vorgabe:Plattenabstand mit flexiblen Platten, Teil 3, mechanische Vorspannung}
  F:=1/4/pi/epo*Q[0]*Q[0]/(2*x[0])/(2*x[0]);            {Anziehung nach dem Coulomb-Gesetz}
                                                {Der Ort jeder Platte liegt bei CD/2+x[i]}
  xp[0]:=0; xpp[0]:=0;                       {Festhalten der Platten bis zum Zeitpunkt t=0}
  MacheFiles:=true;       {Sollen die Ergebnisse auf die Magnetplatte geschrieben werden ?}
  Teil7:=true;
  obUm:=0; unUm:=0; {Initialisierung der Umkehrpunkte für die Triggerung der Input-Impulse}
  Reibung:=false;

  { Anzeigen der Startwerte:}
  Writeln('DFEM-Berechnung des LC - Schwingkreises:'); Writeln;
  Writeln('epo=',epo:20,';  muo=',muo:20,';  v=',v:20);
  Writeln('C=',C:20,' Farad;     L=',L:20,' Henry');
  Writeln('Klass. Schwingkreis, Eingenfrequ. fo=2*pi/Sqrt(L*C)=',2*pi/Sqrt(L*C),' Hz');
  Writeln(' ==> Schwingungsdauer T=1/fo=',2*pi*Sqrt(L*C),' sec.');
  Writeln('Ohm`scher Widerstand des Spulendrahtes:',R,' Ohm');
  Writeln('Drahtlaenge des Spulendrahtes:',DL,' Meter');
  Writeln('Querschnittsfläche des Spulendrahtes:',AD*1e6:10:5,' mm^2');
  Writeln('Volumen der Spule: ',DL*AD*1E6:10:5,' cm^3');
  Writeln('Gewicht der Spule: ',DL*AD*1E6*8.92:10:5,' Gramm');  {Dichte Cu: 8.92 g/cm^3}
  Writeln('Spannung ueber dem Kondensator zu Beginn:',UC:12:5,' Volt');
  Writeln('Ges. Zeitspanne der Berechnung: ',N*dt,' sec. in ',N,' Schritten');
  Writeln; Writeln('Daten der mechanischen Schwingung der Kondensatorplatten:');
  Writeln('Masse der Kondensatorplatten m= ',m*1000:10:5,' Gramm');
  Writeln('Schwingungseigenfrequenz der Kondensatorplatten: fFol= ',fFol:10:7,' Hz.');
  Writeln('Anziehung jeder Kondensatorplatte zu Beginn: Kraft F= ',F,' N');
  Writeln('Verformung jeder Kondensatorplatte zu Beginn: F/D= ',F/D,' m');
  Writeln('Plattenposition der ungeladenen Kondensatorplatten: ',CD/2);
  Writeln('Plattenposition, geladen, zu Beginn: X[0]: ',X[0]);
  Writeln('Genauigkeit der Plattenposition => Differenzkraft: ',Fc+Fd,' N');
  Writeln('Startposition der Platten für die Schwingg, Teil 3: ',SP3:10:7,' m');
  Writeln('Kapazitaet des unverformten Kondensators:  C= ',epo*epr*CA/CD,' Farad');
  Writeln('Kapazitaet des verformten Kondensators: C[0]= ',epo*epr*CA/(2*x[0]),' Farad');
  Writeln('Dabei: Erhöhung der Kapazitaet um ',epo*epr*CA*(1/2/x[0]-1/CD),' Farad');
  Writeln('Gesamtdauer der Berechnung: ',N*dt,' sec.');
  Writeln;     Wait;

{ Beginn des Rechenprogramms.}
  If Not(Teil7) then
  begin
    Writeln('1.Teil -> Klassische Harmonische Schwingung, ohne Dämpfung:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');
    For i:=1 to N do
    begin
      UC:=Q[i-1]/C; UL:=-UC;
      Qpp[i]:=UL/L;
      Qp[i]:=Qp[i-1]+Qpp[i]*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_01.dat'); Writeln;
  end;
{-------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('2.Teil -> Klassische Gedaempfte Schwingung, mit Ohm`schem Widerstand:');
    Writeln('  t/[sec.]  |  Uc/[V] | ');    { R:=2000; {Erhöhter Widerstandswert zum Testen}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];
```

```
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt); }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;    {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_02.dat'); Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('3.Teil -> Schwingung mit geladenem Kondensator noch ohne Raumenergie-Wandlung');
{   Writeln('  t/[sec.]  |           x/[m]          |        Q[i]'); }
    x[0]:=SP3;       {Startposition der Kondensatorplatten für die mechanische Schwingung}
    For i:=1 to N do
    begin
      Fd:=-D*(x[i-1]-CD/2);                              {Federkraft gegenüber CD}
      Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
      xpp[i]:=(Fc+Fd)/m;                                        {Beschleunigung}
      xp[i]:=xp[i-1]+xpp[i]*dt;
      x[i]:=x[i-1]+xp[i]*dt;
      Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
      Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
      If x[i]<=1e-10 then
      begin
        Writeln ('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen.');
        Wait; Wait; Halt;
      end;
      C:=epo*epr*CA/(2*x[i]);
      Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];
      Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;
      Q[i]:=Q[i-1]+Qp[i]*dt;
      Eel[i]:=1/2*Q[i]*Q[i]/C;               {elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];      {elektrische Energie der Spule}
{     Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
    end;
    Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_03.dat'); Writeln;
    EnergieMaxima; Writeln; Writeln(' UND AM LASTWIDERSTAND:');
    Leistung_berechnen;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln;
    Writeln('4.Teil -> Klass. erzwungene Schwingung, Ohm-Widerstand und Sinus-Anregung');
    R4:=70;{Ohm}    Uo4:=12;{Volt}    OmE4:=8680;{s^-1}     {Werte zum Testen}
    L4:=10E-3;{Henry}    C4:=1E-6;{Farad}                   {Werte zum Testen}
    Pin[0]:=0;        {Initialisierung für Leistungsmessung}
    For i:=1 to N do
    begin
      Qpp[i]:=-1/L4/C4*Q[i-1]-R4/2/L4*Qp[i-1]+Uo4/L4*sin(OmE4*i*dt);
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R4/L4*dt);  }     {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R4/2/L4*Qp[i-1])*dt;     {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uo4*sin(OmE4*i*dt)*Qp[i];    {Berechnung der zugeführten Leistung}
    end;
    If MacheFiles then Excel_Datenausgabe('Teil_04.dat'); Writeln;
    Zugefuehrte_Leistung_mitteln;    Writeln;
  end;
{-----------------------------------------------------------------}

  If Not(Teil7) then
  begin
    Writeln('5.Teil -> Erzwungene Schwingung mit Energie-Kontrolle.');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U5(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_05.dat');  Writeln;
  end;
{-----------------------------------------------------------------}
```

```
  If Not(Teil7) then
  begin
    Writeln('6.Teil -> Erzwungene Schwingung mit beliebigem Signal-Input (Impulsbetrieb)');
    Pin[0]:=0;          {Initialisierung für Leistungsmessung}
    R:=70;{Ohm} L:=10E-3;{Henry} C:=1E-6;{Farad}  dt:=1E-7{sec.}; N:=30000; {Werte zum Testen}
    If MacheFiles then Spannung_auf_Oszi_6;   {Graphisches Anzeigen des Spannungs-Input-Signals}
    For i:=1 to N do
    begin
      Uin:=U6(i*dt);
      Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1]+Uin/L; {Hier kann U(t) eine beliebige Signalform haben}
{     Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }  {alternative einfachere Näherung}
      Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
      Q[i]:=Q[i-1]+Qp[i]*dt;
{     Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
      Pin[i]:=Uin*Qp[i];                {Berechnung der zugeführten Leistung}
      Eel[i]:=1/2*Q[i]*Q[i]/C;          {Elektrische Energie des Kondensators}
      Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];  {Elektrische Energie der Spule}
      If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
    end;
    Zugefuehrte_Leistung_mitteln;
    Insgesamt_zugefuehrte_Energie_berechnen;
    If MacheFiles then Excel_Ausgabe_Teil3('Teil_06.dat');  Writeln;
  end;
{-----------------------------------------------------------------}

  Writeln('8.Teil -> Mein Raumenergie-LCR-Kreis, Zeit-stabil durch Leistungsentnahme:');
  Pin[0]:=0;          {Initialisierung für Leistungsmessung}       Fges:=0;    Pse[0]:=0;
  R:=rho*2*pi*SR*SN/AD{Ohm};                 {Widerstand wieder nach den Vorgaben einstellen.}
  L:=muo*mur*SN*SN*SA/SL;{Henry}           {Induktivität wieder nach den Vorgaben einstellen.}
  C:=epo*epr*CA/CD;                          {Kapazität wieder nach den Vorgaben einstellen.}
  Q[0]:=0;  Qp[0]:=0; Qpp[0]:=0;                    {Keine Anfangs-Ladung bei Impulsbetrieb}
  dt:=dtmerk;  N:=Nmerk;                          {Wiederherstellen der Input-Werte-Vorgaben}
  If MacheFiles then Spannung_auf_Oszi_7;   {Graphisches Anzeigen des Spannungs-Input-Signals}
  x[0]:=SP3;             {Startposition der Kondensatorplatten für die mechanische Schwingung}
{##}Utrig[0]:=0;
  For i:=1 to N do
  begin
    Fd:=-D*(x[i-1]-CD/2);                                {Federkraft gegenüber CD}
    Fc:=-Q[i-1]*Q[i-1]/4/pi/epo/(2*x[i-1])/(2*x[i-1]);            {Coulombkraft}
{   Jetzt kommt die antreibende Kraft und außerdem die Kraft zur mechanischen Leistungsentnahme : }
    Fges:=Fc+Fd;                                          {Zuerst hier die antreibende Systemkraft}
{   Es folgt eine (mehrzeilige) Vorgabe einer geeigneten Last-Kraft: }
    ES:=(1/2*D*(x[i-1]-CD/2)*(x[i-1]-CD/2)+1/2*(2*m)*xp[i-1]*xp[i-1]);    {Leistung, Hilfsvariable}
    If x[i-1]<0.75*CD/2 then begin Reibung:=true; Ianf:=i-1; end;
    Flast:=0;  If Reibung then Flast:=0.88*Fges; {Wir ziehen einen Anteil der Gesamtkraft heraus.}
    {Die Leistungsentnahme soll später elektrisch als der Spule über einen Trafo geschehen.}
    Fges:=Fges-Flast;
{   Ende der Kraft-Berechnung. Es wurden die Systemkräfte und die Last-Kraft berechnet.}
{   Jetzt beginnt die Lösung des gekoppelten Differentialgleichungs-Systems: }
    xpp[i]:=Fges/m;                           {Beschleunigung}
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Pent[i]:=Flast*Abs(xp[i]);
    Eent[i]:=Pent[i]*dt;
    Emech[i]:=1/2*D*(x[i]-CD/2)*(x[i]-CD/2);   {Federspann-Energie}
    Emech[i]:=Emech[i]+1/2*(2*m)*xp[i]*xp[i];  {Kinetische Energie}
    If x[i]<=1e-10 then
    begin
      Writeln('Plattenberuehrung. Coulombkraft ist zu stark. Algorithmus abgebrochen');
      Writeln('in Schritt Nr. ',i,' von ',N);
      For j:=i to N do
      begin
        x[j]:=0; xp[j]:=0; xpp[j]:=0;  Q[j]:=0; Qp[j]:=0; Qpp[j]:=0;
        Eel[j]:=0; Emech[j]:=0; Esum[j]:=0; Pin[j]:=0;
        N:=i-1; {Weiter soll ich nicht anzeigen.}
      end;
      Wait; Wait; {Halt;} Goto Raus;
    end;
{   Hier beginnt die Berechnung der elektrischen Schwingung}
    C:=epo*epr*CA/(2*x[i]);
    Uin:=U7(i*dt);  {Diese Spannung soll die Schwingung anregen.}
{##}Utrig[i]:=Uin;
{   Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1]+Uin/L;   {Früher, ohne Leistungsentnahme aus der Spule}
    Qpp[i]:=-1/C/(R+Rlast+Rv)*Qp[i-1]-1/L/C*Rv/(R+Rlast+Rv)*Q[i-1]-(R+Rlast)/L/2*Rv/(R+Rlast+Rv)*Qp[i-
1]+1/L*Rv/(R+Rlast+Rv)*Uin;
                                                                          {Jetzt mit Leistungsentnahme aus
der Spule}
{   Qp[i]:=(Qp[i-1]+Qpp[i]*dt)/(1+R/L*dt);  }   {uralter Test: alternative einfachere Näherung}
    Qp[i]:=Qp[i-1]+(Qpp[i]-(R+Rlast)/2/L*Qp[i-1])*dt;  {vgl. s=1/2*a*t^2}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    Pin[i]:=Uin*Qp[i];                          {Berechnung der zugeführten Leistung}
    Eel[i]:=1/2*Q[i]*Q[i]/C;                   {elektrische Energie des Kondensators}
    Eel[i]:=Eel[i]+1/2*L*Qp[i]*Qp[i];                 {elektrische Energie der Spule}
    Pse[i]:=Abs(L*Qp[i]*Qpp[i]);             {Über Joch und Sekundärspule entnommene Leistung}
    If Eel[i]<0 then begin Writeln('Da timmt wat nich ?!?'); Wait; Halt; end;
{   Writeln(i*dt:11:9,' | ',x[i],' |',Q[i]);}
  end;
```

```
Raus:
  Emech[0]:=Emech[1];  Eel[0]:=Eel[1];  {In Näherung, damit es beim Plotten nicht stört.}
  Writeln('Nachfolgend die Datenkontrolle analog zu Teil_03:');
  EnergieMaxima; Writeln; Writeln('AM LASTWIDERSTAND ENTNOMMEN:');
  Leistung_berechnen;
  Writeln; Writeln('Zugfuehrte Leistung aus dem Spannungs-Input:');
  Zugefuehrte_Leistung_mitteln;
  Insgesamt_zugefuehrte_Energie_berechnen;  Writeln;
  Writeln('Wirkungsgrad mechanisch Eta_mech: ',etamech:10:4,' (* 100 %) gespeichert');
  Writeln('Wirkungsgrad elektrisch Eta_el:   ',etael:10:4,' (* 100 %)');
  Mechanische_Energie_Entnahme;     Writeln;
  Writeln('Mittlere mechanische Leistungs-Entnahme in Teil 8: ',Pentmittel,' Watt');
  If MacheFiles then Excel_Ausgabe_Teil3('Teil_07.dat');  Writeln;
  Ianf:=1; Leistungsmittel_ueber_Sekundaerspule;
  Writeln('Mittlere Leistungs-Entnahme ueber Sekundaerspule : ',Psekmittel,' Watt');

{-----------------------------------------------------------------}
  Wait;   Wait;
End.
```

Es folgt Teil 3

Var_L_001

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Var epo,muo  : Double;  {Naturkonstanten}
    c        : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge}
    qp       : Double;  {Strom, wird später zum Array}
    a        : Double;  {Aufpunkt, Ort an dem das Magnetfeld berechnet werden soll.}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Function H(ax:Double):Double; {Berechnung der Magnetfeld-Stärke an der Position "a" in x-Richtung}
Var i,j             : LongInt;
    Hij,sum,kl1,kl2 : Double;
begin
  sum:=0;
  For i:=1 to n do
  begin
    For j:=1 to m do
    begin
      kl1:=Sqr(di/2+(i-0.5)*Dd);
      kl2:=Sqr(ax-(j-0.5)*Dd);
      Hij:=qp*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
      sum:=sum+Hij;
    end;
  end;
  H:=sum;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  c:=Sqrt(1/muo/epo){m/s};     {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',c, ' m/s');
  m:=250;                      {Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=20;                       {Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=0.2E-3; {Meter}          {Drahtdurchmesser}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser}
{ Abgeleitete Werte: Zum bequemeren Rechnen abgeleitet aus den Input-Werten:}
  ls:=m*Dd; bs:=n*Dd;          {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                 {Spulen-Aussendurchmesser}
  dk:=di;                      {Magnetkern-Durchmesser -> Der Magnetkern füllt die Spule aus.}
  lk:=ls;                      {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Spulenkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');

{ Hier beginnt das Rechenprogramm:}      Writeln;
  Writeln('Hier beginnen die eigentlichen Berechnungen:');
  a:=0.02;                           {Aufpunkt}
  qp:=1;                          {Strom}
  Writeln('H = ',H(a),' A/m');    {Magnetfeld}



{-------------------------------------------------------------------}
  Wait;   Wait;
End.
```

Var_L_002

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzP=24; {200000} {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzP] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    c        : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i        : LongInt; {nur für Testzwecke: Laufvariable}
    F1,F2    : Feld;    {nur für Testzwecke: Arrays für Daten-Output in Excel}
    xkern    : Double;  {nur für Testzwecke: x-Position des Magnetkerns}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
```

```
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
```

```
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  c:=Sqrt(1/muo/epo){m/s};     {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',c, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=15;                       {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=3;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=0.2E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=10;                      {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=2;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di-0.002;                {Magnetkern-Durchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
{ AnzP -> Dieser Parameter wird oben als Konstante eingegeben: Anzahl der Zeitschritte insgesamt}
  dt:=0.0005; {sec.}           {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;          {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                 {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                   {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  dki:=dk-2*nk*Dd;             {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);         {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');

{ Hier beginnt das Rechenprogramm:}      Writeln;
  Writeln('Hier beginnen die eigentlichen Berechnungen:');

  Writeln('Test: Kraft zwischen zwei Leiterschleifen');
  For i:=0 to AnzP do
  begin   {Kraft zwischen zwei Leiterschleifen (in x-Richtung)}
    F1[i]:=Fx(0.2,0.2,1.0,1.0,0.05,0.05+i*dt);   {Zeitschritte als Ortsschritte mißbraucht}
    Writeln('Position = ',0.05+i*dt:9:5,'m => Kraft = ',F1[i],' Newton.');
  end;

  Writeln; Writeln('Test: Kraft zwischen Spule und Magnetkern, je nach x-Position des Kerns:');
  For i:=0 to AnzP do
  begin   {Kraft auf den Magnetkern in der Spule (in x-Richtung)}
    xkern:=-0.0055+i*dt;
    F2[i]:=Fges(xkern);
    Writeln('Kern-Pos = ',xkern*100:7:4,'cm => Fges = ',F2[i],'N');   {Zeitschritte als Ortsschritte mißbraucht}
  end;

  ExcelAusgabe('test.dat',2,F1,F2,F1,F1,F1,F1,F1,F1,F1,F1,F1,F1);


  {------------------------------------------------------------}
  Wait;   Wait;
End.
```

Var_L_003

```
Program Var_L_003;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzP=24; {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzP] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    c        : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i        : LongInt; {nur für Testzwecke: Laufvariable}
    F1       : Feld;    {nur für Testzwecke: Arrays für Daten-Output in Excel}
    AnzFit   : Integer; {Anzahl der Datenpaare für den Kraft-Weg-Fit}
    X,Y,Z    : Array [0..200] of Double; {XY-Datenpaare für Kraft-Weg-Fit und Z=Regression für Y}
    SW       : Double;  {Schrittweite für die Bewegung des Kraft-Weg-Fits}
    Schritte : Integer; {Anzahl der Schritte von der Gleichgewichtsposition bis zum Kraftmaximum}
    a        : Array [1..3] of Double;  {Parameter für die Regressionsfunktion}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzFit do  {von "plotanf" bis "plotend"}
```

```
  begin
    For j:=1 to 3 do
    begin   {Kolumnen drucken}
      If j=1  then Str(X[lv]:17:11,Zahl);
      If j=2  then Str(Y[lv]:17:11,Zahl);
      If j=3  then Str(Z[lv]:17:11,Zahl);
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Procedure Spulengeometrie_ggf_uebernehmen;
begin


end;

Function Fr(x:Double):Double;  {Kraft durch Regressions-Kurve berechnen}
Var merk,lnyw : Double;
    Vorz      : Real;     {Vorzeichen der Ortsangabe}
    rupos     : Double;   {Ruhelage-Position}
begin    {Kraftberechnung wurde nur für positive x-Werte bestimmt und punktsymmetrisch}
  rupos:=(ls-lk)/2;  merk:=0;
  Vorz:=-1;  If x<rupos then Vorz:=+1;
  If Abs(x-rupos)>=1E-10 then
  begin
    x:=Abs(x-rupos);
    lnyw:=ln(a[1])+a[2]*ln(x);    {nach *1 von S.45}
    merk:=exp(lnyw)*exp(-a[3]*x);
  end;
  Fr:=Vorz*merk;   {Spiegelung am Symmetriepunkt des Graphen}
end;

Function QS:Double;  {Quadratische Abweichungssumme}
Var lv   : Integer; {Laufvariable}
    Qsum : Double;  {Quadratsumme}
begin
  Qsum:=0;
  For lv:=0 to AnzFit do Qsum:=Qsum+Sqr(Y[lv]-Fr(X[lv]));
  QS:=Qsum;
end;

Procedure NichtlineareRegression;
Var Sx,Sy,Sxx,Sxy : Double;  {Summen für die Regression der a1,a2}
    lv,lauf       : Integer; {Laufvariable}
    delA          : Array [1..3] of Double;  {Parameter-Variation für die Regressionsfunktion}
    QSmerk        : Double; {Zur Optimierung der Quadrat-Abweichungssumme}
    change        : Boolean; {wurden Parameter verändert ?}
Procedure Omptimiere_A(nr:Integer);
Var altA : Double;
    QSmit,QSminus,QSplus : Double; {Zur Optimierung der Quadrat-Abweichungssumme}
begin
  altA:=a[nr]; QSmit:=QS;
  a[nr]:=altA+delA[nr]; QSplus:=QS;
  a[nr]:=altA-delA[nr]; QSminus:=QS;
  If (QSplus<QSmit)  then a[nr]:=altA+delA[nr];
  If (QSminus<QSmit) then a[nr]:=altA-delA[nr];
  change:=(a[nr]<>altA);
end;
begin
  Writeln ('Nichtlineare Regression der Kraft-Weg-Kennlinie.');
  For lv:=0 to AnzFit do X[lv]:=X[lv]-(ls-lk)/2;  {Kraftnullpunkt in den Symmetriepunkt schieben}
{ Startwerte für die Parameter der Kurve suchen:}        change:=false;
  Sx:=0;  Sy:=0;  Sxx:=0;  Sxy:=0;  {Vorbereitung der Startwerte der Wurzelfunktion}
  For lv:=Round(AnzFit/2+1) to Round(AnzFit/2+Schritte) do
  begin
    Sx:=Sx+ln(Abs(X[lv]));
    Sy:=Sy+ln(Abs(Y[lv]));
    Sxx:=Sxx+Sqr(ln(Abs(Y[lv])));
    Sxy:=Sxy+ln(Abs(X[lv]))*ln(Abs(Y[lv]));
  end;
  a[2]:=Sxy-Sx*Sy/Schritte;                          {Geradensteigung, linear, nach *1 von S.45}
  a[1]:=Sy/Schritte-Sx*a[2]/Schritte; a[1]:=exp(a[1]);    {Achsenabschnitt, entlogarithmieren, nach *1 von S.45}
  a[3]:=1/X[AnzFit]*(ln(a[1])-a[2]*ln(X[AnzFit]));
  Writeln('Startwerte: a1 = ',a[1]:12:7,',  a2 = ',a[2]:12:7,',  a3 = ',a[3]:12:7);
{ Anpassen der Parameter an die Regressionskurve:}
  For lv:=0 to AnzFit do X[lv]:=X[lv]+(ls-lk)/2;  {Kraftnullpunkt in seine Ausgangslage zurückschieben}

{ Hier muß ich jetzt das Genau-Togglen der Parameter "a1,a2,a3" einbauen;}
{ Writeln('Kontrolle zu Beginn: QuadSum = ',QS); }
{ For lv:=0 to AnzFit do Writeln(lv:3,': x = ',X[lv]*100:8:5,'cm  => Fr = ',Fr(X[lv]),'N'); }
```

```
  For lv:=1 to 3 do delA[lv]:=a[lv]/100;  {Startwerte für die Optimierung der Parameter}
  Repeat
    Repeat
      QSmerk:=QS;
      For lv:=1 to 3 do
      begin
        lauf:=0;
        Repeat Omptimiere_A(lv); lauf:=lauf+1; Until (Change=false)or(lauf>5);
        Writeln('QS=',QS,' => a1=',a[1]:12:7,', a2=',a[2]:12:7,', a3=',a[3]:12:7);
      end;
    Until QS>0.99*QSmerk;
    For lv:=1 to 3 do delA[lv]:=delA[lv]/2;
  Until (delA[1]/A[1]<1E-8)and(delA[2]/A[2]<1E-8)and(delA[2]/A[2]<1E-8);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
```

```
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  c:=Sqrt(1/muo/epo){m/s};     {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',c, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=15;                       {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=3;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=0.2E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=10;                      {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=2;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di-0.002;                {Magnetkern-Durchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
{ AnzP -> Dieser Parameter wird oben als Konstante eingegeben: Anzahl der Zeitschritte insgesamt}
  dt:=0.0005; {sec.}           {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;          {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                 {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                   {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  dki:=dk-2*nk*Dd;             {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);         {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
{ Parameter für den Ablauf des Software-Algorithmus: }
{ AnzFit:=20;   {Anzahl der Punkt für den Kraft-Weg-Fit = (2*AnzFit+1) = 41. -> Initialisierung im Upgm.}
{ Schritte:=5;  {Anzahl der Kraft-Weg-Schritte bis zum Kraft-Maximum -> Initialisierung im Upgm.}

{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');

{ Hier beginnt das Rechenprogramm:}      Writeln;
  Writeln('Hier beginnen die eigentlichen Berechnungen:');

  Writeln('Test: Kraft zwischen zwei Leiterschleifen');
  For i:=0 to AnzP do
  begin   {Kraft zwischen zwei Leiterschleifen (in x-Richtung)}
    F1[i]:=Fx(0.2,0.2,1.0,1.0,0.05,0.05+i*dt);   {Zeitschritte als Ortsschritte mißbraucht}
    Writeln('Position = ',0.05+i*dt:9:5,'m => Kraft = ',F1[i],' Newton.');
  end;
  ExcelAusgabe('test.dat',1,F1,F1,F1,F1,F1,F1,F1,F1,F1,F1,F1,F1);

  Writeln; Writeln('Test: Kraft zwischen Spule und Magnetkern, je nach x-Position des Kerns:');
  AnzFit:=20;  {Halbe (!) Anzahl der Punkte für den Kraft-Weg-Fit = (2*AnzFit+1) = 41.}
  Schritte:=5; {Anzahl der Daten-Schritte bis zum Kraft-Maximum}
  X[AnzFit]:=(ls-lk)/2;         {Ruhelageposition ist immer bei X[20]}
  SW:=(ls-X[AnzFit])/Schritte;  {Schrittweite für die Kraftberechnung}
```

```
  For i:=1 to AnzFit do
  begin
    X[AnzFit+i]:=X[AnzFit+i-1]+SW;
    X[AnzFit-i]:=2*X[AnzFit]-X[AnzFit+i];
    If i=Round(1.6*Schritte) then SW:=SW*2;
    If i=Round(2.8*Schritte) then SW:=SW*2;  {Schrittweite stufenweise erhöhen}
  end;
  AnzFit:=2*AnzFit;      {Jetzt wird AnzFit auf seinen endgültigen Werte gebracht.}
  For i:=0 to AnzFit do
  begin
    Y[i]:={Fges}(X[i]);  {Für die echte Kraftberechnung muß ich das "Fges" hereinnehmen.}
    Z[i]:=0; {Der Regressions-Fit kommt später}
    Writeln(i:3,': ',X[i]*100:7:4,' cm -> ',Y[i],' N');
  end;
  Kraftwerte_ins_Excel;
  X[00]:=-0.0215;     Y[00]:=+0.001003581;
  X[01]:=-0.0195;     Y[01]:=+0.001313429;
  X[02]:=-0.0175;     Y[02]:=+0.001742661;
  X[03]:=-0.0155;     Y[03]:=+0.002346609;
  X[04]:=-0.0135;     Y[04]:=+0.003211403;
  X[05]:=-0.0115;     Y[05]:=+0.004475725;
  X[06]:=-0.0095;     Y[06]:=+0.006374093;
  X[07]:=-0.0085;     Y[07]:=+0.007681951;
  X[08]:=-0.0075;     Y[08]:=+0.009332552;
  X[09]:=-0.0065;     Y[09]:=+0.011446280;
  X[10]:=-0.0055;     Y[10]:=+0.014198576;
  X[11]:=-0.0045;     Y[11]:=+0.017835830;
  X[12]:=-0.0035;     Y[12]:=+0.022607882;
  X[13]:=-0.0030;     Y[13]:=+0.025368581;
  X[14]:=-0.0025;     Y[14]:=+0.028071156;
  X[15]:=-0.0020;     Y[15]:=+0.030012747;
  X[16]:=-0.0015;     Y[16]:=+0.030040590;
  X[17]:=-0.0010;     Y[17]:=+0.027059821;
  X[18]:=-0.0005;     Y[18]:=+0.020630968;
  X[19]:=+0.0000;     Y[19]:=+0.011186763;
  X[20]:=+0.0005;     Y[20]:=+0.000000000;
  X[21]:=+0.0010;       Y[21]:=-0.011186763;
  X[22]:=+0.0015;     Y[22]:=-0.020630968;
  X[23]:=+0.0020;     Y[23]:=-0.027059821;
  X[24]:=+0.0025;     Y[24]:=-0.030040590;
  X[25]:=+0.0030;     Y[25]:=-0.030012747;
  X[26]:=+0.0035;     Y[26]:=-0.028071156;
  X[27]:=+0.0040;     Y[27]:=-0.025368581;
  X[28]:=+0.0045;     Y[28]:=-0.022607882;
  X[29]:=+0.0055;     Y[29]:=-0.017835830;
  X[30]:=+0.0065;     Y[30]:=-0.014198576;
  X[31]:=+0.0075;     Y[31]:=-0.011446280;
  X[32]:=+0.0085;     Y[32]:=-0.009332552;
  X[33]:=+0.0095;     Y[33]:=-0.007681951;
  X[34]:=+0.0105;     Y[34]:=-0.006374093;
  X[35]:=+0.0125;     Y[35]:=-0.004475725;
  X[36]:=+0.0145;     Y[36]:=-0.003211403;
  X[37]:=+0.0165;     Y[37]:=-0.002346609;
  X[38]:=+0.0185;     Y[38]:=-0.001742661;
  X[39]:=+0.0205;     Y[39]:=-0.001313429;
  X[40]:=+0.0225;     Y[40]:=-0.001003581;
  Kraftwerte_ins_Excel;

{ Hier beginnt die Berechnung der nichtlinearen Regressionskurve: }
  NichtlineneareRegression;
  For i:=0 to AnzFit do Writeln(i:3,': ',X[i]*100:7:4,' cm -> ',Y[i],' N, ~',Fr(X[i]));  {Werte kontrollieren}
  For i:=0 to AnzFit do Z[i]:=Fr(X[i]);  {Der Regressions-Fit kommt jetzt}
  Kraftwerte_ins_Excel;

{-------------------------------------------------------------------}
  Writeln('Fertig, Adele.');  Wait;  Wait;
End.
```

Var_L_004

```
Program Var_L_004;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzP=24; {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzP] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    c        : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    AnzFit   : Integer; {Anzahl der Datenpaare für den Kraft-Weg-Fit}
    X,Y,Z    : Array [0..200] of Double; {XY-Datenpaare für Kraft-Weg-Fit und Z=Regression für Y}
    SW       : Double;  {Schrittweite für die Bewegung des Kraft-Weg-Fits}
    Schritte : Integer; {Anzahl der Schritte von der Gleichgewichtsposition bis zum Kraftmaximum}
    a        : Array [1..3] of Double;  {Parameter für die Regressionsfunktion}
    mussneurechnen : Boolean; {Können die Parameter übernommen werden ?}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzFit do  {von "plotanf" bis "plotend"}
  begin
```

```
    For j:=1 to 3 do
    begin   {Kolumnen drucken}
      If j=1  then Str(X[lv]:17:11,Zahl);
      If j=2  then Str(Y[lv]:17:11,Zahl);
      If j=3  then Str(Z[lv]:17:11,Zahl);
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Procedure Spulengeometrie_aufschreiben(Name:String);
Var fout : Text;    {Die Daten aus dem letzten Programm-Lauf}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  Writeln(fout,m);
  Writeln(fout,n);
  Writeln(fout,mk);
  Writeln(fout,nk);
  Writeln(fout,Dd);
  Writeln(fout,di);
  Writeln(fout,dk);
  Writeln(fout,Br);
  Writeln(fout,A[1]);
  Writeln(fout,A[2]);
  Writeln(fout,A[3]);
  Close(fout);
end;

Procedure Spulengeometrie_ggf_uebernehmen(Name:String);
Var fin : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lokm,lokn,lokmk,loknk   : LongInt; {lokale Parameter zur Überprüfung}
    lokDd,lokdi,lokdk,lokBr : Double;  {lokale Parameter zur Überprüfung}
begin
  mussneurechnen:=false;
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,lokm);   If Abs((lokm-m)/(lokm+m))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokn);   If Abs((lokn-n)/(lokn+n))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokmk);  If Abs((lokmk-mk)/(lokmk+mk))>1E-10 then mussneurechnen:=true;
  Readln(fin,loknk);  If Abs((loknk-nk)/(loknk+nk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokDd);  If Abs((lokDd-Dd)/(lokDd+Dd))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdi);  If Abs((lokdi-di)/(lokdi+di))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdk);  If Abs((lokdk-dk)/(lokdk+dk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokBr);  If Abs((lokBr-Br)/(lokBr+Br))>1E-10 then mussneurechnen:=true;
  Writeln('>> Muessen Parameter neu gerechnet werden ?  = ',mussneurechnen);
  If Not(mussneurechnen) then
  begin
    Readln(fin,A[1]);
    Readln(fin,A[2]);
    Readln(fin,A[3]);
  end;
  Close(fin);
end;

Function Fr(x:Double):Double;  {Kraft durch Regressions-Kurve berechnen}
Var merk,lnyw : Double;
    Vorz      : Real;     {Vorzeichen der Ortsangabe}
    rupos     : Double;   {Ruhelage-Position}
begin    {Kraftberechnung wurde nur für positive x-Werte bestimmt und punktsymmetrisch}
  rupos:=(ls-lk)/2;  merk:=0;
  Vorz:=-1;  If x<rupos then Vorz:=+1;
  If Abs(x-rupos)>=1E-10 then
  begin
    x:=Abs(x-rupos);
    lnyw:=ln(a[1])+a[2]*ln(x);    {nach *1 von S.45}
    merk:=exp(lnyw)*exp(-a[3]*x);
  end;
  Fr:=Vorz*merk;   {Spiegelung am Symmetriepunkt des Graphen}
end;

Function QS:Double;  {Quadratische Abweichungssumme}
Var lv   : Integer; {Laufvariable}
    Qsum : Double;  {Quadratsumme}
begin
  Qsum:=0;
  For lv:=0 to AnzFit do Qsum:=Qsum+Sqr(Y[lv]-Fr(X[lv]));
  QS:=Qsum;
end;

Procedure NichtlineneareRegression;
Var Sx,Sy,Sxx,Sxy : Double;  {Summen für die Regression der a1,a2}
    lv,lauf       : Integer; {Laufvariable}
```

```
    delA          : Array [1..3] of Double;  {Parameter-Variation für die Regressionsfunktion}
    QSmerk        : Double; {Zur Optimierung der Quadrat-Abweichungssumme}
    change        : Boolean; {wurden Parameter verändert ?}
Procedure Omptimiere_A(nr:Integer);
Var altA : Double;
    QSmit,QSminus,QSplus : Double; {Zur Optimierung der Quadrat-Abweichungssumme}
begin
  altA:=a[nr]; QSmit:=QS;
  a[nr]:=altA+delA[nr]; QSplus:=QS;
  a[nr]:=altA-delA[nr]; QSminus:=QS;
  If (QSplus<QSmit)  then a[nr]:=altA+delA[nr];
  If (QSminus<QSmit) then a[nr]:=altA-delA[nr];
  change:=(a[nr]<>altA);
end;
begin
  Writeln ('Nichtlineare Regression der Kraft-Weg-Kennlinie.');
  For lv:=0 to AnzFit do X[lv]:=X[lv]-(ls-lk)/2;  {Kraftnullpunkt in den Symmetriepunkt schieben}
{ Startwerte für die Parameter der Kurve suchen:}        change:=false;
  Sx:=0;  Sy:=0;  Sxx:=0;  Sxy:=0;  {Vorbereitung der Startwerte der Wurzelfunktion}
  For lv:=Round(AnzFit/2+1) to Round(AnzFit/2+Schritte) do
  begin
    Sx:=Sx+ln(Abs(X[lv]));
    Sy:=Sy+ln(Abs(Y[lv]));
    Sxx:=Sxx+Sqr(ln(Abs(Y[lv])));
    Sxy:=Sxy+ln(Abs(X[lv]))*ln(Abs(Y[lv]));
  end;
  a[2]:=Sxy-Sx*Sy/Schritte;                          {Geradensteigung, linear, nach *1 von S.45}
  a[1]:=Sy/Schritte-Sx*a[2]/Schritte; a[1]:=exp(a[1]);    {Achsenabschnitt, entlogarithmieren, nach *1 von S.45}
  a[3]:=1/X[AnzFit]*(ln(a[1])-a[2]*ln(X[AnzFit]));
  Writeln('Startwerte: a1 = ',a[1]:12:7,',  a2 = ',a[2]:12:7,',  a3 = ',a[3]:12:7);
{ Anpassen der Parameter an die Regressionskurve:}
  For lv:=0 to AnzFit do X[lv]:=X[lv]+(ls-lk)/2;  {Kraftnullpunkt in seine Ausgangslage zurückschieben}

{ Hier muß ich jetzt das Genau-Togglen der Parameter "a1,a2,a3" einbauen;}
{ Writeln('Kontrolle zu Beginn: QuadSum = ',QS); }
{ For lv:=0 to AnzFit do Writeln(lv:3,': x = ',X[lv]*100:8:5,'cm  => Fr = ',Fr(X[lv]),'N'); }
  For lv:=1 to 3 do delA[lv]:=a[lv]/100;  {Startwerte für die Optimierung der Parameter}
  Repeat
    Repeat
      QSmerk:=QS;
      For lv:=1 to 3 do
      begin
        lauf:=0;
        Repeat Omptimiere_A(lv); lauf:=lauf+1; Until (Change=false)or(lauf>5);
        Writeln('QS=',QS,' => a1=',a[1]:12:7,', a2=',a[2]:12:7,', a3=',a[3]:12:7);
      end;
    Until QS>0.99*QSmerk;
    For lv:=1 to 3 do delA[lv]:=delA[lv]/2;
  Until (delA[1]/A[1]<1E-8)and(delA[2]/A[2]<1E-8)and(delA[2]/A[2]<1E-8);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
```

```
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Procedure Kraft_Berechnung_vorbereiten;
Var i : Integer;
begin
  Writeln; Writeln('Test: Kraft zwischen Spule und Magnetkern, je nach x-Position des Kerns:');
  AnzFit:=20;  {Halbe (!) Anzahl der Punkte für den Kraft-Weg-Fit = (2*AnzFit+1) = 41.}
  Schritte:=5; {Anzahl der Daten-Schritte bis zum Kraft-Maximum}
  X[AnzFit]:=(ls-lk)/2;         {Ruhelageposition ist immer bei X[20]}
  SW:=(ls-X[AnzFit])/Schritte;  {Schrittweite für die Kraftberechnung}
  For i:=1 to AnzFit do
  begin
    X[AnzFit+i]:=X[AnzFit+i-1]+SW;
    X[AnzFit-i]:=2*X[AnzFit]-X[AnzFit+i];
    If i=Round(1.6*Schritte) then SW:=SW*2;
    If i=Round(2.8*Schritte) then SW:=SW*2;  {Schrittweite stufenweise erhöhen}
  end;
  AnzFit:=2*AnzFit;      {Jetzt wird AnzFit auf seinen endgültigen Werte gebracht.}
  For i:=0 to AnzFit do
  begin
    Y[i]:=Fges(X[i]);  {Für die echte Kraftberechnung muß ich das "Fges" hereinnehmen.}
```

```
    Z[i]:=0; {Der Regressions-Fit kommt später}
    Writeln(i:3,': ',X[i]*100:7:4,' cm -> ',Y[i],' N');
  end;
  Kraftwerte_ins_Excel;
{ Hier beginnt die Berechnung der nichtlinearen Regressionskurve: }
  NichtlineneareRegression;
  For i:=0 to AnzFit do Writeln(i:3,': ',X[i]*100:7:4,' cm -> ',Y[i],' N, ~',Fr(X[i]));  {Werte kontrollieren}
  For i:=0 to AnzFit do Z[i]:=Fr(X[i]);  {Der Regressions-Fit kommt jetzt}
  Kraftwerte_ins_Excel;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  c:=Sqrt(1/muo/epo){m/s};     {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',c, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=15;                       {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=3;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=0.2E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=10;                      {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=2;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di-0.002;                {Magnetkern-Durchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
{ AnzP -> Dieser Parameter wird oben als Konstante eingegeben: Anzahl der Zeitschritte insgesamt}
  dt:=0.0005; {sec.}           {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;          {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                 {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                   {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  dki:=dk-2*nk*Dd;             {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);         {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
{ Parameter für den Ablauf des Software-Algorithmus: }
{ AnzFit:=20;   {Anzahl der Punkt für den Kraft-Weg-Fit = (2*AnzFit+1) = 41. -> Initialisierung im Upgm.}
{ Schritte:=5;  {Anzahl der Kraft-Weg-Schritte bis zum Kraft-Maximum -> Initialisierung im Upgm.}

{ Anzeige der Werte, ggf. Übernahme vorhandener Parameter der Kraft-Weg-Regression:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Analyse in ',AnzP,' Zeit-Schritten der Dauer ',dt,' sec.');
  Spulengeometrie_ggf_uebernehmen('Spule_merk.dat');
  If mussneurechnen then Kraft_Berechnung_vorbereiten;
  Writeln('F(x)-Regr.-parameter: A1,2,3: ',A[1]:15:8,'  ',A[2]:15:8,'  ',A[3]:15:8);




{-------------------------------------------------------------------}
  Spulengeometrie_aufschreiben('Spule_merk.dat');
  Writeln('Fertig, Adele.');  Wait;  Wait;
End.
```

Var_L_005

```
Program Var_L_005;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzP=24; {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzP] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    c        : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    AnzFit   : Integer; {Anzahl der Datenpaare für den Kraft-Weg-Fit}
    X,Y,Z    : Array [0..200] of Double; {XY-Datenpaare für Kraft-Weg-Fit und Z=Regression für Y}
    SW       : Double;  {Schrittweite für die Bewegung des Kraft-Weg-Fits}
    Schritte : Integer; {Anzahl der Schritte von der Gleichgewichtsposition bis zum Kraftmaximum}
    a        : Array [1..3] of Double;  {Parameter für die Regressionsfunktion}
    mussneurechnen : Boolean; {Können die Parameter übernommen werden ?}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzFit do  {von "plotanf" bis "plotend"}
  begin
```

```
    For j:=1 to 3 do
    begin   {Kolumnen drucken}
      If j=1  then Str(X[lv]:17:11,Zahl);
      If j=2  then Str(Y[lv]:17:11,Zahl);
      If j=3  then Str(Z[lv]:17:11,Zahl);
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Procedure Spulengeometrie_aufschreiben(Name:String);
Var fout : Text;    {Die Daten aus dem letzten Programm-Lauf}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  Writeln(fout,m);
  Writeln(fout,n);
  Writeln(fout,mk);
  Writeln(fout,nk);
  Writeln(fout,Dd);
  Writeln(fout,di);
  Writeln(fout,dk);
  Writeln(fout,Br);
  Writeln(fout,A[1]);
  Writeln(fout,A[2]);
  Writeln(fout,A[3]);
  Close(fout);
end;

Procedure Spulengeometrie_ggf_uebernehmen(Name:String);
Var fin : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lokm,lokn,lokmk,loknk   : LongInt; {lokale Parameter zur Überprüfung}
    lokDd,lokdi,lokdk,lokBr : Double;  {lokale Parameter zur Überprüfung}
begin
  mussneurechnen:=false;
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,lokm);   If Abs((lokm-m)/(lokm+m))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokn);   If Abs((lokn-n)/(lokn+n))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokmk);  If Abs((lokmk-mk)/(lokmk+mk))>1E-10 then mussneurechnen:=true;
  Readln(fin,loknk);  If Abs((loknk-nk)/(loknk+nk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokDd);  If Abs((lokDd-Dd)/(lokDd+Dd))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdi);  If Abs((lokdi-di)/(lokdi+di))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdk);  If Abs((lokdk-dk)/(lokdk+dk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokBr);  If Abs((lokBr-Br)/(lokBr+Br))>1E-10 then mussneurechnen:=true;
  Writeln('>> Muessen Parameter neu gerechnet werden ?  = ',mussneurechnen);
  If Not(mussneurechnen) then
  begin
    Readln(fin,A[1]);
    Readln(fin,A[2]);
    Readln(fin,A[3]);
  end;
  Close(fin);
end;

Function Fr(x:Double):Double;  {Kraft durch Regressions-Kurve berechnen}
Var merk,lnyw : Double;
    Vorz      : Real;     {Vorzeichen der Ortsangabe}
    rupos     : Double;   {Ruhelage-Position}
begin    {Kraftberechnung wurde nur für positive x-Werte bestimmt und punktsymmetrisch}
  rupos:=(ls-lk)/2;  merk:=0;
  Vorz:=-1;  If x<rupos then Vorz:=+1;
  If Abs(x-rupos)>=1E-10 then
  begin
    x:=Abs(x-rupos);
    lnyw:=ln(a[1])+a[2]*ln(x);    {nach *1 von S.45}
    merk:=exp(lnyw)*exp(-a[3]*x);
  end;
  Fr:=Vorz*merk;   {Spiegelung am Symmetriepunkt des Graphen}
end;

Function QS:Double;  {Quadratische Abweichungssumme}
Var lv   : Integer; {Laufvariable}
    Qsum : Double;  {Quadratsumme}
begin
  Qsum:=0;
  For lv:=0 to AnzFit do Qsum:=Qsum+Sqr(Y[lv]-Fr(X[lv]));
  QS:=Qsum;
end;

Procedure NichtlineneareRegression;
Var Sx,Sy,Sxx,Sxy : Double;  {Summen für die Regression der a1,a2}
    lv,lauf       : Integer; {Laufvariable}
```

```
    delA          : Array [1..3] of Double;  {Parameter-Variation für die Regressionsfunktion}
    QSmerk        : Double; {Zur Optimierung der Quadrat-Abweichungssumme}
    change        : Boolean; {wurden Parameter verändert ?}
Procedure Omptimiere_A(nr:Integer);
Var altA : Double;
    QSmit,QSminus,QSplus : Double; {Zur Optimierung der Quadrat-Abweichungssumme}
begin
  altA:=a[nr]; QSmit:=QS;
  a[nr]:=altA+delA[nr]; QSplus:=QS;
  a[nr]:=altA-delA[nr]; QSminus:=QS;
  If (QSplus<QSmit)  then a[nr]:=altA+delA[nr];
  If (QSminus<QSmit) then a[nr]:=altA-delA[nr];
  change:=(a[nr]<>altA);
end;
begin
  Writeln ('Nichtlineare Regression der Kraft-Weg-Kennlinie.');
  For lv:=0 to AnzFit do X[lv]:=X[lv]-(ls-lk)/2;  {Kraftnullpunkt in den Symmetriepunkt schieben}
{ Startwerte für die Parameter der Kurve suchen:}        change:=false;
  Sx:=0;  Sy:=0;  Sxx:=0;  Sxy:=0;  {Vorbereitung der Startwerte der Wurzelfunktion}
  For lv:=Round(AnzFit/2+1) to Round(AnzFit/2+Schritte) do
  begin
    Sx:=Sx+ln(Abs(X[lv]));
    Sy:=Sy+ln(Abs(Y[lv]));
    Sxx:=Sxx+Sqr(ln(Abs(Y[lv])));
    Sxy:=Sxy+ln(Abs(X[lv]))*ln(Abs(Y[lv]));
  end;
  a[2]:=Sxy-Sx*Sy/Schritte;                          {Geradensteigung, linear, nach *1 von S.45}
  a[1]:=Sy/Schritte-Sx*a[2]/Schritte; a[1]:=exp(a[1]);    {Achsenabschnitt, entlogarithmieren, nach *1 von S.45}
  a[3]:=1/X[AnzFit]*(ln(a[1])-a[2]*ln(X[AnzFit]));
  Writeln('Startwerte: a1 = ',a[1]:12:7,',  a2 = ',a[2]:12:7,',  a3 = ',a[3]:12:7);
{ Anpassen der Parameter an die Regressionskurve:}
  For lv:=0 to AnzFit do X[lv]:=X[lv]+(ls-lk)/2;  {Kraftnullpunkt in seine Ausgangslage zurückschieben}
  Writeln('QS=',QS,' => a1=',a[1]:12:7,', a2=',a[2]:12:7,', a3=',a[3]:12:7);
{ Hier werde ich jetzt das Genau-Togglen der Parameter "a1,a2,a3" einbauen:}
  For lv:=1 to 3 do delA[lv]:=a[lv]/100;  {Startwerte für die Optimierung der Parameter}
{ Repeat
    Repeat
      QSmerk:=QS;
      For lv:=1 to 3 do
      begin
        lauf:=0;
        Repeat Omptimiere_A(lv); lauf:=lauf+1; Until (Change=false)or(lauf>5);
      end;
    Until QS>0.99*QSmerk;
    Writeln('QS=',QS,' => a1=',a[1]:12:7,', a2=',a[2]:12:7,', a3=',a[3]:12:7);
    For lv:=1 to 3 do delA[lv]:=delA[lv]/2;
  Until (delA[1]/A[1]<1E-8)and(delA[2]/A[2]<1E-8)and(delA[2]/A[2]<1E-8);  }
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
```

```
   Wert:=1+100*Anteil;
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Procedure Kraft_Berechnung_vorbereiten;
Var i : Integer;
begin
  Writeln; Writeln('Test: Kraft zwischen Spule und Magnetkern, je nach x-Position des Kerns:');
  AnzFit:=20;  {Halbe (!) Anzahl der Punkte für den Kraft-Weg-Fit = (2*AnzFit+1) = 41.}
  Schritte:=5; {Anzahl der Daten-Schritte bis zum Kraft-Maximum}
  X[AnzFit]:=(ls-lk)/2;         {Ruhelageposition ist immer bei X[20]}
  SW:=(ls-X[AnzFit])/Schritte;  {Schrittweite für die Kraftberechnung}
  For i:=1 to AnzFit do
  begin
    X[AnzFit+i]:=X[AnzFit+i-1]+SW;
    X[AnzFit-i]:=2*X[AnzFit]-X[AnzFit+i];
    If i=Round(1.6*Schritte) then SW:=SW*2;
    If i=Round(2.8*Schritte) then SW:=SW*2;  {Schrittweite stufenweise erhöhen}
  end;
  AnzFit:=2*AnzFit;      {Jetzt wird AnzFit auf seinen endgültigen Werte gebracht.}
  For i:=0 to AnzFit do
  begin
    Y[i]:=Fges(X[i]);  {Für die echte Kraftberechnung muß ich das "Fges" hereinnehmen.}
    Z[i]:=0; {Der Regressions-Fit kommt später}
    Writeln(i:3,': ',X[i]*100:7:4,' cm -> ',Y[i],' N');
```

```
  end;
  Kraftwerte_ins_Excel;
{ Hier beginnt die Berechnung der nichtlinearen Regressionskurve: }
  NichtlinenearеRegression;
  For i:=0 to AnzFit do Writeln(i:3,': ',X[i]*100:7:4,' cm -> ',Y[i],' N, ~',Fr(X[i]));  {Werte kontrollieren}
  For i:=0 to AnzFit do Z[i]:=Fr(X[i]);  {Der Regressions-Fit kommt jetzt}
  Kraftwerte_ins_Excel;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  c:=Sqrt(1/muo/epo){m/s};     {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',c, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=12;                       {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=1;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=1.5E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=8;                      {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=1;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di-0.002;                {Magnetkern-Durchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
{ AnzP -> Dieser Parameter wird oben als Konstante eingegeben: Anzahl der Zeitschritte insgesamt}
  dt:=0.0005; {sec.}           {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;          {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                 {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                   {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  dki:=dk-2*nk*Dd;             {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);         {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
{ Parameter für den Ablauf des Software-Algorithmus: }
{ AnzFit:=20;   {Anzahl der Punkt für den Kraft-Weg-Fit = (2*AnzFit+1) = 41. -> Initialisierung im Upgm.}
{ Schritte:=5;  {Anzahl der Kraft-Weg-Schritte bis zum Kraft-Maximum -> Initialisierung im Upgm.}

{ Anzeige der Werte, ggf. Übernahme vorhandener Parameter der Kraft-Weg-Regression:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Analyse in ',AnzP,' Zeit-Schritten der Dauer ',dt,' sec.');
  Spulengeometrie_ggf_uebernehmen('Spule_merk.dat');
  If mussneurechnen then Kraft_Berechnung_vorbereiten;
  Writeln('F(x)-Regr.-parameter: A1,2,3: ',A[1]:15:8,'  ',A[2]:15:8,'  ',A[3]:15:8);




{-----------------------------------------------------------------}
  Spulengeometrie_aufschreiben('Spule_merk.dat');
  Writeln('Fertig, Adele.');  Wait;  Wait;
End.
```

Var_L_006

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzP=24; {200000} {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzP] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    c        : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    X,Y,xs,ys: Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
```

```
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(X[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(Y[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
```

```
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xkern<=X[-lv] then merk:=Y[-lv];
  If xkern>=X[+lv] then merk:=Y[+lv];
  If (xkern>X[-lv])and(xkern<X[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until X[lv]>xpos;
    merk:=Y[lv-1]+(xpos-X[lv-1])/(X[lv]-X[lv-1])*(Y[lv]-Y[lv-1]);  {Lineare Interpolation}
  end;
  Fs:=merk;
end;


Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  c:=Sqrt(1/muo/epo){m/s};     {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',c, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=10;                       {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=1;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=1.0E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=10;                      {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=1;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di-0.002;                {Magnetkern-Durchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
{ AnzP -> Dieser Parameter wird oben als Konstante eingegeben: Anzahl der Zeitschritte insgesamt}
  dt:=0.0005; {sec.}           {Größe der Zeitschritte}
  Absted:=1;                   {Jeder wievielte Punkte soll geplottet werden}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;          {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                 {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                   {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  dki:=dk-2*nk*Dd;             {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);         {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');

{ Hier beginnt das Rechenprogramm:}      Writeln;
  Writeln; Writeln('Test: Kraft zwischen Spule und Magnetkern, je nach x-Position des Kerns:');
  X[0]:=(ls-lk)/2; Y[0]:=Fges(X[0]);        {Ruhelageposition ist immer bei X[0]}
```

```
  If Abs(Y[0])>1E-10 then
  begin
    Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',Y[0]:15:7,' N');
    Wait; Wait; Halt;
  end;
  Writeln('  0:',X[0]*100:15:7,' cm',Y[0]:15:7,' N');
  sw:=(ls-X[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
  AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
  For i:=1 to AnzSch do
  begin
    X[i]:=X[0]+i*SW;  Y[i]:=Fges(X[i]);
    X[-i]:=X[0]-i*SW; Y[-i]:=Fges(X[-i]);
  end;
  Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
    imax:=0; Repeat imax:=imax+1; Until -Y[imax+1]<-Y[imax];  {An dieser Stelle liegt das Kraftmaximum}
    If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
    begin
      i:=1;
      Repeat
        AnzAlt:=AnzSch;
        For j:=AnzAlt downto i do
        begin
          X[j+1]:=X[j];       Y[j+1]:=Y[j];
          X[-j-1]:=X[-j];     Y[-j-1]:=Y[-j];
        end;
        X[i]:=(X[i-1]+X[i+1])/2;       Y[i]:=Fges(X[i]);
        X[-i]:=(X[-i+1]+X[-i-1])/2;    Y[-i]:=Fges(X[-i]);
        AnzSch:=AnzSch+1;
        i:=i+2
      Until (i>2*imax+3);
    end;
  Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
  imax:=0; Repeat imax:=imax+1; Until -Y[imax+1]<-Y[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
  Repeat {Hier muß ich auch noch nach außen verlängern}
    AnzSch:=AnzSch+1;
    X[AnzSch]:=X[AnzSch-1]+sw;    Y[AnzSch]:=Fges(X[AnzSch]);
    X[-AnzSch]:=X[-AnzSch+1]-sw;  Y[-AnzSch]:=Fges(X[-AnzSch]);
  Until X[AnzSch]>5*X[imax];

{ Kontrolle der Kraft-Weg-Daten:}     Writeln;
  For i:=-AnzSch to +AnzSch do Writeln(i,':',X[i]*100:15:7,' cm',Y[i]:15:7,' N');
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Kraftberechnung:}
  xkern:=-0.017; Writeln('x = ',xkern*100:15:7,' cm => ',Fs(xkern):15:7,' N ');

  For i:=-500 to +500 do
  begin
    xs[i]:=i/1000*(X[+AnzSch]-X[-AnzSch]);
    ys[i]:=Fs(xs[i]);           Writeln(xs[i]*100:15:7,' cm => ',ys[i]:15:7,' N');
  end;
  X:=xs;  Y:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel;

{ ---------------------------------------------------------------- }
  Wait;   Wait;
End.
```

Var_L_007

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=2000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    cl       : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    L        : Double;  {Induktivität der Spule 1}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    AnzP     : LongInt; {Anzahl der tatsächlich berechneten Zeit-Schritte}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    XX,YY,xs,ys : Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}
    C        : Double;  {Kapazität des Kondensators}
    rho,R    : Double;  {Spezifischer und Ohm'scher Widerstand des Spulendrahtes}
    RD,RL    : Double;  {Drahtwiderstand des Spulendrahtes und Lastwiderstand, beide Ohm'sch}
    Q,Qp,Qpp    : Feld;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Feld;  {Auslenkung jeder einzelnen Kondensatorplatte}
    OMel,OMmech : Double;  {elektrische und mechanische Kreisfrequenz der schwingenden Systeme}
    UC,Ul       : Double;  {Kondensatorspannung}


Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
```

```
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(XX[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(YY[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
```

```
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xkern<=XX[-lv] then merk:=YY[-lv];
  If xkern>=XX[+lv] then merk:=YY[+lv];
  If (xkern>XX[-lv])and(xkern<XX[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until XX[lv]>xpos;
    merk:=YY[lv-1]+(xpos-XX[lv-1])/(XX[lv]-XX[lv-1])*(YY[lv]-YY[lv-1]);  {Lineare Interpolation}
  end;
  Fs:=merk;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  cl:=Sqrt(1/muo/epo){m/s};    {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',cl, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=4;                        {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=1;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=1.0E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=5;                       {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=1;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di*0.9;                  {Magnetkern-Durchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
  AnzP:=1000;                  {Anzahl der Zeitschritte insgesamt}
  dt:=0.001; {sec.}            {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Andere Komponenten (Kondensator und Widerstand):}
  C:=200E-6; {Farad}                              {Kapazität des Kondensators}
  rho:=1.7E-8; {Ohm*m}                            {Spez. Widerstand von Kupfer, je nach Temperatur,
Kohlrausch,T193}
  RD:=rho*(2*pi*di*n*m)/(pi*(Dd/2)*(Dd/2)){Ohm};  {Drahtwiderstand: Ohm`scher Widerstand des Spulendrahtes}
  RL:=1000*RD;                                       {Lastwiderstand, einstellbar je nach Verbraucher}
  R:=RL+RD;                                       {gesamter dämpfender Widerstand}
```

```
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;                {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                       {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                         {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  dki:=dk-2*nk*Dd;                   {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);               {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
  L:=n*n*m*m*(pi*Sqr(da/2))/ls;      {Induktivität der zylindrischen Spule, ohne Magnetkern drinnen, also ohne
mur}
  OMel:=1/Sqrt(L*C);                 {Eigen-Kreisfrequenz der harmonischen elektrischen Schwingkreises}
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Kapazitaet des Kondensators: C = ',C,' Farad');
  Writeln('Ohm`sche Widerstaende: R = ',R,' Ohm');
  Writeln('Induktivitaet der Spule ohne Magnetkern: L = ',L,' Henry');
  Writeln('Eigen-Kreisfrequenz harmon. el. Osz.: OMel = ',OMel,' Hz');

{ Hier beginnt das Rechenprogramm:}
  Writeln; Writeln('Vorbereitung: Kraft zwischen Spule und Magnetkern, als Fkt der x-Pos des Kerns.');
  XX[0]:=(ls-lk)/2; YY[0]:=Fges(XX[0]);        {Ruhelageposition ist immer bei X[0]}
  If Abs(YY[0])>1E-10 then
  begin
    Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',YY[0]:15:7,' N');
    Wait; Wait; Halt;
  end;
  sw:=(ls-XX[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
  AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
  For i:=1 to AnzSch do
  begin
    XX[i]:=XX[0]+i*SW;  YY[i]:=Fges(XX[i]);
    XX[-i]:=XX[0]-i*SW; YY[-i]:=Fges(XX[-i]);
  end;
  Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
    imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt das Kraftmaximum}
    If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
    begin
      i:=1;
      Repeat
        AnzAlt:=AnzSch;
        For j:=AnzAlt downto i do
        begin
          XX[j+1]:=XX[j];       YY[j+1]:=YY[j];
          XX[-j-1]:=XX[-j];     YY[-j-1]:=YY[-j];
        end;
        XX[i]:=(XX[i-1]+XX[i+1])/2;       YY[i]:=Fges(XX[i]);
        XX[-i]:=(XX[-i+1]+XX[-i-1])/2;    YY[-i]:=Fges(XX[-i]);
        AnzSch:=AnzSch+1;
        i:=i+2
      Until (i>2*imax+3);
    end;
  Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
  imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
  Repeat {Hier muß ich auch noch nach außen verlängern}
    AnzSch:=AnzSch+1;
    XX[AnzSch]:=XX[AnzSch-1]+sw;    YY[AnzSch]:=Fges(XX[AnzSch]);
    XX[-AnzSch]:=XX[-AnzSch+1]-sw;  YY[-AnzSch]:=Fges(XX[-AnzSch]);
  Until XX[AnzSch]>5*XX[imax];

{ Kontrolle der Kraft-Weg-Daten:}     Writeln;
{ For i:=-AnzSch to +AnzSch do Writeln(i,':',XX[i]*100:15:7,' cm',YY[i]:15:7,' N'); }
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Kraftberechnung:}
{ For i:=-500 to +500 do
  begin xs[i]:=i/1000*(XX[+AnzSch]-XX[-AnzSch]); ys[i]:=Fs(xs[i]); end;
  XX:=xs;  YY:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel; }

{ Nun beginnt die DFEM-Simulation der Systems:}
{ Teil_01: Harmonische Schwingung}
  Writeln('Teil_01: Harmonische Schwingung');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_02: Gedämpfte Schwingung}
  Writeln('Teil_02: Gedaempfte Schwingung (harmonisch, plus dazu Widerstand)');
```

```
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);


{ ---------------------------------------------------------------- }
  Wait;   Wait;
End.
```

Var_L_008

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=2000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    cl       : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    L        : Double;  {Induktivität der Spule 1}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    AnzP     : LongInt; {Anzahl der tatsächlich berechneten Zeit-Schritte}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    XX,YY,xs,ys : Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}
    C        : Double;  {Kapazität des Kondensators}
    rho,R    : Double;  {Spezifischer und Ohm'scher Widerstand des Spulendrahtes}
    RD,RL    : Double;  {Drahtwiderstand des Spulendrahtes und Lastwiderstand, beide Ohm'sch}
    Q,Qp,Qpp    : Feld;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Feld;  {Auslenkung jeder einzelnen Kondensatorplatte}
    OMel,OMmech : Double;  {elektrische und mechanische Kreisfrequenz der schwingenden Systeme}
    UC,Ul       : Double;  {Kondensatorspannung}


Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
```

```
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(XX[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(YY[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
```

```
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xkern<=XX[-lv] then merk:=YY[-lv];
  If xkern>=XX[+lv] then merk:=YY[+lv];
  If (xkern>XX[-lv])and(xkern<XX[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until XX[lv]>xpos;
    merk:=YY[lv-1]+(xpos-XX[lv-1])/(XX[lv]-XX[lv-1])*(YY[lv]-YY[lv-1]);  {Lineare Interpolation}
  end;
  Fs:=merk;
end;

Function Uind11(r1,r2,I2,x2i,x2im1,x1:Double):Double;
Var nen1,nen2,Klam : Double; {Induzierte Spannung Einzelwindung zu Einzelwindung, zwei Nenner, Klammer}
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil,Bx,nen : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   nen:=Sqr(r1)+Sqr(r1-r2);
   Bx:=muo*qp[i]*Sqr(r1)/2/Sqrt(nen*nen*nen); {Erzeugt die Spule 1 am Ort des Magnetkerns.}
   Anteil:=(muo*Bx)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
  nen1:=Sqr(r2)+Sqr(x1-x2i);
  nen2:=Sqr(r2)+Sqr(x1-x2im1);
  Klam:=1/Sqrt(nen1*nen1*nen1)-1/Sqrt(nen2*nen2*nen2);
  Uind11:=-muo*mur*pi*Sqr(r1)*Sqr(r2)*I2/2/dt*Klam; {Verwenden wir einen Magnetkern mit mur=1}
end;

Function Uindges:Double;  {Hier brauche ich kein Schnellrechen-Verfahren, weil keine Integration Zeit kostet.}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Usum : Double;        {Spannungs-Summe, Hintereinanderschaltung der Windungen}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Usum:=0;
  For ins:=1 to n do {Spule radial}
  begin
```

```
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=x[i]+(imk-0.5)*Dd;      {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
{         Writeln('xs: ',xs*100:10:7,' cm, ys: ',ys*100:10:7,' cm, xk: ',xk*100:10:7,' cm, yk: ',yk*100:10:7,'
cm'); }
          Usum:=Usum+Uind11(ys,yk,Ikern,xk,xk-x[i]+x[i-1],xs);
        end;
      end;
    end;
  end;
  Uindges:=Usum;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  cl:=Sqrt(1/muo/epo){m/s};    {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',cl, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=4;                        {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=1;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=1.0E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=5;                       {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=1;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di*0.9;                  {Magnetkern-Durchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
  AnzP:=1000;                  {Anzahl der Zeitschritte insgesamt}
  dt:=0.001; {sec.}            {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Andere Komponenten (Kondensator und Widerstand):}
  C:=200E-6; {Farad}                                {Kapazität des Kondensators}
  rho:=1.7E-8; {Ohm*m}                              {Spez. Widerstand von Kupfer, je nach Temperatur,
Kohlrausch,T193}
  RD:=rho*(2*pi*di*n*m)/(pi*(Dd/2)*(Dd/2)){Ohm};    {Drahtwiderstand: Ohm`scher Widerstand des Spulendrahtes}
  RL:=1000*RD;                                      {Lastwiderstand, einstellbar je nach Verbraucher}
  R:=RL+RD;                                         {gesamter dämpfender Widerstand}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;                {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                       {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                         {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  dki:=dk-2*nk*Dd;                   {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);               {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
  L:=n*n*m*m*(pi*Sqr(da/2))/ls;      {Induktivität der zylindrischen Spule, ohne Magnetkern drinnen, also ohne
mur}
  OMel:=1/Sqrt(L*C);                 {Eigen-Kreisfrequenz der harmonischen elektrischen Schwingkreises}
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Kapazitaet des Kondensators: C = ',C,' Farad');
  Writeln('Ohm`sche Widerstaende: R = ',R,' Ohm');
  Writeln('Induktivitaet der Spule ohne Magnetkern: L = ',L,' Henry');
  Writeln('Eigen-Kreisfrequenz harmon. el. Osz.: OMel = ',OMel,' Hz');

{ Hier beginnt das Rechenprogramm:}


{Hier kommt ein Test der Magnetkraft zwischen Spule und Magnetkern:}
  Writeln; Writeln('Vorbereitung: Kraft zwischen Spule und Magnetkern, als Fkt der x-Pos des Kerns.');
  XX[0]:=(ls-lk)/2; YY[0]:=Fges(XX[0]);        {Ruhelageposition ist immer bei X[0]}
  If Abs(YY[0])>1E-10 then
  begin
    Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',YY[0]:15:7,' N');
    Wait; Wait; Halt;
  end;
  sw:=(ls-XX[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
  AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
  For i:=1 to AnzSch do
  begin
    XX[i]:=XX[0]+i*SW;  YY[i]:=Fges(XX[i]);
    XX[-i]:=XX[0]-i*SW; YY[-i]:=Fges(XX[-i]);
  end;
  Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
```

```
    imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt das Kraftmaximum}
    If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
    begin
      i:=1;
      Repeat
        AnzAlt:=AnzSch;
        For j:=AnzAlt downto i do
        begin
          XX[j+1]:=XX[j];       YY[j+1]:=YY[j];
          XX[-j-1]:=XX[-j];     YY[-j-1]:=YY[-j];
        end;
        XX[i]:=(XX[i-1]+XX[i+1])/2;       YY[i]:=Fges(XX[i]);
        XX[-i]:=(XX[-i+1]+XX[-i-1])/2;    YY[-i]:=Fges(XX[-i]);
        AnzSch:=AnzSch+1;
        i:=i+2
      Until (i>2*imax+3);
    end;
  Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
  imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
  Repeat {Hier muß ich auch noch nach außen verlängern}
    AnzSch:=AnzSch+1;
    XX[AnzSch]:=XX[AnzSch-1]+sw;    YY[AnzSch]:=Fges(XX[AnzSch]);
    XX[-AnzSch]:=XX[-AnzSch+1]-sw;  YY[-AnzSch]:=Fges(XX[-AnzSch]);
  Until XX[AnzSch]>5*XX[imax];

{ Kontrolle der Kraft-Weg-Daten:}     Writeln;
{ For i:=-AnzSch to +AnzSch do Writeln(i,':',XX[i]*100:15:7,' cm',YY[i]:15:7,' N'); }
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Kraftberechnung:}
{ For i:=-500 to +500 do
  begin xs[i]:=i/1000*(XX[+AnzSch]-XX[-AnzSch]); ys[i]:=Fs(xs[i]); end;
  XX:=xs;  YY:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel; }
{ Unter Angabe der Kernposition kann jetzt die Magnetkraft "Fs" aufgerufen werden.}


{Hier kommt ein Test der induzierten Spannung, die die Bewegung des Magnetkerns in der Spule 1 induziert:}
  i:=17; x[i]:=-0.0040; x[i-1]:=-0.0041;   {Zwecks Kontrolle des "Uingges"-Unterprogramms}
  Writeln('Indizierte Test-Spannung: Uind = ',Uindges,' Volt.');   {Alles ok, alle Hilfsprogramme sind fertig.}
{ Unter Angabe der Kernposition und ihrer Änderung kann jetzt den induzierte Spannung "Uindges" aufgerufen
werden.}


{ Nun beginnt die DFEM-Simulation der Systems:}
{ Teil_01: Harmonische Schwingung}
  Writeln('Teil_01: Harmonische Schwingung');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_02: Gedämpfte Schwingung}
  Writeln('Teil_02: Gedaempfte Schwingung (harmonisch, plus dazu Widerstand)');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);


{ ------------------------------------------------------------- }
  Wait;   Wait;
End.
```

Var_L_009

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=5000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    cl       : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    L        : Double;  {Induktivität der Spule 1}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    AnzP     : LongInt; {Anzahl der tatsächlich berechneten Zeit-Schritte}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    XX,YY,xs,ys : Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}
    C        : Double;  {Kapazität des Kondensators}
    rho,R    : Double;  {Spezifischer und Ohm'scher Widerstand des Spulendrahtes}
    RD,RL    : Double;  {Drahtwiderstand des Spulendrahtes und Lastwiderstand, beide Ohm'sch}
    Q,Qp,Qpp    : Feld;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Feld;  {Auslenkung jeder einzelnen Kondensatorplatte}
    OMel,OMmech : Double;  {elektrische und mechanische Kreisfrequenz der schwingenden Systeme}
    UC,Ul       : Double;  {Kondensatorspannung}
    xo          : Double;  {Ruhelageposition der Feder (ohne Federkraft)}
    mo          : Double;  {Träge Masse des Magnetkerns}
    rhoKern     : Double;  {Träge Masse des Magnetkerns}
    D           : Double;  {Hooke'sche Federoknstante}
    rupomag     : Double;  {Ruhelageposition der Magnetkraft}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
```

```
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(XX[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(YY[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
```

```
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xpos<=XX[-lv] then merk:=YY[-lv];
  If xpos>=XX[+lv] then merk:=YY[+lv];
  If (xpos>XX[-lv])and(xpos<XX[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until XX[lv]>xpos;
    merk:=YY[lv-1]+(xpos-XX[lv-1])/(XX[lv]-XX[lv-1])*(YY[lv]-YY[lv-1]);  {Lineare Interpolation}
  end;
  Fs:=merk;
end;

Function Uind11(r1,r2,I2,x2i,x2im1,x1:Double):Double;
Var nen1,nen2,Klam : Double; {Induzierte Spannung Einzelwindung zu Einzelwindung, zwei Nenner, Klammer}
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil,Bx,nen : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   nen:=Sqr(r1)+Sqr(r1-r2);
   Bx:=muo*qp[i]*Sqr(r1)/2/Sqrt(nen*nen*nen); {Erzeugt die Spule 1 am Ort des Magnetkerns.}
   Anteil:=(muo*Bx)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
  nen1:=Sqr(r2)+Sqr(x1-x2i);
  nen2:=Sqr(r2)+Sqr(x1-x2im1);
  Klam:=1/Sqrt(nen1*nen1*nen1)-1/Sqrt(nen2*nen2*nen2);
  Uind11:=-muo*mur*pi*Sqr(r1)*Sqr(r2)*I2/2/dt*Klam; {Verwenden wir einen Magnetkern mit mur=1}
end;

Function Uindges:Double;  {Hier brauche ich kein Schnellrechen-Verfahren, weil keine Integration Zeit kostet.}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Usum : Double;        {Spannungs-Summe, Hintereinanderschaltung der Windungen}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
```

```
  Usum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=x[i]+(imk-0.5)*Dd;      {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
{         Writeln('xs: ',xs*100:10:7,' cm, ys: ',ys*100:10:7,' cm, xk: ',xk*100:10:7,' cm, yk: ',yk*100:10:7,'
cm'); }
          Usum:=Usum+Uind11(ys,yk,Ikern,xk,xk-x[i]+x[i-1],xs);
        end;
      end;
    end;
  end;
  Uindges:=Usum;
end;

Function Fmech:Double;       {Federkraft der mechanischen Feder}
begin                                  {x-Achse positiv nach links}
  Fmech:=-D*(x[i-1]-xo);               {Federkraft negativ nach links}
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  cl:=Sqrt(1/muo/epo){m/s};    {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',cl, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=4;                        {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=1;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=1.0E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=5;                       {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=1;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di*0.9;                  {Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
  AnzP:=5000;                  {Anzahl der Zeitschritte insgesamt}
  dt:=0.001; {sec.}            {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Andere Komponenten (Kondensator und Widerstand):}
  C:=200E-6; {Farad}                              {Kapazität des Kondensators}
  rho:=1.7E-8; {Ohm*m}                            {Spez. Widerstand von Kupfer, je nach Temperatur,
Kohlrausch,T193}
  RD:=rho*(2*pi*di*n*m)/(pi*(Dd/2)*(Dd/2)){Ohm};  {Drahtwiderstand: Ohm`scher Widerstand des Spulendrahtes}
  RL:=1000*RD;                                    {Lastwiderstand, einstellbar je nach Verbraucher}
  R:=RL+RD;                                       {gesamter dämpfender Widerstand}
{ Mechanische Feder und mechanische Schwingung:}
  xo:=(m*Dd-mk*Dd)/2; {willkürlich}  {Ruhelageposition der Feder (ohne Federkraft)}
  rhoKern:=7.8E3;                    {Dichte des Magnetkern-Materials, Eisen, Kohlrausch Bd.3}
  D:=80; {N/m}                       {Hooke'sche Federoknstante}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;                {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                       {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                         {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  rupomag:=(ls-lk)/2;                {Ruhelageposition der Magnetkraft}
  dki:=dk-2*nk*Dd;                   {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);               {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
  L:=n*n*m*m*(pi*Sqr(da/2))/ls;      {Induktivität der zylindrischen Spule, ohne Magnetkern drinnen, also ohne
mur}
  mo:=rhoKern*pi*Sqr(dk/2)*lk;       {Träge Masse des Magnetkerns}
  OMel:=1/Sqrt(L*C);                 {Eigen-Kreisfrequenz der harmonischen elektrischen Schwingkreises}
  OMmech:=Sqrt(D/mo);                {Eigen-Kreisfrequenz der mechanischen Schwingung}
  If (AnzP>AnzPmax) then
  begin
    Writeln ('AnzP = ',AnzP,' Zeitschritte koennen nicht verarbeitet werden. STOP.');
    Wait; Wait; Halt;
  end;
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Kapazitaet des Kondensators: C = ',C,' Farad');
  Writeln('Ohm`sche Widerstaende: R = ',R,' Ohm');
  Writeln('Induktivitaet der Spule ohne Magnetkern: L = ',L,' Henry');
  Writeln('Ruhelageposition der Magnetkraft: rupomag = ',rupomag*100:14:7,' cm');
```

```
  Writeln('Ruhelageposition der Feder (ohne Federkraft) xo = ',xo*100:14:7,' cm');
  Writeln('Traege Masse des Magnetkerns mo = ',mo,' kg');
  Writeln('Eigen-Kreisfreq harmon. el. Osz.: OMel = ',OMel:8:4,' Hz => T = ',2*pi/OMel:15,'sec.');
  Writeln('Eigen-Kreisfreq mechan. Schwingg: OMmech = ',OMmech:8:4,' Hz=> T= ',2*pi/OMmech:15,'sec.');
  Writeln('Analyseschritte: ',dt:14,' sec., Analysedauer: ',dt*AnzP:14,' sec.');
{ Hier beginnt das Rechenprogramm:}
{ Zuerst kommt ein Test der Magnetkraft zwischen Spule und Magnetkern:}
  Writeln; Writeln('Vorbereitung: Kraft zwischen Spule und Magnetkern, als Fkt der x-Pos des Kerns.');
  XX[0]:=(ls-lk)/2; YY[0]:=Fges(XX[0]);        {Ruhelageposition ist immer bei X[0]}
  If Abs(YY[0])>1E-10 then
  begin
    Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',YY[0]:15:7,' N');
    Wait; Wait; Halt;
  end;
  sw:=(ls-XX[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
  AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
  For i:=1 to AnzSch do
  begin
    XX[i]:=XX[0]+i*SW;  YY[i]:=Fges(XX[i]);
    XX[-i]:=XX[0]-i*SW; YY[-i]:=Fges(XX[-i]);
  end;
  Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
    imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt das Kraftmaximum}
    If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
    begin
      i:=1;
      Repeat
        AnzAlt:=AnzSch;
        For j:=AnzAlt downto i do
        begin
          XX[j+1]:=XX[j];       YY[j+1]:=YY[j];
          XX[-j-1]:=XX[-j];     YY[-j-1]:=YY[-j];
        end;
        XX[i]:=(XX[i-1]+XX[i+1])/2;       YY[i]:=Fges(XX[i]);
        XX[-i]:=(XX[-i+1]+XX[-i-1])/2;    YY[-i]:=Fges(XX[-i]);
        AnzSch:=AnzSch+1;
        i:=i+2
      Until (i>2*imax+3);
    end;
  Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
  imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
  Repeat {Hier muß ich auch noch nach außen verlängern}
    AnzSch:=AnzSch+1;
    XX[AnzSch]:=XX[AnzSch-1]+sw;    YY[AnzSch]:=Fges(XX[AnzSch]);
    XX[-AnzSch]:=XX[-AnzSch+1]-sw;  YY[-AnzSch]:=Fges(XX[-AnzSch]);
  Until XX[AnzSch]>5*XX[imax];

{ Kontrolle der Kraft-Weg-Daten:}     Writeln;
{ For i:=-AnzSch to +AnzSch do Writeln(i,':',XX[i]*100:15:7,' cm',YY[i]:15:7,' N'); }
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Magnet-Kraftberechnung:}
{ For i:=-500 to +500 do
  begin xs[i]:=i/1000*(XX[+AnzSch]-XX[-AnzSch]); ys[i]:=Fs(xs[i]); end;
  XX:=xs;  YY:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel; }
{ Unter Angabe der Kernposition kann jetzt die Magnetkraft "Fs" aufgerufen werden.}

{Hier kommt ein Test der induzierten Spannung, die die Bewegung des Magnetkerns in der Spule 1 induziert:}
  i:=17; x[i]:=-0.0040; x[i-1]:=-0.0041;   {Zwecks Kontrolle des "Uingges"-Unterprogramms}
  Writeln('Induzierte Test-Spannung: Uind = ',Uindges,' Volt.');   {Alles ok, alle Hilfsprogramme sind fertig.}
{ Unter Angabe der Kernposition und ihrer Änderung kann jetzt den induzierte Spannung "Uindges" aufgerufen
werden.}

{ Zu guter Letzt folgt die Federkraft der mechanischen Feder:}
  i:=17; x[i-1]:=0.02; {Zwecks Kontrolle des "Fmech"-Unterprogramms}
  Writeln('Federkraft der mechanischen Feder: Fmech = ',Fmech,' N');
{ Unter Angabe der Kernposition kann jetzt "Fmech" aufgerufen werden.}

{ ------------------------------------------------------------------------------------------- }
{ Nun beginnt die DFEM-Simulation der Systems:}
{ Teil_01: Harmonische Schwingung}
  Writeln('Teil_01: Harmonische elektrische Schwingung: L,C');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------- }
{ Teil_02: Gedämpfte Schwingung}
  Writeln('Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
```

```
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------ }
{ Teil_03: Schwingung mit mechanischer Bewegung des Spulenkerns, Magnetkraft und induzierter Spannung}
  Writeln('Teil_03: Reine mechanische Bewegung des Spulenkerns, Schwingung');
  Q[0]:=0;  Qp[0]:=0;  Qpp[0]:=0;  {Die Schwingung startet durch mechanische Anregung}
  x[0]:=rupomag+0.01; {Meter}   xp[0]:=0;  xpp[0]:=0;    {Startwerte der mechanischen Anregung}
  For i:=1 to AnzP do  {Lsg. der Dgln.}
  begin
    xpp[i]:=(Fmech{+Fs(x[i-1])})/m;
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;


{    Wait; Wait; }
  end;
  ExcelAusgabe('test_03.dat',3,x,xp,xpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

  { ------------------------------------------------------------------------------------------ }
  Wait;   Wait;
End.
```

Var_L_010

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=5000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    cl       : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    L        : Double;  {Induktivität der Spule 1}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    AnzP     : LongInt; {Anzahl der tatsächlich berechneten Zeit-Schritte}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    XX,YY,xs,ys : Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}
    C        : Double;  {Kapazität des Kondensators}
    rho,R    : Double;  {Spezifischer und Ohm'scher Widerstand des Spulendrahtes}
    RD,RL    : Double;  {Drahtwiderstand des Spulendrahtes und Lastwiderstand, beide Ohm'sch}
    Q,Qp,Qpp    : Feld;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Feld;  {Auslenkung jeder einzelnen Kondensatorplatte}
    OMel,OMmech : Double;  {elektrische und mechanische Kreisfrequenz der schwingenden Systeme}
    UC,Ul       : Double;  {Kondensatorspannung}
    xo          : Double;  {Ruhelageposition der Feder (ohne Federkraft)}
    mo          : Double;  {Träge Masse des Magnetkerns}
    rhoKern     : Double;  {Träge Masse des Magnetkerns}
    D           : Double;  {Hooke'sche Federoknstante}
    rupomag     : Double;  {Ruhelageposition der Magnetkraft}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:17:11,Zahl);
        If j=2  then Str(KB[lv]:17:11,Zahl);
        If j=3  then Str(KC[lv]:17:11,Zahl);
        If j=4  then Str(KD[lv]:17:11,Zahl);
        If j=5  then Str(KE[lv]:17:11,Zahl);
        If j=6  then Str(KF[lv]:17:11,Zahl);
        If j=7  then Str(KG[lv]:17:11,Zahl);
        If j=8  then Str(KH[lv]:17:11,Zahl);
        If j=9  then Str(KI[lv]:17:11,Zahl);
        If j=10 then Str(KJ[lv]:17:11,Zahl);
        If j=11 then Str(KK[lv]:17:11,Zahl);
        If j=12 then Str(KL[lv]:17:11,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
```

```
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(XX[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(YY[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
```

```
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xpos<=XX[-lv] then merk:=YY[-lv];
  If xpos>=XX[+lv] then merk:=YY[+lv];
  If (xpos>XX[-lv])and(xpos<XX[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until XX[lv]>xpos;
    merk:=YY[lv-1]+(xpos-XX[lv-1])/(XX[lv]-XX[lv-1])*(YY[lv]-YY[lv-1]);  {Lineare Interpolation}
  end;                  {Fges => XX,YY waren mit 1.0 Amp. Strom in Spule 1 gerechnet worden.}
  Fs:=merk*qp[i-1];     {Die endgültige Magnetkraft als Funktion des Stroms sei linear skaliert.}
end;

Function Uind11(r1,r2,I2,x2i,x2im1,x1:Double):Double;
Var nen1,nen2,Klam : Double; {Induzierte Spannung Einzelwindung zu Einzelwindung, zwei Nenner, Klammer}
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil,Bx,nen : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   nen:=Sqr(r1)+Sqr(r1-r2);
   Bx:=muo*qp[i]*Sqr(r1)/2/Sqrt(nen*nen*nen); {Erzeugt die Spule 1 am Ort des Magnetkerns.}
   Anteil:=(muo*Bx)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
  nen1:=Sqr(r2)+Sqr(x1-x2i);
  nen2:=Sqr(r2)+Sqr(x1-x2im1);
  Klam:=1/Sqrt(nen1*nen1*nen1)-1/Sqrt(nen2*nen2*nen2);
  Uind11:=-muo*mur*pi*Sqr(r1)*Sqr(r2)*I2/2/dt*Klam; {Verwenden wir einen Magnetkern mit mur=1}
end;

Function Uindges:Double;  {Hier brauche ich kein Schnellrechen-Verfahren, weil keine Integration Zeit kostet.}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Usum : Double;        {Spannungs-Summe, Hintereinanderschaltung der Windungen}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
```

```
  Usum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=x[i]+(imk-0.5)*Dd;      {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
{         Writeln('xs: ',xs*100:10:7,' cm, ys: ',ys*100:10:7,' cm, xk: ',xk*100:10:7,' cm, yk: ',yk*100:10:7,'
cm'); }
          Usum:=Usum+Uind11(ys,yk,Ikern,xk,xk-x[i]+x[i-1],xs);
        end;
      end;
    end;
  end;
  Uindges:=Usum;
end;

Function Fmech:Double;       {Federkraft der mechanischen Feder}
begin                                  {x-Achse positiv nach links}
  Fmech:=-D*(x[i-1]-xo);               {Federkraft negativ nach links}
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  cl:=Sqrt(1/muo/epo){m/s};    {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',cl, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=4;                        {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=1;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=1.0E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=5;                       {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=1;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di*0.9;                  {Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
  AnzP:=5000;                  {Anzahl der Zeitschritte insgesamt}
  dt:=0.001; {sec.}            {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Andere Komponenten (Kondensator und Widerstand):}
  C:=200E-6; {Farad}                              {Kapazität des Kondensators}
  rho:=1.7E-8; {Ohm*m}                            {Spez. Widerstand von Kupfer, je nach Temperatur,
Kohlrausch,T193}
  RD:=rho*(2*pi*di*n*m)/(pi*(Dd/2)*(Dd/2)){Ohm};  {Drahtwiderstand: Ohm`scher Widerstand des Spulendrahtes}
  RL:=1000*RD;                                    {Lastwiderstand, einstellbar je nach Verbraucher}
  R:=RL+RD;                                       {gesamter dämpfender Widerstand}
{ Mechanische Feder und mechanische Schwingung:}
  xo:=(m*Dd-mk*Dd)/2; {willkürlich}  {Ruhelageposition der Feder (ohne Federkraft)}
  rhoKern:=7.8E3;                    {Dichte des Magnetkern-Materials, Eisen, Kohlrausch Bd.3}
  D:=80; {N/m}                       {Hooke'sche Federoknstante}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;                {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                       {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                         {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  rupomag:=(ls-lk)/2;                {Ruhelageposition der Magnetkraft}
  dki:=dk-2*nk*Dd;                   {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);               {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
  L:=n*n*m*m*(pi*Sqr(da/2))/ls;      {Induktivität der zylindrischen Spule, ohne Magnetkern drinnen, also ohne
mur}
  mo:=rhoKern*pi*Sqr(dk/2)*lk;       {Träge Masse des Magnetkerns}
  OMel:=1/Sqrt(L*C);                 {Eigen-Kreisfrequenz der harmonischen elektrischen Schwingkreises}
  OMmech:=Sqrt(D/mo);                {Eigen-Kreisfrequenz der mechanischen Schwingung}
  If (AnzP>AnzPmax) then
  begin
    Writeln ('AnzP = ',AnzP,' Zeitschritte koennen nicht verarbeitet werden. STOP.');
    Wait; Wait; Halt;
  end;
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Kapazitaet des Kondensators: C = ',C,' Farad');
  Writeln('Ohm`sche Widerstaende: R = ',R,' Ohm');
  Writeln('Induktivitaet der Spule ohne Magnetkern: L = ',L,' Henry');
  Writeln('Ruhelageposition der Magnetkraft: rupomag = ',rupomag*100:14:7,' cm');
```

```
  Writeln('Ruhelageposition der Feder (ohne Federkraft) xo = ',xo*100:14:7,' cm');
  Writeln('Traege Masse des Magnetkerns mo = ',mo,' kg');
  Writeln('Eigen-Kreisfreq harmon. el. Osz.: OMel = ',OMel:8:4,' Hz => T = ',2*pi/OMel:15,'sec.');
  Writeln('Eigen-Kreisfreq mechan. Schwingg: OMmech = ',OMmech:8:4,' Hz=> T= ',2*pi/OMmech:15,'sec.');
  Writeln('Analyseschritte: ',dt:14,' sec., Analysedauer: ',dt*AnzP:14,' sec.');
{ Hier beginnt das Rechenprogramm:}
{ Zuerst kommt ein Test der Magnetkraft zwischen Spule und Magnetkern:}
  Writeln; Writeln('Vorbereitung: Kraft zwischen Spule und Magnetkern, als Fkt der x-Pos des Kerns.');
  XX[0]:=(ls-lk)/2; YY[0]:=Fges(XX[0]);        {Ruhelageposition ist immer bei X[0]}
  If Abs(YY[0])>1E-10 then
  begin
    Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',YY[0]:15:7,' N');
    Wait; Wait; Halt;
  end;
  sw:=(ls-XX[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
  AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
  For i:=1 to AnzSch do
  begin
    XX[i]:=XX[0]+i*SW;  YY[i]:=Fges(XX[i]);
    XX[-i]:=XX[0]-i*SW; YY[-i]:=Fges(XX[-i]);
  end;
  Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
    imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt das Kraftmaximum}
    If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
    begin
      i:=1;
      Repeat
        AnzAlt:=AnzSch;
        For j:=AnzAlt downto i do
        begin
          XX[j+1]:=XX[j];       YY[j+1]:=YY[j];
          XX[-j-1]:=XX[-j];     YY[-j-1]:=YY[-j];
        end;
        XX[i]:=(XX[i-1]+XX[i+1])/2;       YY[i]:=Fges(XX[i]);
        XX[-i]:=(XX[-i+1]+XX[-i-1])/2;    YY[-i]:=Fges(XX[-i]);
        AnzSch:=AnzSch+1;
        i:=i+2
      Until (i>2*imax+3);
    end;
  Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
  imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
  Repeat {Hier muß ich auch noch nach außen verlängern}
    AnzSch:=AnzSch+1;
    XX[AnzSch]:=XX[AnzSch-1]+sw;    YY[AnzSch]:=Fges(XX[AnzSch]);
    XX[-AnzSch]:=XX[-AnzSch+1]-sw;  YY[-AnzSch]:=Fges(XX[-AnzSch]);
  Until XX[AnzSch]>5*XX[imax];

{ Kontrolle der Kraft-Weg-Daten:}     Writeln;
{ For i:=-AnzSch to +AnzSch do Writeln(i,':',XX[i]*100:15:7,' cm',YY[i]:15:7,' N'); }
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Magnet-Kraftberechnung:}
{ For i:=-500 to +500 do
  begin xs[i]:=i/1000*(XX[+AnzSch]-XX[-AnzSch]); ys[i]:=Fs(xs[i]); end;
  XX:=xs;  YY:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel; }
{ Unter Angabe der Kernposition kann jetzt die Magnetkraft "Fs" aufgerufen werden.}

{Hier kommt ein Test der induzierten Spannung, die die Bewegung des Magnetkerns in der Spule 1 induziert:}
  i:=17; x[i]:=-0.0040; x[i-1]:=-0.0041;   {Zwecks Kontrolle des "Uingges"-Unterprogramms}
  Writeln('Induzierte Test-Spannung: Uind = ',Uindges,' Volt.');   {Alles ok, alle Hilfsprogramme sind fertig.}
{ Unter Angabe der Kernposition und ihrer Änderung kann jetzt den induzierte Spannung "Uindges" aufgerufen
werden.}

{ Zu guter Letzt folgt die Federkraft der mechanischen Feder:}
  i:=17; x[i-1]:=0.02; {Zwecks Kontrolle des "Fmech"-Unterprogramms}
  Writeln('Federkraft der mechanischen Feder: Fmech = ',Fmech,' N');
{ Unter Angabe der Kernposition kann jetzt "Fmech" aufgerufen werden.}

{ ------------------------------------------------------------------------------------------- }
{ Nun beginnt die DFEM-Simulation der Systems:}
{ Teil_01: Harmonische Schwingung}
  Writeln('Teil_01: Harmonische elektrische Schwingung: L,C');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------- }
{ Teil_02: Gedämpfte Schwingung}
  Writeln('Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
```

```
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------ }
{ Teil_03: Schwingung mit mechanischer Bewegung des Spulenkerns, Magnetkraft und induzierter Spannung}
  Writeln('Teil_03: Reine mechanische Bewegung des Spulenkerns, Schwingung');
  Q[0]:=0;  Qp[0]:=0;  Qpp[0]:=0;  {Die Schwingung startet durch mechanische Anregung}
  x[0]:=rupomag+0.01; {Meter}   xp[0]:=0;  xpp[0]:=0;    {Startwerte der mechanischen Anregung}
  For i:=1 to AnzP do  {Lsg. der Dgln.}
  begin
    xpp[i]:=(Fmech+Fs(x[i-1]))/m;
    xp[i]:=xp[i-1]+xpp[i]*dt;
    x[i]:=x[i-1]+xp[i]*dt;
    Q[i]:=1.0;  Qp[i]:=1.0;  Qpp[i]:=1.0;
{   Writeln(i,': Fs = ',Fs(x[i-1]),' N, Fmech = ',Fmech,' N'); }


{   Wait; Wait; }
  end;
  ExcelAusgabe('test_03.dat',6,x,xp,xpp,Q,Qp,Qpp,x,x,x,x,x,x);

  { ------------------------------------------------------------------------------------------ }
  Wait;   Wait;
End.
```

Var_L_011

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=5000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    cl       : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    L        : Double;  {Induktivität der Spule 1}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    WikFak   : Double;  {Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    AnzP     : LongInt; {Anzahl der tatsächlich berechneten Zeit-Schritte}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    XX,YY,xs,ys : Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}
    C        : Double;  {Kapazität des Kondensators}
    rho,R    : Double;  {Spezifischer und Ohm'scher Widerstand des Spulendrahtes}
    RD,RL    : Double;  {Drahtwiderstand des Spulendrahtes und Lastwiderstand, beide Ohm'sch}
    Q,Qp,Qpp    : Feld;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Feld;  {Auslenkung jeder einzelnen Kondensatorplatte}
    K1,K2,K3    : Feld;  {Drei Kontroll-Felder für Plot-Zwecke}
    OMel,OMmech : Double;  {elektrische und mechanische Kreisfrequenz der schwingenden Systeme}
    UC,Ul       : Double;  {Kondensatorspannung}
    xo          : Double;  {Ruhelageposition der Feder (ohne Federkraft)}
    mo          : Double;  {Träge Masse des Magnetkerns}
    rhoKern     : Double;  {Träge Masse des Magnetkerns}
    D           : Double;  {Hooke'sche Federoknstante}
    rupomag     : Double;  {Ruhelageposition der Magnetkraft}
    mussneurechnen : Boolean; {Können die Parameter der Spulengeometrie übernommen werden ?}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
```

```
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(XX[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(YY[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
```

```
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xpos<=XX[-lv] then merk:=YY[-lv];
  If xpos>=XX[+lv] then merk:=YY[+lv];
  If (xpos>XX[-lv])and(xpos<XX[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until XX[lv]>xpos;
    merk:=YY[lv-1]+(xpos-XX[lv-1])/(XX[lv]-XX[lv-1])*(YY[lv]-YY[lv-1]);  {Lineare Interpolation}
  end;                {Fges => XX,YY waren mit 1.0 Amp. Strom in Spule 1 gerechnet worden.}
  Fs:=merk*qp[i-1];    {Die endgültige Magnetkraft als Funktion des Stroms sei linear skaliert.}
end;

Function Uind11(r1,r2,I2,x2i,x2im1,x1:Double):Double;
Var nen1,nen2,Klam : Double; {Induzierte Spannung Einzelwindung zu Einzelwindung, zwei Nenner, Klammer}
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil,Bx,nen : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   nen:=Sqr(r1)+Sqr(r1-r2);
   Bx:=muo*qp[i]*Sqr(r1)/2/Sqrt(nen*nen*nen); {Erzeugt die Spule 1 am Ort des Magnetkerns.}
   Anteil:=(muo*Bx)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
  nen1:=Sqr(r2)+Sqr(x1-x2i);
  nen2:=Sqr(r2)+Sqr(x1-x2im1);
  Klam:=1/Sqrt(nen1*nen1*nen1)-1/Sqrt(nen2*nen2*nen2);
  Uind11:=-muo*mur*pi*Sqr(r1)*Sqr(r2)*I2/2/dt*Klam; {Verwenden wir einen Magnetkern mit mur=1}
end;

Function Uindges:Double;  {Hier brauche ich kein Schnellrechen-Verfahren, weil keine Integration Zeit kostet.}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
```

```
    Usum : Double;        {Spannungs-Summe, Hintereinanderschaltung der Windungen}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Usum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=x[i]+(imk-0.5)*Dd;      {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
{         Writeln('xs: ',xs*100:10:7,' cm, ys: ',ys*100:10:7,' cm, xk: ',xk*100:10:7,' cm, yk: ',yk*100:10:7,'
cm'); }
          Usum:=Usum+Uind11(ys,yk,Ikern,xk,xk-x[i]+x[i-1],xs);
        end;
      end;
    end;
  end;
  Uindges:=Usum;
end;

Function Fmech:Double;       {Federkraft der mechanischen Feder}
begin                                  {x-Achse positiv nach links}
  Fmech:=-D*(x[i-1]-xo);               {Federkraft negativ nach links}
end;

Procedure Spulengeometrie_ggf_uebernehmen(Name:String);
Var fin : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lokm,lokn,lokmk,loknk   : LongInt; {lokale Parameter zur Überprüfung}
    lokDd,lokdi,lokdk,lokBr : Double;  {lokale Parameter zur Überprüfung}
    loWikFak                : LongInt;
    lv : Integer;
begin
  mussneurechnen:=false;
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,lokm);     If Abs((lokm-m)/(lokm+m))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokn);     If Abs((lokn-n)/(lokn+n))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokmk);    If Abs((lokmk-mk)/(lokmk+mk))>1E-10 then mussneurechnen:=true;
  Readln(fin,loknk);    If Abs((loknk-nk)/(loknk+nk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokDd);    If Abs((lokDd-Dd)/(lokDd+Dd))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdi);    If Abs((lokdi-di)/(lokdi+di))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdk);    If Abs((lokdk-dk)/(lokdk+dk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokBr);    If Abs((lokBr-Br)/(lokBr+Br))>1E-10 then mussneurechnen:=true;
  Readln(fin,loWikFak); If Abs((loWikFak-WikFak)/(loWikFak+WikFak))>1E-10 then mussneurechnen:=true;
  Writeln('>> Muessen Parameter neu gerechnet werden ?  = ',mussneurechnen);
  If Not(mussneurechnen) then
  begin
    Readln(fin,AnzSch);
    For lv:=-AnzSch to AnzSch do
    begin
      Readln(fin,XX[lv]); Readln(fin,YY[lv]);
    end;
  end;
  Close(fin);
end;

Procedure Spulengeometrie_aufschreiben(Name:String);
Var fout : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lv : Integer;
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  Writeln(fout,m);
  Writeln(fout,n);
  Writeln(fout,mk);
  Writeln(fout,nk);
  Writeln(fout,Dd);
  Writeln(fout,di);
  Writeln(fout,dk);
  Writeln(fout,Br);
  Writeln(fout,WikFak);
  Writeln(fout,AnzSch);
  For lv:=-AnzSch to AnzSch do
  begin
    Writeln(fout,XX[lv]); Writeln(fout,YY[lv]);
  end;
  Close(fout);
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
```

```
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  cl:=Sqrt(1/muo/epo){m/s};    {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',cl, ' m/s');
{ Spule 1, außen, Bleistift in *1 von S.36:}
  m:=4;                        {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
  n:=1;                        {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
  Dd:=1.0E-3; {Meter}          {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
  di:=0.02;   {Meter}          {Spuleninnendurchmesser, Spule 1}
  WikFak:=1;                   {Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz}
{ Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
  mk:=5;                       {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
  nk:=1;                       {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
  dk:=di*0.9;                  {Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
  Br:=1.2; {Tesla}             {remanentes longitudinales Feld des Magnetkerns}
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
  AnzP:=5000;                  {Anzahl der Zeitschritte insgesamt}
  dt:=0.001; {sec.}            {Größe der Zeitschritte}
  Abstd:=1;                    {Jeder wievielte Punkte soll geplottet werden}
{ Andere Komponenten (Kondensator und Widerstand):}
  C:=200E-6; {Farad}                              {Kapazität des Kondensators}
  rho:=1.7E-8; {Ohm*m}                            {Spez. Widerstand von Kupfer, je nach Temperatur,
Kohlrausch,T193}
  RD:=rho*(2*pi*di*n*m)/(pi*(Dd/2)*(Dd/2)){Ohm};  {Drahtwiderstand: Ohm`scher Widerstand des Spulendrahtes}
  RL:=0;                                          {Lastwiderstand, einstellbar je nach Verbraucher}
  R:=RL+RD;                                       {gesamter dämpfender Widerstand}
{ Mechanische Feder und mechanische Schwingung:}
  xo:=(m*Dd-mk*Dd)/2; {willkürlich}  {Ruhelageposition der Feder (ohne Federkraft)}
  rhoKern:=7.8E3;                    {Dichte des Magnetkern-Materials, Eisen, Kohlrausch Bd.3}
  D:=80; {N/m}                       {Hooke'sche Federoknstante}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;                {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                       {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                         {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  rupomag:=(ls-lk)/2;                {Ruhelageposition der Magnetkraft}
  dki:=dk-2*nk*Dd;                   {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);               {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
  L:=n*n*m*m*(pi*Sqr(da/2))/ls*Sqr(WikFak); {Induktivität der zylindrischen Spule, ohne Magnetkern drinnen, also
ohne mur}
  mo:=rhoKern*pi*Sqr(dk/2)*lk;       {Träge Masse des Magnetkerns}
  OMel:=1/Sqrt(L*C);                 {Eigen-Kreisfrequenz der harmonischen elektrischen Schwingkreises}
  OMmech:=Sqrt(D/mo);                {Eigen-Kreisfrequenz der mechanischen Schwingung}
  If (AnzP>AnzPmax) then
  begin
    Writeln ('AnzP = ',AnzP,' Zeitschritte koennen nicht verarbeitet werden. STOP.');
    Wait; Wait; Halt;
  end;
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Kapazitaet des Kondensators: C = ',C,' Farad');
  Writeln('Ohm`sche Widerstaende: R = ',R,' Ohm');
  Writeln('Induktivitaet der Spule ohne Magnetkern: L = ',L,' Henry');
  Writeln('Ruhelageposition der Magnetkraft: rupomag = ',rupomag*100:14:7,' cm');
  Writeln('Ruhelageposition der Feder (ohne Federkraft) xo = ',xo*100:14:7,' cm');
  Writeln('Traege Masse des Magnetkerns mo = ',mo,' kg');
  Writeln('Eigen-Kreisfreq harmon. el. Osz.: OMel = ',OMel:8:4,' Hz => T = ',2*pi/OMel:15,'sec.');
  Writeln('Eigen-Kreisfreq mechan. Schwingg: OMmech = ',OMmech:8:4,' Hz=> T= ',2*pi/OMmech:15,'sec.');
  Writeln('Analyseschritte: ',dt:14,' sec., Analysedauer: ',dt*AnzP:14,' sec.');
{ Hier beginnt das Rechenprogramm:}
{ Zuerst kommt ein Test der Magnetkraft zwischen Spule und Magnetkern:}
  Writeln; Writeln('Vorbereitung: Kraft zwischen Spule und Magnetkern, als Fkt der x-Pos des Kerns.');
  Spulengeometrie_ggf_uebernehmen('Spule_merk.dat');
  If mussneurechnen then
  begin
    XX[0]:=(ls-lk)/2; YY[0]:=Fges(XX[0]);        {Ruhelageposition ist immer bei X[0]}
    If Abs(YY[0])>1E-10 then
    begin
      Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',YY[0]:15:7,' N');
      Wait; Wait; Halt;
    end;
    sw:=(ls-XX[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
    AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
    For i:=1 to AnzSch do
    begin
      XX[i]:=XX[0]+i*SW;  YY[i]:=Fges(XX[i]);
      XX[-i]:=XX[0]-i*SW; YY[-i]:=Fges(XX[-i]);
    end;
    Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
      imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt das Kraftmaximum}
      If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
      begin
        i:=1;
        Repeat
          AnzAlt:=AnzSch;
          For j:=AnzAlt downto i do
```

```
          begin
            XX[j+1]:=XX[j];       YY[j+1]:=YY[j];
            XX[-j-1]:=XX[-j];     YY[-j-1]:=YY[-j];
          end;
          XX[i]:=(XX[i-1]+XX[i+1])/2;       YY[i]:=Fges(XX[i]);
          XX[-i]:=(XX[-i+1]+XX[-i-1])/2;    YY[-i]:=Fges(XX[-i]);
          AnzSch:=AnzSch+1;
          i:=i+2
        Until (i>2*imax+3);
      end;
    Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
    imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
    Repeat {Hier muß ich auch noch nach außen verlängern}
      AnzSch:=AnzSch+1;
      XX[AnzSch]:=XX[AnzSch-1]+sw;    YY[AnzSch]:=Fges(XX[AnzSch]);
      XX[-AnzSch]:=XX[-AnzSch+1]-sw;  YY[-AnzSch]:=Fges(XX[-AnzSch]);
    Until XX[AnzSch]>5*XX[imax];
  end;
{ Kontrolle der Kraft-Weg-Daten:}
{ For i:=-AnzSch to +AnzSch do Writeln(i,':',XX[i]*100:15:7,' cm',YY[i]:15:7,' N'); }
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Magnet-Kraftberechnung:}
{ For i:=-500 to +500 do
  begin xs[i]:=i/1000*(XX[+AnzSch]-XX[-AnzSch]); ys[i]:=Fs(xs[i]); end;
  XX:=xs;  YY:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel; }
{ Unter Angabe der Kernposition kann jetzt die Magnetkraft "Fs" aufgerufen werden.}

{Hier kommt ein Test der induzierten Spannung, die die Bewegung des Magnetkerns in der Spule 1 induziert:}
  i:=17; x[i]:=-0.0040; x[i-1]:=-0.0041;   {Zwecks Kontrolle des "Uingges"-Unterprogramms}
  Writeln('Induzierte Test-Spannung: Uind = ',Uindges,' Volt.');   {Alles ok, alle Hilfsprogramme sind fertig.}
{ Unter Angabe der Kernposition und ihrer Änderung kann jetzt den induzierte Spannung "Uindges" aufgerufen
werden.}

{ Zu guter Letzt folgt die Federkraft der mechanischen Feder:}
  i:=17; x[i-1]:=0.02; {Zwecks Kontrolle des "Fmech"-Unterprogramms}
  Writeln('Federkraft der mechanischen Feder: Fmech = ',Fmech,' N');
{ Unter Angabe der Kernposition kann jetzt "Fmech" aufgerufen werden.}

{ ------------------------------------------------------------------------------------------------ }
{ Nun beginnt die DFEM-Simulation der Systems:}
{ Teil_01: Harmonische Schwingung}
  Writeln('Teil_01: Harmonische elektrische Schwingung: L,C');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------------ }
{ Teil_02: Gedämpfte Schwingung}
  Writeln('Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------------ }
{ Teil_03: Schwingung mit mechanischer Bewegung des Spulenkerns, Magnetkraft und induzierter Spannung}
  Writeln('Teil_03: Reine mechanische Bewegung des Spulenkerns, Schwingung');
  UC:=0;{Volt}   Q[0]:=C*UC;{Coulomb}    {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                  {Startwerte der Integrationskonstanten}
  x[0]:=rupomag+0.10;{Meter}   xp[0]:=0;  xpp[0]:=0;  {Die Schwingung startet durch elektr. Anregung}
  For i:=1 to AnzP do  {Lsg. der Dgln.}
  begin
    xpp[i]:=(Fmech+WikFak*Fs(x[i-1]))/m; {Magnetkraft und Federkraft wirken,  WikFak beachten}
    xp[i]:=xp[i-1]+xpp[i]*dt;            K2[i]:=WikFak*Fs(x[i-1]);
    x[i]:=x[i-1]+xp[i]*dt;
{   Writeln(i,': Fs = ',WikFak*Fs(x[i-1]),' N, Fmech = ',Fmech,' N'); }
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    K1[i]:=-(Uindges*WikFak)/L; Qpp[i]:=Qpp[i]+K1[i];  {Induzierte Spannung nach *1 von S.51 in Spule 1 bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Wait; Wait; }
```

```
  end;
  ExcelAusgabe('test_03.dat',8,x,xp,xpp,Q,Qp,Qpp,K1,K2,x,x,x,x);

{ ------------------------------------------------------------------------------------------------ }
  Wait;   Wait;       mussneurechnen:=true;
  If mussneurechnen then Spulengeometrie_aufschreiben('Spule_merk.dat');
End.
```

Var_L_012

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    cl       : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    L        : Double;  {Induktivität der Spule 1}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    WikFak   : LongInt; {Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    AnzP     : LongInt; {Anzahl der tatsächlich berechneten Zeit-Schritte}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    XX,YY,xs,ys : Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}
    C        : Double;  {Kapazität des Kondensators}
    rho,R    : Double;  {Spezifischer und Ohm'scher Widerstand des Spulendrahtes}
    RD,RL    : Double;  {Drahtwiderstand des Spulendrahtes und Lastwiderstand, beide Ohm'sch}
    Q,Qp,Qpp    : Feld;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Feld;  {Auslenkung jeder einzelnen Kondensatorplatte}
    K1,K2,K3    : Feld;  {Drei Kontroll-Felder für Plot-Zwecke}
    OMel,OMmech : Double;  {elektrische und mechanische Kreisfrequenz der schwingenden Systeme}
    UC,Ul       : Double;  {Kondensatorspannung}
    xo          : Double;  {Ruhelageposition der Feder (ohne Federkraft)}
    mo          : Double;  {Träge Masse des Magnetkerns}
    rhoKern     : Double;  {Träge Masse des Magnetkerns}
    D           : Double;  {Hooke'sche Federoknstante}
    rupomag     : Double;  {Ruhelageposition der Magnetkraft}
    mussneurechnen : Boolean; {Können die Parameter der Spulengeometrie übernommen werden ?}
    Lesen          : String;  {Nummer der zu lesenden Spulendaten}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
```

```
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(XX[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(YY[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;

Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
```

```
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;  If n*m>50 then write('Dauer*',n*m,',  ');
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xpos<=XX[-lv] then merk:=YY[-lv];
  If xpos>=XX[+lv] then merk:=YY[+lv];
  If (xpos>XX[-lv])and(xpos<XX[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until XX[lv]>xpos;
    merk:=YY[lv-1]+(xpos-XX[lv-1])/(XX[lv]-XX[lv-1])*(YY[lv]-YY[lv-1]);  {Lineare Interpolation}
  end;                  {Fges => XX,YY waren mit 1.0 Amp. Strom in Spule 1 gerechnet worden.}
  Fs:=merk*qp[i-1];     {Die endgültige Magnetkraft als Funktion des Stroms sei linear skaliert.}
end;

Function Uind11(r1,r2,I2,x2i,x2im1,x1:Double):Double;
Var nen1,nen2,Klam : Double; {Induzierte Spannung Einzelwindung zu Einzelwindung, zwei Nenner, Klammer}
 Function mur:Double;       {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil,Bx,nen : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   nen:=Sqr(r1)+Sqr(r1-r2);
   Bx:=muo*qp[i]*Sqr(r1)/2/Sqrt(nen*nen*nen); {Erzeugt die Spule 1 am Ort des Magnetkerns.}
   Anteil:=(muo*Bx)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
  nen1:=Sqr(r2)+Sqr(x1-x2i);
  nen2:=Sqr(r2)+Sqr(x1-x2im1);
  Klam:=1/Sqrt(nen1*nen1*nen1)-1/Sqrt(nen2*nen2*nen2);
  Uind11:=-muo*mur*pi*Sqr(r1)*Sqr(r2)*I2/2/dt*Klam; {Verwenden wir einen Magnetkern mit mur=1}
end;

Function Uindges:Double;  {Hier brauche ich kein Schnellrechen-Verfahren, weil keine Integration Zeit kostet.}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
```

```
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Usum : Double;        {Spannungs-Summe, Hintereinanderschaltung der Windungen}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Usum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=x[i]+(imk-0.5)*Dd;      {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
{         Writeln('xs: ',xs*100:10:7,' cm, ys: ',ys*100:10:7,' cm, xk: ',xk*100:10:7,' cm, yk: ',yk*100:10:7,'
cm'); }
          Usum:=Usum+Uind11(ys,yk,Ikern,xk,xk-x[i]+x[i-1],xs);
        end;
      end;
    end;
  end;
  Uindges:=Usum;
end;

Function Fmech:Double;       {Federkraft der mechanischen Feder}
begin                                  {x-Achse positiv nach links}
  Fmech:=-D*(x[i-1]-xo);               {Federkraft negativ nach links}
end;

Procedure Lesen_Spulendaten;
Var fin  : Text;    {Die Daten aus dem genannten Programm-Lauf lesen}
    Name : String;  {Das zu lesende Datenfile}
    lv : Integer;
    Kommentar : Double;
begin
  Name:='Spule_merk-'+Lesen+'.dat';
  Writeln('Lesen des Spulen-Datenfiles: ',Name);
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,m);
  Readln(fin,n);
  Readln(fin,Dd);
  Readln(fin,di);
  Readln(fin,WikFak);
  Readln(fin,mk);
  Readln(fin,nk);
  Readln(fin,dk);
  Readln(fin,Br);
  Readln(fin,Kommentar);
  Readln(fin,Kommentar);
  Readln(fin,Kommentar);
  Readln(fin,Kommentar);
  Readln(fin,AnzSch);
  For lv:=-AnzSch to AnzSch do
  begin
    Readln(fin,XX[lv]); Readln(fin,YY[lv]);
  end;
  Close(fin);
end;

Procedure Spulengeometrie_ggf_uebernehmen(Name:String);
Var fin : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lokm,lokn,lokmk,loknk   : LongInt; {lokale Parameter zur Überprüfung}
    lokDd,lokdi,lokdk,lokBr : Double;  {lokale Parameter zur Überprüfung}
    loWikFak                : LongInt;
    lv : Integer;     Kommentar : String;
begin
  mussneurechnen:=false;
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,lokm);     If Abs((lokm-m)/(lokm+m))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokn);     If Abs((lokn-n)/(lokn+n))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokDd);    If Abs((lokDd-Dd)/(lokDd+Dd))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdi);    If Abs((lokdi-di)/(lokdi+di))>1E-10 then mussneurechnen:=true;
  Readln(fin,loWikFak); If Abs((loWikFak-WikFak)/(loWikFak+WikFak))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokmk);    If Abs((lokmk-mk)/(lokmk+mk))>1E-10 then mussneurechnen:=true;
  Readln(fin,loknk);    If Abs((loknk-nk)/(loknk+nk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdk);    If Abs((lokdk-dk)/(lokdk+dk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokBr);    If Abs((lokBr-Br)/(lokBr+Br))>1E-10 then mussneurechnen:=true;
  Readln(fin,Kommentar);  Readln(fin,Kommentar);  Readln(fin,Kommentar);  Readln(fin,Kommentar);
  Readln(fin,AnzSch);
  Writeln('>> Muessen Parameter neu gerechnet werden ?  = ',mussneurechnen);
  If Not(mussneurechnen) then
  begin
    For lv:=-AnzSch to AnzSch do
    begin
      Readln(fin,XX[lv]); Readln(fin,YY[lv]);
```

```
    end;
  end;
  Close(fin);
end;

Procedure Spulengeometrie_aufschreiben(Name:String);
Var fout : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lv : Integer;
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  Writeln(fout,m,'          Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander');
  Writeln(fout,n,'          Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander');
  Writeln(fout,Dd,'         Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)');
  Writeln(fout,di,'         Spuleninnendurchmesser, Spule 1');
  Writeln(fout,WikFak,'     Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz');
  Writeln(fout,mk,'         Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander');
  Writeln(fout,nk,'         Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander');
  Writeln(fout,dk,'         Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.');
  Writeln(fout,Br,'         remanentes longitudinales Feld des Magnetkerns');
  Writeln(fout,m*Dd,'       Spulenlänge');
  Writeln(fout,n*Dd,'       Spulenbreite');
  Writeln(fout,di+2*bs,'    Spulen-Aussendurchmesser');
  Writeln(fout,mk*Dd,'      Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.');
  Writeln(fout,AnzSch);
  For lv:=-AnzSch to AnzSch do
  begin
    Writeln(fout,XX[lv]); Writeln(fout,YY[lv]);
  end;
  Close(fout);
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  cl:=Sqrt(1/muo/epo){m/s};    {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',cl, ' m/s');
  Lesen:='00';                 {'00' ist: kein Spulen-File lesen, sondern neu rechnen}
  If Lesen='00' then
  begin
{   Spule 1, außen, Bleistift in *1 von S.36:}
    m:=7;                       {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
    n:=1;                       {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
    Dd:=2.5E-3; {Meter}         {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
    di:=0.06;   {Meter}         {Spuleninnendurchmesser, Spule 1}
    WikFak:=1;                  {Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz}
{   Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
    mk:=5;                      {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
    nk:=1;                      {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
    dk:=di*0.9;                 {Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
    Br:=1.8; {Tesla}            {remanentes longitudinales Feld des Magnetkerns}
  end;
  mussneurechnen:=true;
  If Lesen<>'00' then begin Lesen_Spulendaten; mussneurechnen:=false; end;
  If Lesen='00' then begin Spulengeometrie_ggf_uebernehmen('Spule_merk.dat'); end;
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
  AnzP:=AnzPmax;              {Anzahl der Zeitschritte insgesamt}
  dt:=0.005; {sec.}           {Größe der Zeitschritte}
  Abstd:=1;                   {Jeder wievielte Punkte soll geplottet werden}
{ Andere Komponenten (Kondensator und Widerstand):}
  C:=200E-6; {Farad}                              {Kapazität des Kondensators}
  rho:=1.7E-8; {Ohm*m}                            {Spez. Widerstand von Kupfer, je nach Temperatur,
Kohlrausch,T193}
  RD:=rho*(2*pi*di*n*m)/(pi*(Dd/2)*(Dd/2)){Ohm};  {Drahtwiderstand: Ohm`scher Widerstand des Spulendrahtes}
  RL:=0;                                          {Lastwiderstand, einstellbar je nach Verbraucher}
  R:=RL+RD;                                       {gesamter dämpfender Widerstand}
{ Mechanische Feder und mechanische Schwingung:}
  xo:=(m*Dd-mk*Dd)/2; {willkürlich}  {Ruhelageposition der Feder (ohne Federkraft)}
  rhoKern:=7.8E3;                    {Dichte des Magnetkern-Materials, Eisen, Kohlrausch Bd.3}
  D:=0.8; {N/m}                       {Hooke'sche Federoknstante}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;                {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                       {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                         {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  rupomag:=(ls-lk)/2;                {Ruhelageposition der Magnetkraft}
  dki:=dk-2*nk*Dd;                   {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);               {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
  L:=n*n*m*m*(pi*Sqr(da/2))/ls*Sqr(WikFak); {Induktivität der zylindrischen Spule, ohne Magnetkern drinnen, also
ohne mur}
  mo:=rhoKern*pi*Sqr(dk/2)*lk;       {Träge Masse des Magnetkerns}
  OMel:=1/Sqrt(L*C);                 {Eigen-Kreisfrequenz der harmonischen elektrischen Schwingkreises}
  OMmech:=Sqrt(D/mo);                {Eigen-Kreisfrequenz der mechanischen Schwingung}
  If (AnzP>AnzPmax) then
  begin
    Writeln ('AnzP = ',AnzP,' Zeitschritte koennen nicht verarbeitet werden. STOP.');
```

```
    Wait; Wait; Halt;
  end;
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Kapazitaet des Kondensators: C = ',C,' Farad');
  Writeln('Ohm`sche Widerstaende: R = ',R,' Ohm');
  Writeln('Induktivitaet der Spule ohne Magnetkern: L = ',L,' Henry');
  Writeln('Ruhelageposition der Magnetkraft: rupomag = ',rupomag*100:14:7,' cm');
  Writeln('Ruhelageposition der Feder (ohne Federkraft) xo = ',xo*100:14:7,' cm');
  Writeln('Traege Masse des Magnetkerns mo = ',mo,' kg');
  Writeln('Eigen-Kreisfreq harmon. el. Osz.: OMel = ',OMel:8:4,' Hz => T = ',2*pi/OMel:15,'sec.');
  Writeln('Eigen-Kreisfreq mechan. Schwingg: OMmech = ',OMmech:8:4,' Hz=> T= ',2*pi/OMmech:15,'sec.');
  Writeln('Analyseschritte: ',dt:14,' sec., Analysedauer: ',dt*AnzP:14,' sec.');
{ Hier beginnt das Rechenprogramm:}
{ Zuerst kommt ein Test der Magnetkraft zwischen Spule und Magnetkern:}
  If mussneurechnen then
  begin
    Writeln; Writeln('Vorbereitung: Kraft zwischen Spule und Magnetkern, als Fkt der x-Pos des Kerns.');
    XX[0]:=(ls-lk)/2; YY[0]:=Fges(XX[0]);        {Ruhelageposition ist immer bei X[0]}
    If Abs(YY[0])>1E-10 then
    begin
      Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',YY[0]:15:7,' N');
      Wait; Wait; Halt;
    end;
    sw:=(ls-XX[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
    AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
    For i:=1 to AnzSch do
    begin
      XX[i]:=XX[0]+i*SW;  YY[i]:=Fges(XX[i]);
      XX[-i]:=XX[0]-i*SW; YY[-i]:=Fges(XX[-i]);
    end;
    Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
      imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt das Kraftmaximum}
      If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
      begin
        i:=1;
        Repeat
          AnzAlt:=AnzSch;
          For j:=AnzAlt downto i do
          begin
            XX[j+1]:=XX[j];       YY[j+1]:=YY[j];
            XX[-j-1]:=XX[-j];     YY[-j-1]:=YY[-j];
          end;
          XX[i]:=(XX[i-1]+XX[i+1])/2;       YY[i]:=Fges(XX[i]);
          XX[-i]:=(XX[-i+1]+XX[-i-1])/2;    YY[-i]:=Fges(XX[-i]);
          AnzSch:=AnzSch+1;
          i:=i+2
        Until (i>2*imax+3);
      end;
    Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
    imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
    Repeat {Hier muß ich auch noch nach außen verlängern}
      AnzSch:=AnzSch+1;
      XX[AnzSch]:=XX[AnzSch-1]+sw;    YY[AnzSch]:=Fges(XX[AnzSch]);
      XX[-AnzSch]:=XX[-AnzSch+1]-sw;  YY[-AnzSch]:=Fges(XX[-AnzSch]);
    Until XX[AnzSch]>5*XX[imax];
  end;
{ Kontrolle der Kraft-Weg-Daten:}
{ For i:=-AnzSch to +AnzSch do Writeln(i,':',XX[i]*100:15:7,' cm',YY[i]:15:7,' N'); }
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Magnet-Kraftberechnung:}
{ For i:=-500 to +500 do
  begin xs[i]:=i/1000*(XX[+AnzSch]-XX[-AnzSch]); ys[i]:=Fs(xs[i]); end;
  XX:=xs;  YY:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel; }
{ Unter Angabe der Kernposition kann jetzt die Magnetkraft "Fs" aufgerufen werden.}

{Hier kommt ein Test der induzierten Spannung, die die Bewegung des Magnetkerns in der Spule 1 induziert:}
  i:=17; x[i]:=-0.0040; x[i-1]:=-0.0041;   {Zwecks Kontrolle des "Uingges"-Unterprogramms}
  Writeln('Induzierte Test-Spannung: Uind = ',Uindges,' Volt.');   {Alles ok, alle Hilfsprogramme sind fertig.}
{ Unter Angabe der Kernposition und ihrer Änderung kann jetzt den induzierte Spannung "Uindges" aufgerufen
werden.}

{ Zu guter Letzt folgt die Federkraft der mechanischen Feder:}
  i:=17; x[i-1]:=0.02; {Zwecks Kontrolle des "Fmech"-Unterprogramms}
  Writeln('Federkraft der mechanischen Feder: Fmech = ',Fmech,' N');
{ Unter Angabe der Kernposition kann jetzt "Fmech" aufgerufen werden.}

{ ------------------------------------------------------------------------------------------ }
{ Nun beginnt die DFEM-Simulation der Systems:}
{ Teil_01: Harmonische Schwingung}
  Writeln('Teil_01: Harmonische elektrische Schwingung: L,C');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
```

```
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------- }
{ Teil_02: Gedämpfte Schwingung}
  Writeln('Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------- }
{ Teil_03: Schwingung mit mechanischer Bewegung des Spulenkerns, Magnetkraft und induzierter Spannung}
  Writeln('Teil_03: Reine mechanische Bewegung des Spulenkerns, Schwingung');
  UC:=0;{Volt}   Q[0]:=C*UC;{Coulomb}    {Die Schwingung startet durch mechanische Anregung}
  Qpp[0]:=0;  Qp[0]:=0;                  {Startwerte der Integrationskonstanten}
  x[0]:=rupomag+0.10;{Meter}   xp[0]:=0;  xpp[0]:=0;  {Anfangswert: Initiale Auslenkung}
  For i:=1 to AnzP do  {Lsg. der Dgln.}
  begin
    xpp[i]:=(Fmech+WikFak*Fs(x[i-1]))/m; {Magnetkraft und Federkraft wirken,  WikFak beachten}
    xp[i]:=xp[i-1]+xpp[i]*dt;            K2[i]:=WikFak*Fs(x[i-1]);
    x[i]:=x[i-1]+xp[i]*dt;
{   Writeln(i,': Fs = ',WikFak*Fs(x[i-1]),' N, Fmech = ',Fmech,' N'); }
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    K1[i]:=-(Uindges*WikFak); Qpp[i]:=Qpp[i]+K1[i]/L;  {Induzierte Spannung nach *1 von S.51 in Spule 1 bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    If (n*m*nk*mk>200)and((i mod 100)=0) then write('*');
{   Wait; Wait; }
  end;
  ExcelAusgabe('test_03.dat',7,x,xp,xpp,Q,Qp,Qpp,K1,K2,x,x,x,x);

{ ------------------------------------------------------------------------------------------- }
  Wait;   Wait;       mussneurechnen:=true;
  If mussneurechnen then Spulengeometrie_aufschreiben('Spule_merk.dat');
End.
```

Var_L_013

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    cl       : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    L        : Double;  {Induktivität der Spule 1}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    WikFak   : LongInt; {Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    AnzP     : LongInt; {Anzahl der tatsächlich berechneten Zeit-Schritte}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    XX,YY,xs,ys : Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}
    C        : Double;  {Kapazität des Kondensators}
    rho,R    : Double;  {Spezifischer und Ohm'scher Widerstand des Spulendrahtes}
    RD,RL    : Double;  {Drahtwiderstand des Spulendrahtes und Lastwiderstand, beide Ohm'sch}
    Q,Qp,Qpp    : Feld;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Feld;  {Auslenkung jeder einzelnen Kondensatorplatte}
    K1,K2,K3    : Feld;  {Drei Kontroll-Felder für Plot-Zwecke}
    OMel,OMmech : Double;  {elektrische und mechanische Kreisfrequenz der schwingenden Systeme}
    UC,Ul       : Double;  {Kondensatorspannung}
    xo          : Double;  {Ruhelageposition der Feder (ohne Federkraft)}
    mo          : Double;  {Träge Masse des Magnetkerns}
    rhoKern     : Double;  {Träge Masse des Magnetkerns}
    D           : Double;  {Hooke'sche Federoknstante}
    rupomag     : Double;  {Ruhelageposition der Magnetkraft}
    mussneurechnen : Boolean; {Können die Parameter der Spulengeometrie übernommen werden ?}
    Lesen          : String;  {Nummer der zu lesenden Spulendaten}
    UAmpl          : Double;  {Spannungsamplitude der getriggerten Anregung}
    Pulsdauer      : LongInt; {Pulsdauer der gertiggerten Anregung in Einheiten von dt}
    Umk            : LongInt; {Umkehrpunkt, Angabe in "i" der Zeitschritte-Zählung}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
```

```
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(XX[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(YY[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;
```

```
Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;  If n*m>50 then write('Dauer*',n*m,',  ');
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xpos<=XX[-lv] then merk:=YY[-lv];
  If xpos>=XX[+lv] then merk:=YY[+lv];
  If (xpos>XX[-lv])and(xpos<XX[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until XX[lv]>xpos;
    merk:=YY[lv-1]+(xpos-XX[lv-1])/(XX[lv]-XX[lv-1])*(YY[lv]-YY[lv-1]);  {Lineare Interpolation}
  end;                  {Fges => XX,YY waren mit 1.0 Amp. Strom in Spule 1 gerechnet worden.}
  Fs:=merk*qp[i-1];     {Die endgültige Magnetkraft als Funktion des Stroms sei linear skaliert.}
end;

Function Uind11(r1,r2,I2,x2i,x2im1,x1:Double):Double;
Var nen1,nen2,Klam : Double; {Induzierte Spannung Einzelwindung zu Einzelwindung, zwei Nenner, Klammer}
 Function mur:Double;       {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil,Bx,nen : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   nen:=Sqr(r1)+Sqr(r1-r2);
   Bx:=muo*qp[i]*Sqr(r1)/2/Sqrt(nen*nen*nen); {Erzeugt die Spule 1 am Ort des Magnetkerns.}
   Anteil:=(muo*Bx)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
  nen1:=Sqr(r2)+Sqr(x1-x2i);
  nen2:=Sqr(r2)+Sqr(x1-x2im1);
  Klam:=1/Sqrt(nen1*nen1*nen1)-1/Sqrt(nen2*nen2*nen2);
  Uind11:=-muo*mur*pi*Sqr(r1)*Sqr(r2)*I2/2/dt*Klam; {Verwenden wir einen Magnetkern mit mur=1}
end;
```

```
Function Uindges:Double;  {Hier brauche ich kein Schnellrechen-Verfahren, weil keine Integration Zeit kostet.}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Usum : Double;        {Spannungs-Summe, Hintereinanderschaltung der Windungen}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Usum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=x[i]+(imk-0.5)*Dd;      {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
{         Writeln('xs: ',xs*100:10:7,' cm, ys: ',ys*100:10:7,' cm, xk: ',xk*100:10:7,' cm, yk: ',yk*100:10:7,'
cm'); }
          Usum:=Usum+Uind11(ys,yk,Ikern,xk,xk-x[i]+x[i-1],xs);
        end;
      end;
    end;
  end;
  Uindges:=Usum;
end;

Function Fmech:Double;       {Federkraft der mechanischen Feder}
begin                                    {x-Achse positiv nach links}
  Fmech:=-D*(x[i-1]-xo);                 {Federkraft negativ nach links}
end;

Procedure Lesen_Spulendaten;
Var fin  : Text;    {Die Daten aus dem genannten Programm-Lauf lesen}
    Name : String;  {Das zu lesende Datenfile}
    lv : Integer;
    Kommentar : Double;
begin
  Name:='Spule_merk-'+Lesen+'.dat';
  Writeln('Lesen des Spulen-Datenfiles: ',Name);
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,m);
  Readln(fin,n);
  Readln(fin,Dd);
  Readln(fin,di);
  Readln(fin,WikFak);
  Readln(fin,mk);
  Readln(fin,nk);
  Readln(fin,dk);
  Readln(fin,Br);
  Readln(fin,Kommentar);   m:=m+0*Round(Kommentar);
  Readln(fin,Kommentar);   m:=m+0*Round(Kommentar);
  Readln(fin,Kommentar);   m:=m+0*Round(Kommentar);
  Readln(fin,Kommentar);   m:=m+0*Round(Kommentar);
  Readln(fin,AnzSch);      m:=m+0*Round(Kommentar);
  For lv:=-AnzSch to AnzSch do
  begin
    Readln(fin,XX[lv]); Readln(fin,YY[lv]);
  end;
  Close(fin);
end;

Procedure Spulengeometrie_ggf_uebernehmen(Name:String);
Var fin : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lokm,lokn,lokmk,loknk   : LongInt; {lokale Parameter zur Überprüfung}
    lokDd,lokdi,lokdk,lokBr : Double;  {lokale Parameter zur Überprüfung}
    loWikFak                : LongInt;
    lv : Integer;     Kommentar : String;
begin
  mussneurechnen:=false;
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,lokm);     If Abs((lokm-m)/(lokm+m))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokn);     If Abs((lokn-n)/(lokn+n))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokDd);    If Abs((lokDd-Dd)/(lokDd+Dd))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdi);    If Abs((lokdi-di)/(lokdi+di))>1E-10 then mussneurechnen:=true;
  Readln(fin,loWikFak); If Abs((loWikFak-WikFak)/(loWikFak+WikFak))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokmk);    If Abs((lokmk-mk)/(lokmk+mk))>1E-10 then mussneurechnen:=true;
  Readln(fin,loknk);    If Abs((loknk-nk)/(loknk+nk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdk);    If Abs((lokdk-dk)/(lokdk+dk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokBr);    If Abs((lokBr-Br)/(lokBr+Br))>1E-10 then mussneurechnen:=true;
  Readln(fin,Kommentar);  Readln(fin,Kommentar);  Readln(fin,Kommentar);  Readln(fin,Kommentar);
  Readln(fin,AnzSch);
  Writeln('>> Muessen Parameter neu gerechnet werden ?  = ',mussneurechnen);
  If Not(mussneurechnen) then
  begin
```

```
    For lv:=-AnzSch to AnzSch do
    begin
      Readln(fin,XX[lv]); Readln(fin,YY[lv]);
    end;
  end;
  Close(fin);
end;

Procedure Spulengeometrie_aufschreiben(Name:String);
Var fout : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lv : Integer;
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  Writeln(fout,m,'          Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander');
  Writeln(fout,n,'          Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander');
  Writeln(fout,Dd,'         Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)');
  Writeln(fout,di,'         Spuleninnendurchmesser, Spule 1');
  Writeln(fout,WikFak,'     Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz');
  Writeln(fout,mk,'         Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander');
  Writeln(fout,nk,'         Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander');
  Writeln(fout,dk,'         Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.');
  Writeln(fout,Br,'         remanentes longitudinales Feld des Magnetkerns');
  Writeln(fout,m*Dd,'       Spulenlänge');
  Writeln(fout,n*Dd,'       Spulenbreite');
  Writeln(fout,di+2*bs,'    Spulen-Aussendurchmesser');
  Writeln(fout,mk*Dd,'      Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.');
  Writeln(fout,AnzSch);
  For lv:=-AnzSch to AnzSch do
  begin
    Writeln(fout,XX[lv]); Writeln(fout,YY[lv]);
  end;
  Close(fout);
end;

Function Uin(zp:LongInt):Double;
Var Umerk : Double;  {Spannung merken}
begin
  Umerk:=0;
  If zp<=Pulsdauer then Umerk:=UAmpl;   {Start-Impuls geben}
  If zp>=Pulsdauer then                 {Getriggerte Pulse im Betrieb geben}
  begin
    If xp[zp-1]*xp[zp]<=0 then            {Umkehrpunkte der Bewegung bestimmen}
    begin
      Umk:=zp; Writeln(zp,' Umkehrpunkt bei ',x[zp]);
    end;
    If (zp>=Umk)and(zp<=(Umk+Pulsdauer)) then
    begin
      If x[zp]>0 then Umerk:=UAmpl;  {Spannung anlegen nach dem oberen Umkehrpunkt}
    end;
  end;
  Uin:=Umerk;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  cl:=Sqrt(1/muo/epo){m/s};    {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',cl, ' m/s');
  Lesen:='00';                 {'00' ist: kein Spulen-File lesen, sondern neu rechnen}
  If Lesen='00' then
  begin
{   Spule 1, außen, Bleistift in *1 von S.36:}
    m:=15;                      {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
    n:=2;                       {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
    Dd:=5E-3; {Meter}         {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
    di:=0.06;   {Meter}         {Spuleninnendurchmesser, Spule 1}
    WikFak:=1;                  {Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz}
{   Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
    mk:=6;                     {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
    nk:=2;                      {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
    dk:=di*0.9;                 {Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
    Br:=1.8; {Tesla}            {remanentes longitudinales Feld des Magnetkerns}
  end;
  mussneurechnen:=true;
  If Lesen<>'00' then begin Lesen_Spulendaten; mussneurechnen:=false; end;
  If Lesen='00' then begin Spulengeometrie_ggf_uebernehmen('Spule_merk.dat'); end;
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
  AnzP:=AnzPmax;              {Anzahl der Zeitschritte insgesamt}
  dt:=0.005; {sec.}           {Größe der Zeitschritte}
  Abstd:=1;                   {Jeder wievielte Punkte soll geplottet werden}
{ Andere Komponenten (Kondensator und Widerstand):}
  C:=200E-6; {Farad}                            {Kapazität des Kondensators}
  rho:=1.7E-8; {Ohm*m}                          {Spez. Widerstand von Kupfer, je nach Temperatur,
Kohlrausch,T193}
```

```
  RD:=rho*(2*pi*di*n*m)/(pi*(Dd/2)*(Dd/2)){Ohm};   {Drahtwiderstand: Ohm`scher Widerstand des Spulendrahtes}
  RL:=0;                                           {Lastwiderstand, einstellbar je nach Verbraucher}
  R:=RL+RD;                                        {gesamter dämpfender Widerstand}
{ Mechanische Feder und mechanische Schwingung:}
  xo:=(m*Dd-mk*Dd)/2; {willkürlich}  {Ruhelageposition der Feder (ohne Federkraft)}
  rhoKern:=7.8E3;                    {Dichte des Magnetkern-Materials, Eisen, Kohlrausch Bd.3}
  D:=0.8; {N/m}                      {Hooke'sche Federoknstante}
{ Getriggerte Anregungsspannung:}
  UAmpl:=0.10;{Volt}                 {Spannungsamplitude der getriggerten Anregung}
  Pulsdauer:=200;                    {Pulsdauer der gertiggerten Anregung in Einheiten von dt}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;                {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                       {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                         {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  rupomag:=(ls-lk)/2;                {Ruhelageposition der Magnetkraft}
  dki:=dk-2*nk*Dd;                   {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);               {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
  L:=n*n*m*m*(pi*Sqr(da/2))/ls*Sqr(WikFak); {Induktivität der zylindrischen Spule, ohne Magnetkern drinnen, also
ohne mur}
  mo:=rhoKern*pi*Sqr(dk/2)*lk;       {Träge Masse des Magnetkerns}
  OMel:=1/Sqrt(L*C);                 {Eigen-Kreisfrequenz der harmonischen elektrischen Schwingkreises}
  OMmech:=Sqrt(D/mo);                {Eigen-Kreisfrequenz der mechanischen Schwingung}
  If (AnzP>AnzPmax) then
  begin
    Writeln ('AnzP = ',AnzP,' Zeitschritte koennen nicht verarbeitet werden. STOP.');
    Wait; Wait; Halt;
  end;
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Kapazitaet des Kondensators: C = ',C,' Farad');
  Writeln('Ohm`sche Widerstaende: R = ',R,' Ohm');
  Writeln('Induktivitaet der Spule ohne Magnetkern: L = ',L,' Henry');
  Writeln('Ruhelageposition der Magnetkraft: rupomag = ',rupomag*100:14:7,' cm');
  Writeln('Ruhelageposition der Feder (ohne Federkraft) xo = ',xo*100:14:7,' cm');
  Writeln('Traege Masse des Magnetkerns mo = ',mo,' kg');
  Writeln('Eigen-Kreisfreq harmon. el. Osz.: OMel = ',OMel:8:4,' Hz => T = ',2*pi/OMel:15,'sec.');
  Writeln('Eigen-Kreisfreq mechan. Schwingg: OMmech = ',OMmech:8:4,' Hz=> T= ',2*pi/OMmech:15,'sec.');
  Writeln('Analyseschritte: ',dt:14,' sec., Analysedauer: ',dt*AnzP:14,' sec.');
{ Hier beginnt das Rechenprogramm:}
{ Zuerst kommt ein Test der Magnetkraft zwischen Spule und Magnetkern:}
  If mussneurechnen then
  begin
    Writeln; Writeln('Vorbereitung: Kraft zwischen Spule und Magnetkern, als Fkt der x-Pos des Kerns.');
    XX[0]:=(ls-lk)/2; YY[0]:=Fges(XX[0]);        {Ruhelageposition ist immer bei X[0]}
    If Abs(YY[0])>1E-10 then
    begin
      Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',YY[0]:15:7,' N');
      Wait; Wait; Halt;
    end;
    sw:=(ls-XX[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
    AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
    For i:=1 to AnzSch do
    begin
      XX[i]:=XX[0]+i*SW;  YY[i]:=Fges(XX[i]);
      XX[-i]:=XX[0]-i*SW; YY[-i]:=Fges(XX[-i]);
    end;
    Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
      imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt das Kraftmaximum}
      If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
      begin
        i:=1;
        Repeat
          AnzAlt:=AnzSch;
          For j:=AnzAlt downto i do
          begin
            XX[j+1]:=XX[j];       YY[j+1]:=YY[j];
            XX[-j-1]:=XX[-j];     YY[-j-1]:=YY[-j];
          end;
          XX[i]:=(XX[i-1]+XX[i+1])/2;       YY[i]:=Fges(XX[i]);
          XX[-i]:=(XX[-i+1]+XX[-i-1])/2;    YY[-i]:=Fges(XX[-i]);
          AnzSch:=AnzSch+1;
          i:=i+2
        Until (i>2*imax+3);
      end;
    Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
    imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
    Repeat {Hier muß ich auch noch nach außen verlängern}
      AnzSch:=AnzSch+1;
      XX[AnzSch]:=XX[AnzSch-1]+sw;    YY[AnzSch]:=Fges(XX[AnzSch]);
      XX[-AnzSch]:=XX[-AnzSch+1]-sw;  YY[-AnzSch]:=Fges(XX[-AnzSch]);
    Until XX[AnzSch]>5*XX[imax];
  end;
{ Kontrolle der Kraft-Weg-Daten:}
{ For i:=-AnzSch to +AnzSch do Writeln(i,':',XX[i]*100:15:7,' cm',YY[i]:15:7,' N'); }
```

```
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Magnet-Kraftberechnung:}
{ For i:=-500 to +500 do
  begin xs[i]:=i/1000*(XX[+AnzSch]-XX[-AnzSch]); ys[i]:=Fs(xs[i]); end;
  XX:=xs;  YY:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel; }
{ Unter Angabe der Kernposition kann jetzt die Magnetkraft "Fs" aufgerufen werden.}

{Hier kommt ein Test der induzierten Spannung, die die Bewegung des Magnetkerns in der Spule 1 induziert:}
  i:=17; x[i]:=-0.0040; x[i-1]:=-0.0041;   {Zwecks Kontrolle des "Uingges"-Unterprogramms}
  Writeln('Induzierte Test-Spannung: Uind = ',Uindges,' Volt.');   {Alles ok, alle Hilfsprogramme sind fertig.}
{ Unter Angabe der Kernposition und ihrer Änderung kann jetzt den induzierte Spannung "Uindges" aufgerufen
werden.}

{ Zu guter Letzt folgt die Federkraft der mechanischen Feder:}
  i:=17; x[i-1]:=0.02; {Zwecks Kontrolle des "Fmech"-Unterprogramms}
  Writeln('Federkraft der mechanischen Feder: Fmech = ',Fmech,' N');
{ Unter Angabe der Kernposition kann jetzt "Fmech" aufgerufen werden.}

{ ------------------------------------------------------------------------------------------ }
{ Nun beginnt die DFEM-Simulation der Systems:}
{ Teil_01: Harmonische Schwingung}
  Writeln('Teil_01: Harmonische elektrische Schwingung: L,C');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------ }
{ Teil_02: Gedämpfte Schwingung}
  Writeln('Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------ }
{ Teil_03: Schwingung mit mechanischer Bewegung des Spulenkerns, Magnetkraft und induzierter Spannung}
  Writeln('Teil_03: Reine mechanische Bewegung des Spulenkerns, Schwingung');
  UC:=0;{Volt}   Q[0]:=C*UC;{Coulomb}    {Anfangsbedingungen: ruhendes System}
  Qpp[0]:=0;  Qp[0]:=0;                  {Anregung erfolgt später durch Input-Pulse am Kondensator}
  x[0]:=rupomag+0.00;{Meter}   xp[0]:=0;  xpp[0]:=0;  {Keine mechanische Anregung}
  For i:=1 to AnzP do  {Lsg. der Dgln.}
  begin
    xpp[i]:=(Fmech+WikFak*Fs(x[i-1]))/m; {Magnetkraft und Federkraft wirken,  WikFak beachten}
    xp[i]:=xp[i-1]+xpp[i]*dt;            K2[i]:=WikFak*Fs(x[i-1]);
    x[i]:=x[i-1]+xp[i]*dt;
{   Writeln(i,': Fs = ',WikFak*Fs(x[i-1]),' N, Fmech = ',Fmech,' N'); }
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    K1[i]:=-(Uindges*WikFak); Qpp[i]:=Qpp[i]+K1[i]/L;  {Induzierte Spannung nach *1 von S.51 in Spule 1 bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    K3[i]:=Uin(i);                    {Anregungs-Spannung auf dem Oszi anschauen}
    Q[i]:=Q[i-1]+Qp[i]*dt+C*K3[i];   {Hier: Die Anregungs-Spannung über den Kondensator einkoppeln}
    If (n*m*nk*mk>200)and((i mod 100)=0) then write('*');
  end;
  ExcelAusgabe('test_03.dat',9,x,xp,xpp,Q,Qp,Qpp,K1,K2,K3,x,x,x);

{ ------------------------------------------------------------------------------------------ }
  Wait;   Wait;       mussneurechnen:=true;
  If mussneurechnen then Spulengeometrie_aufschreiben('Spule_merk.dat');
End.
```

Var_L_014

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo  : Double;  {Naturkonstanten}
    cl       : Double;  {Lichtgeschwindigkeit}
    ls,bs    : Double;  {Spulenlänge,Spulenbreite, Schleifen 1 nach *2 von S.40}
    L        : Double;  {Induktivität der Spule 1}
    di,da    : Double;  {Innendurchmesser und Außendurchmesser des Spulenkörpers}
    Dd       : Double;  {Drahtdurchmesser}
    n,m      : LongInt; {Windungszahl, radial und axial der Spule}
    dk,lk    : Double;  {Spulenkern (Magnetkern) -> Durchmesser und Länge, Schleifen 2 nach *2 von S.40}
    dki      : Double;  {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
    nk,mk    : LongInt; {Windungszahl, radial und axial zur Simulation des Magnetkerns}
    WikFak   : LongInt; {Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz}
    dt       : Double;  {Dauer der Zeitschritte zur Lsg. der Dgl.}
    AnzP     : LongInt; {Anzahl der tatsächlich berechneten Zeit-Schritte}
    Abstd    : Integer; {Jeder wievielte Punkte soll geplottet werden}
    Ikern    : Double;  {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
    Br       : Double;  {remanentes longitudinales Feld des Magnetkerns}
    i,j      : LongInt; {nur für Testzwecke: Laufvariable}
    AnzSch,AnzAlt : Integer; {Anzahl der Stützpunkte für die schnelle Kraftberechnung}
    XX,YY,xs,ys : Array[-500..+500] of Double; {Maximal 1000 Stützpunkte für die schnelle Kraftberechnung}
    sw       : Double;  {Schrittweite für die Kraftermittlung}
    imax     : Integer; {An dieser Stelle liegt das Kraftmaximum}
    xkern    : Double;  {x-Position des Magnetkerns}
    C        : Double;  {Kapazität des Kondensators}
    rho,R    : Double;  {Spezifischer und Ohm'scher Widerstand des Spulendrahtes}
    RD,RL    : Double;  {Drahtwiderstand des Spulendrahtes und Lastwiderstand, beide Ohm'sch}
    Q,Qp,Qpp    : Feld;  {Ladung auf dem Kondensator als Fkt der Zeit}
    x,xp,xpp    : Feld;  {Auslenkung jeder einzelnen Kondensatorplatte}
    K1,K2,K3    : Feld;  {Drei Kontroll-Felder für Plot-Zwecke}
    OMel,OMmech : Double;  {elektrische und mechanische Kreisfrequenz der schwingenden Systeme}
    UC,Ul       : Double;  {Kondensatorspannung}
    xo          : Double;  {Ruhelageposition der Feder (ohne Federkraft)}
    mo          : Double;  {Träge Masse des Magnetkerns}
    rhoKern     : Double;  {Träge Masse des Magnetkerns}
    D           : Double;  {Hooke'sche Federoknstante}
    rupomag     : Double;  {Ruhelageposition der Magnetkraft}
    mussneurechnen : Boolean; {Können die Parameter der Spulengeometrie übernommen werden ?}
    Lesen          : String;  {Nummer der zu lesenden Spulendaten}
    UAmpl          : Double;  {Spannungsamplitude der getriggerten Anregung}
    Pulsdauer      : LongInt; {Pulsdauer der gertiggerten Anregung in Einheiten von dt}
    Umk            : LongInt; {Umkehrpunkt, Angabe in "i" der Zeitschritte-Zählung}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
```

```
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Procedure Kraftwerte_ins_Excel;
Var fout   : Text;
    lv,j,k : Integer; {Laufvariablen}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,'Kraftwerte.dat'); Rewrite(fout); {File öffnen}
  For lv:=-AnzSch to AnzSch do  {von "plotanf" bis "plotend"}
  begin
    For j:=1 to 2 do
    begin   {Kolumnen drucken}
      If j=1  then Str(XX[lv]*100:17:11,Zahl);  {Zentimeter}
      If j=2  then Str(YY[lv]:17:11,Zahl);      {Newton}
      For k:=1 to Length(Zahl) do
      begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
        If Zahl[k]<>'.' then write(fout,Zahl[k]);
        If Zahl[k]='.' then write(fout,',');
      end;
      Write(fout,chr(9));  {Daten-Trennung, Tabulator}
    end;
    Writeln(fout,'');    {Zeilen-Trennung}
  end;
  Close(fout);
end;

Function Fx(r1,r2,I1,I2,x1,x2:Double):Double; {Kraft zwischen den beiden Leiterschleifen (in x-Richtung)}
Var q1,om1,Hy : Double;
 Function Fkt(phi:Double):Double; {Die zu integrierende Funktion}
 Var Nen : Double;
 begin
   Nen:=Sqr(x1-x2)+Sqr(r1*sin(phi)-r2)+Sqr(r1*cos(phi));
   Fkt:=-r1*om1*sin(phi)*(x1-x2)/Sqrt(Nen*Nen*Nen);
 end;
 Function Simpson(n:LongInt):Double;  {Numerische Integration nach dem Simpson-Verfahren}
 Var ug,og : Double;  {Integralgrenzen}
     i     : LongInt; {Laufvariable}
     sum   : Double;  {Wert des Integrals}
 begin
   ug:=0;  og:=2*pi;  {Untergrenze und Obergrenze der Integration}
   Sum:=Fkt(ug)+Fkt(og);
   For i:=1 to 2*n-1 do
   begin
     If (i mod 2)=0 then Sum:=Sum+2*Fkt(ug+i*(og-ug)/2/n);
     If (i mod 2)=1 then Sum:=Sum+4*Fkt(ug+i*(og-ug)/2/n);
   end;
   Simpson:=Sum*(og-ug)/2/n/3;
 end;
 Function Integral:Double;
 Var Streif       : LongInt;
     Wert1,Wert2  : Double;
 begin
   Streif:=1000;
   Wert2:=Simpson(Streif);
   Repeat
     Wert1:=Wert2;
     Streif:=Streif*2;
     Wert2:=Simpson(Streif);
   Until (Wert2-Wert1)<1E-8;
   Integral:=Wert2;
 end;
 Function mur:Double;      {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   Anteil:=(muo*Hy)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
{ For i:=0 to 20 do begin Hy:=i*1E5; Writeln(muo*Hy,' Tesla => mur = ',mur); end; Wait; Wait; Halt; {Kontrolle von
mur}
  q1:=1;  om1:=I1*2*pi/q1;
  Hy:=I1/4/pi/om1*Integral;
  Fx:=-muo*mur*I1*I2/2/om1*r2*Hy;
end;
```

```
Function Remanenz(Br:Double):Double;  {Simulation als longitudinales Feld in x-Richtung in der Kern-Achse}
Var Bist   : Double;  {Ist-Wert der magnetischen Induktion}
    Istart : Double;  {Strom-Anfangswert zu Berechnungszwecken}
 Function H:Double;
 Var ax              : Double;  {Aufpunkt zur Bestimmung der Feldstärke}
     i,j             : LongInt;
     Hij,sum,kl1,kl2 : Double;
 begin
   ax:=lk; {Aufpunkt zur Bestimmung der Feldstärke: Er liegt am Ende des Magnetkerns}
   Istart:=1; {Ampere}     {Strom-Anfangswert zu Berechnungszwecken}
   sum:=0; {Feldstärke}
   For i:=1 to nk do
   begin
     For j:=1 to mk do
     begin
       kl1:=Sqr(dki/2+(i-0.5)*Dd);
       kl2:=Sqr(ax-(j-0.5)*Dd);
       Hij:=Istart*kl1/2/Sqrt((kl1+kl2)*(kl1+kl2)*(kl1+kl2));
       sum:=sum+Hij;
     end;
   end;
   H:=sum;
 end;
begin
  Bist:=muo*H;
  Remanenz:=Istart*Br/Bist;
end;

Function Fges(kernpos:Double):Double; {Gesamtkraft zwischen Spule und Magnetkern, x = Position des Magnetkerns}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Fsum : Double;        {Kraft-Summe}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Fsum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=kernpos+(imk-0.5)*Dd;   {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
          Fsum:=Fsum+Fx(ys,yk,1.00,Ikern,xs,xk);
        end;
      end;  If n*m>50 then write('Dauer*',n*m,',  ');
    end;    Write('.');
  end;
  Fges:=Fsum;
end;

Function Fs(xpos:Double):Double;
Var lv   : Integer;
    merk : Double;
begin
  lv:=AnzSch;     merk:=0;
  If xpos<=XX[-lv] then merk:=YY[-lv];
  If xpos>=XX[+lv] then merk:=YY[+lv];
  If (xpos>XX[-lv])and(xpos<XX[+lv]) then
  begin
    lv:=-AnzSch-1; Repeat lv:=lv+1; Until XX[lv]>xpos;
    merk:=YY[lv-1]+(xpos-XX[lv-1])/(XX[lv]-XX[lv-1])*(YY[lv]-YY[lv-1]);  {Lineare Interpolation}
  end;                  {Fges => XX,YY waren mit 1.0 Amp. Strom in Spule 1 gerechnet worden.}
  Fs:=merk*qp[i-1];     {Die endgültige Magnetkraft als Funktion des Stroms sei linear skaliert.}
end;

Function Uind11(r1,r2,I2,x2i,x2im1,x1:Double):Double;
Var nen1,nen2,Klam : Double; {Induzierte Spannung Einzelwindung zu Einzelwindung, zwei Nenner, Klammer}
 Function mur:Double;       {Hier kann eine echte Hysterese-Schleife definiert werden !}
 Var Wert,Anteil,Bx,nen : Double; {Sogar Ummagnetisierungs-Verluste können abgebildet werden.}
 begin
   nen:=Sqr(r1)+Sqr(r1-r2);
   Bx:=muo*qp[i]*Sqr(r1)/2/Sqrt(nen*nen*nen); {Erzeugt die Spule 1 am Ort des Magnetkerns.}
   Anteil:=(muo*Bx)/2.00;  {Anteil an der Sättigungsmagnetisierung von 2.00 Tesla.}
   Wert:=1+100*Anteil;     {Bei der Berechnung des MUR muß ich ggf. eine Sättigung berücksichtigen}
   Wert:=1+0*Wert;         {Falls ich ohne MUR arbeiten will.}
   mur:=Wert;              {Zunächst verwende ich ein Material mit mur=1}
 end;
begin
  nen1:=Sqr(r2)+Sqr(x1-x2i);
  nen2:=Sqr(r2)+Sqr(x1-x2im1);
  Klam:=1/Sqrt(nen1*nen1*nen1)-1/Sqrt(nen2*nen2*nen2);
  Uind11:=-muo*mur*pi*Sqr(r1)*Sqr(r2)*I2/2/dt*Klam; {Verwenden wir einen Magnetkern mit mur=1}
end;
```

```
Function Uindges:Double;  {Hier brauche ich kein Schnellrechen-Verfahren, weil keine Integration Zeit kostet.}
Var ins,ims : LongInt;    {Zählvariablen für alle Windungen der Spule}
    ink,imk : LongInt;    {Zählvariablen für alle Windungen des Magnetkerns}
    Usum : Double;        {Spannungs-Summe, Hintereinanderschaltung der Windungen}
    xs,ys,xk,yk : Double; {Positionen der Windungen}
begin
  Usum:=0;
  For ins:=1 to n do {Spule radial}
  begin
    For ims:=1 to m do {Spule axial}
    begin
      For ink:=1 to nk do {Kern radial}
      begin
        For imk:=1 to mk do {Kern axial}
        begin
          xs:=(ims-0.5)*Dd;           {Spule axial}
          ys:=di/2+(ins-0.5)*Dd;      {Spule radial, radiale Position entspricht dem halben Durchmesser}
          xk:=x[i]+(imk-0.5)*Dd;      {Kern axial}
          yk:=dki/2+(ink-0.5)*Dd;     {Kern radial, radiale Position entspricht dem halben Durchmesser}
{         Writeln('xs: ',xs*100:10:7,' cm, ys: ',ys*100:10:7,' cm, xk: ',xk*100:10:7,' cm, yk: ',yk*100:10:7,'
cm'); }
          Usum:=Usum+Uind11(ys,yk,Ikern,xk,xk-x[i]+x[i-1],xs);
        end;
      end;
    end;
  end;
  Uindges:=Usum;
end;

Function Fmech:Double;       {Federkraft der mechanischen Feder}
begin                                    {x-Achse positiv nach links}
  Fmech:=-D*(x[i-1]-xo);                 {Federkraft negativ nach links}
end;

Procedure Lesen_Spulendaten;
Var fin  : Text;    {Die Daten aus dem genannten Programm-Lauf lesen}
    Name : String;  {Das zu lesende Datenfile}
    lv : Integer;
    Kommentar : Double;
begin
  Name:='Spule_merk-'+Lesen+'.dat';
  Writeln('Lesen des Spulen-Datenfiles: ',Name);
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,m);
  Readln(fin,n);
  Readln(fin,Dd);
  Readln(fin,di);
  Readln(fin,WikFak);
  Readln(fin,mk);
  Readln(fin,nk);
  Readln(fin,dk);
  Readln(fin,Br);
  Readln(fin,Kommentar);   m:=m+0*Round(Kommentar);
  Readln(fin,Kommentar);   m:=m+0*Round(Kommentar);
  Readln(fin,Kommentar);   m:=m+0*Round(Kommentar);
  Readln(fin,Kommentar);   m:=m+0*Round(Kommentar);
  Readln(fin,AnzSch);      m:=m+0*Round(Kommentar);
  For lv:=-AnzSch to AnzSch do
  begin
    Readln(fin,XX[lv]); Readln(fin,YY[lv]);
  end;
  Close(fin);
end;

Procedure Spulengeometrie_ggf_uebernehmen(Name:String);
Var fin : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lokm,lokn,lokmk,loknk   : LongInt; {lokale Parameter zur Überprüfung}
    lokDd,lokdi,lokdk,lokBr : Double;  {lokale Parameter zur Überprüfung}
    loWikFak                : LongInt;
    lv : Integer;     Kommentar : String;
begin
  mussneurechnen:=false;
  Assign(fin,Name); Reset(fin); {File öffnen}
  Readln(fin,lokm);     If Abs((lokm-m)/(lokm+m))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokn);     If Abs((lokn-n)/(lokn+n))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokDd);    If Abs((lokDd-Dd)/(lokDd+Dd))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdi);    If Abs((lokdi-di)/(lokdi+di))>1E-10 then mussneurechnen:=true;
  Readln(fin,loWikFak); If Abs((loWikFak-WikFak)/(loWikFak+WikFak))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokmk);    If Abs((lokmk-mk)/(lokmk+mk))>1E-10 then mussneurechnen:=true;
  Readln(fin,loknk);    If Abs((loknk-nk)/(loknk+nk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokdk);    If Abs((lokdk-dk)/(lokdk+dk))>1E-10 then mussneurechnen:=true;
  Readln(fin,lokBr);    If Abs((lokBr-Br)/(lokBr+Br))>1E-10 then mussneurechnen:=true;
  Readln(fin,Kommentar);  Readln(fin,Kommentar);  Readln(fin,Kommentar);  Readln(fin,Kommentar);
  Readln(fin,AnzSch);
  Writeln('>> Muessen Parameter neu gerechnet werden ?  = ',mussneurechnen);
  If Not(mussneurechnen) then
  begin
```

```
    For lv:=-AnzSch to AnzSch do
    begin
      Readln(fin,XX[lv]); Readln(fin,YY[lv]);
    end;
  end;
  Close(fin);
end;

Procedure Spulengeometrie_aufschreiben(Name:String);
Var fout : Text;    {Die Daten aus dem letzten Programm-Lauf}
    lv : Integer;
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  Writeln(fout,m,'          Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander');
  Writeln(fout,n,'          Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander');
  Writeln(fout,Dd,'         Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)');
  Writeln(fout,di,'         Spuleninnendurchmesser, Spule 1');
  Writeln(fout,WikFak,'     Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz');
  Writeln(fout,mk,'         Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander');
  Writeln(fout,nk,'         Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander');
  Writeln(fout,dk,'         Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.');
  Writeln(fout,Br,'         remanentes longitudinales Feld des Magnetkerns');
  Writeln(fout,m*Dd,'       Spulenlänge');
  Writeln(fout,n*Dd,'       Spulenbreite');
  Writeln(fout,di+2*bs,'    Spulen-Aussendurchmesser');
  Writeln(fout,mk*Dd,'      Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.');
  Writeln(fout,AnzSch);
  For lv:=-AnzSch to AnzSch do
  begin
    Writeln(fout,XX[lv]); Writeln(fout,YY[lv]);
  end;
  Close(fout);
end;

Function Uin(zp:LongInt):Double;
Var Umerk : Double;  {Spannung merken}
begin
  Umerk:=0;
  If zp<=Pulsdauer then Umerk:=UAmpl;   {Start-Impuls geben}
  If zp>=Pulsdauer then                 {Getriggerte Pulse im Betrieb geben}
  begin
    If xp[zp-1]*xp[zp]<=0 then            {Umkehrpunkte der Bewegung bestimmen}
    begin
      Umk:=zp; Writeln(zp,' Umkehrpunkt bei ',x[zp]);
    end;
    If (zp>=Umk)and(zp<=(Umk+Pulsdauer)) then
    begin
      If x[zp]>0 then Umerk:=UAmpl;  {Spannung anlegen nach dem oberen Umkehrpunkt}
    end;
  end;
  Uin:=Umerk;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Leistungsstarker Raumenergie-Konverter: Versuch mit variabler Spule');  Writeln;
{ Allgemeine Werte-Vorgaben -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  cl:=Sqrt(1/muo/epo){m/s};    {Lichtgeschwindigkeit} Writeln('Lichtgeschwindigkeit c = ',cl, ' m/s');
  Lesen:='00';                 {'00' ist: kein Spulen-File lesen, sondern neu rechnen}
  If Lesen='00' then
  begin
{   Spule 1, außen, Bleistift in *1 von S.36:}
    m:=15;                      {Spule 1: Windungszahl axial, Anzahl der Windungen pro Lage, nebeneinander}
    n:=2;                       {Spule 1: Windungszahl radial, Anzahl der Lagen, übereinander}
    Dd:=5E-3; {Meter}         {Drahtdurchmesser (für Spule 1 und ebenso für Magnetkern-Simulation)}
    di:=0.06;   {Meter}         {Spuleninnendurchmesser, Spule 1}
    WikFak:=1;                  {Faktor: Wicklungszahl-Erhöhung bei schneller Konvergenz}
{   Magnetkern, entspricht Leiterschleifen 2, innen, grün in *1 von S.36:}
    mk:=6;                     {Magnetkern, Leiter Nr.2: Windungszahl axial, Anzahl der Windungen pro Lage,
nebeneinander}
    nk:=2;                      {Magnetkern, Leiter Nr.2: Windungszahl radial, Anzahl der Lagen, übereinander}
    dk:=di*0.9;                 {Magnetkern-Außendurchmesser -> Der Magnetkern füllt die Spule nicht ganz aus.}
    Br:=1.8; {Tesla}            {remanentes longitudinales Feld des Magnetkerns}
  end;
  mussneurechnen:=true;
  If Lesen<>'00' then begin Lesen_Spulendaten; mussneurechnen:=false; end;
  If Lesen='00' then begin Spulengeometrie_ggf_uebernehmen('Spule_merk.dat'); end;
{ mur : Die Permeabilität wird über ein Upgm. eingeführt, damit Hysterese-Schleifen abgebildet werden können }
  AnzP:=AnzPmax;              {Anzahl der Zeitschritte insgesamt}
  dt:=0.005; {sec.}           {Größe der Zeitschritte}
  Abstd:=1;                   {Jeder wievielte Punkte soll geplottet werden}
{ Andere Komponenten (Kondensator und Widerstand):}
  C:=200E-6; {Farad}                            {Kapazität des Kondensators}
  rho:=1.7E-8; {Ohm*m}                          {Spez. Widerstand von Kupfer, je nach Temperatur,
Kohlrausch,T193}
```

```
  RD:=rho*(2*pi*di*n*m)/(pi*(Dd/2)*(Dd/2)){Ohm};   {Drahtwiderstand: Ohm`scher Widerstand des Spulendrahtes}
  RL:=0;                                           {Lastwiderstand, einstellbar je nach Verbraucher}
  R:=RL+RD;                                        {gesamter dämpfender Widerstand}
{ Mechanische Feder und mechanische Schwingung:}
  xo:=(m*Dd-mk*Dd)/2; {willkürlich}  {Ruhelageposition der Feder (ohne Federkraft)}
  rhoKern:=7.8E3;                    {Dichte des Magnetkern-Materials, Eisen, Kohlrausch Bd.3}
  D:=0.8; {N/m}                      {Hooke'sche Federoknstante}
{ Getriggerte Anregungsspannung:}
  UAmpl:=0.10;{Volt}                 {Spannungsamplitude der getriggerten Anregung}
  Pulsdauer:=200;                    {Pulsdauer der gertiggerten Anregung in Einheiten von dt}
{ Abgeleitete Werte, keine Eingabe möglich: Die Werte dienen dem bequemeren Rechnen:}
  ls:=m*Dd; bs:=n*Dd;                {Spulenlänge,Spulenbreite}
  da:=di+2*bs;                       {Spulen-Aussendurchmesser}
  lk:=mk*Dd;                         {Magnetkern-Länge -> Der Magnetkern füllt die Spule aus.}
  rupomag:=(ls-lk)/2;                {Ruhelageposition der Magnetkraft}
  dki:=dk-2*nk*Dd;                   {Hilfsgröße: Innendurchmesser zur Simulation der Magnetkern-Windungen}
  Ikern:=Remanenz(Br);               {Strom(DC) zwecks Simulation des Feldes des Magnetkerns}
  L:=n*n*m*m*(pi*Sqr(da/2))/ls*Sqr(WikFak); {Induktivität der zylindrischen Spule, ohne Magnetkern drinnen, also
ohne mur}
  mo:=rhoKern*pi*Sqr(dk/2)*lk;       {Träge Masse des Magnetkerns}
  OMel:=1/Sqrt(L*C);                 {Eigen-Kreisfrequenz der harmonischen elektrischen Schwingkreises}
  OMmech:=Sqrt(D/mo);                {Eigen-Kreisfrequenz der mechanischen Schwingung}
  If (AnzP>AnzPmax) then
  begin
    Writeln ('AnzP = ',AnzP,' Zeitschritte koennen nicht verarbeitet werden. STOP.');
    Wait; Wait; Halt;
  end;
{ Anzeige der Werte:}
  Writeln('Spulenlaenge ls = ',ls*100:9:5,' cm, und Spulenkoerper-Breite bs = ',bs*100:9:5,' cm ');
  Writeln('Spulen: Innendrm di = ',di*100:9:5,' cm, Aussendrm da = ',da*100:9:5,' cm ');
  Writeln('Magnetkern: Durchmesser, dk = ',dk*100:9:5,' cm, Laenge lk = ',lk*100:9:5,' cm ');
  Writeln('Innendrm der Magnetkern-Simulations-Wdgn: dki = ',dki*100:9:5,' cm');
  Writeln('Magnetkern-Strom Ikern zur Simulation des remanenten Feldes:',Ikern:15:5,' Amp');
  Writeln('Kapazitaet des Kondensators: C = ',C,' Farad');
  Writeln('Ohm`sche Widerstaende: R = ',R,' Ohm');
  Writeln('Induktivitaet der Spule ohne Magnetkern: L = ',L,' Henry');
  Writeln('Ruhelageposition der Magnetkraft: rupomag = ',rupomag*100:14:7,' cm');
  Writeln('Ruhelageposition der Feder (ohne Federkraft) xo = ',xo*100:14:7,' cm');
  Writeln('Traege Masse des Magnetkerns mo = ',mo,' kg');
  Writeln('Eigen-Kreisfreq harmon. el. Osz.: OMel = ',OMel:8:4,' Hz => T = ',2*pi/OMel:15,'sec.');
  Writeln('Eigen-Kreisfreq mechan. Schwingg: OMmech = ',OMmech:8:4,' Hz=> T= ',2*pi/OMmech:15,'sec.');
  Writeln('Analyseschritte: ',dt:14,' sec., Analysedauer: ',dt*AnzP:14,' sec.');
{ Hier beginnt das Rechenprogramm:}
{ Zuerst kommt ein Test der Magnetkraft zwischen Spule und Magnetkern:}
  If mussneurechnen then
  begin
    Writeln; Writeln('Vorbereitung: Kraft zwischen Spule und Magnetkern, als Fkt der x-Pos des Kerns.');
    XX[0]:=(ls-lk)/2; YY[0]:=Fges(XX[0]);        {Ruhelageposition ist immer bei X[0]}
    If Abs(YY[0])>1E-10 then
    begin
      Writeln('Stoerung. Kraft in Ruhelageposition nicht Null: ',YY[0]:15:7,' N');
      Wait; Wait; Halt;
    end;
    sw:=(ls-XX[0])/5;     {Erster Einstieg für die Schrittweite der Kraftberechnung}
    AnzSch:=8; {Gesamtzahl der Schritte ist 2*AnzSch+1, nämlich von -AnzSch .. +AnzSch}
    For i:=1 to AnzSch do
    begin
      XX[i]:=XX[0]+i*SW;  YY[i]:=Fges(XX[i]);
      XX[-i]:=XX[0]-i*SW; YY[-i]:=Fges(XX[-i]);
    end;
    Repeat  {Hier muß ich falls nötig die Schrittweite verfeinern}
      imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt das Kraftmaximum}
      If imax<7 then  {In diesem Fall muß ich zusätzliche Stützpunkte dazwischenschieben}
      begin
        i:=1;
        Repeat
          AnzAlt:=AnzSch;
          For j:=AnzAlt downto i do
          begin
            XX[j+1]:=XX[j];       YY[j+1]:=YY[j];
            XX[-j-1]:=XX[-j];     YY[-j-1]:=YY[-j];
          end;
          XX[i]:=(XX[i-1]+XX[i+1])/2;       YY[i]:=Fges(XX[i]);
          XX[-i]:=(XX[-i+1]+XX[-i-1])/2;    YY[-i]:=Fges(XX[-i]);
          AnzSch:=AnzSch+1;
          i:=i+2
        Until (i>2*imax+3);
      end;
    Until (imax>=7);   {Jetzt sollte die Schrittweite fein genug sein.}
    imax:=0; Repeat imax:=imax+1; Until -YY[imax+1]<-YY[imax];  {An dieser Stelle liegt jetzt das Kraftmaximum}
    Repeat {Hier muß ich auch noch nach außen verlängern}
      AnzSch:=AnzSch+1;
      XX[AnzSch]:=XX[AnzSch-1]+sw;    YY[AnzSch]:=Fges(XX[AnzSch]);
      XX[-AnzSch]:=XX[-AnzSch+1]-sw;  YY[-AnzSch]:=Fges(XX[-AnzSch]);
    Until XX[AnzSch]>5*XX[imax];
  end;
{ Kontrolle der Kraft-Weg-Daten:}
{ For i:=-AnzSch to +AnzSch do Writeln(i,':',XX[i]*100:15:7,' cm',YY[i]:15:7,' N'); }
```

```
  Kraftwerte_ins_Excel;

{ Hier kommt ein Test der schnellen Magnet-Kraftberechnung:}
{ For i:=-500 to +500 do
  begin xs[i]:=i/1000*(XX[+AnzSch]-XX[-AnzSch]); ys[i]:=Fs(xs[i]); end;
  XX:=xs;  YY:=ys;   AnzSch:=500;   Kraftwerte_ins_Excel; }
{ Unter Angabe der Kernposition kann jetzt die Magnetkraft "Fs" aufgerufen werden.}

{Hier kommt ein Test der induzierten Spannung, die die Bewegung des Magnetkerns in der Spule 1 induziert:}
  i:=17; x[i]:=-0.0040; x[i-1]:=-0.0041;   {Zwecks Kontrolle des "Uingges"-Unterprogramms}
  Writeln('Induzierte Test-Spannung: Uind = ',Uindges,' Volt.');   {Alles ok, alle Hilfsprogramme sind fertig.}
{ Unter Angabe der Kernposition und ihrer Änderung kann jetzt den induzierte Spannung "Uindges" aufgerufen
werden.}

{ Zu guter Letzt folgt die Federkraft der mechanischen Feder:}
  i:=17; x[i-1]:=0.02; {Zwecks Kontrolle des "Fmech"-Unterprogramms}
  Writeln('Federkraft der mechanischen Feder: Fmech = ',Fmech,' N');
{ Unter Angabe der Kernposition kann jetzt "Fmech" aufgerufen werden.}

{ ------------------------------------------------------------------------------------------ }
{ Nun beginnt die DFEM-Simulation der Systems:}
{ Teil_01: Harmonische Schwingung}
  Writeln('Teil_01: Harmonische elektrische Schwingung: L,C');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------ }
{ Teil_02: Gedämpfte Schwingung}
  Writeln('Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;  Qp[0]:=0;                {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ ------------------------------------------------------------------------------------------ }
{ Teil_03: Schwingung mit mechanischer Bewegung des Spulenkerns, Magnetkraft und induzierter Spannung}
  Writeln('Teil_03: Reine mechanische Bewegung des Spulenkerns, Schwingung');
  UC:=0;{Volt}   Q[0]:=C*UC;{Coulomb}    {Anfangsbedingungen: ruhendes System}
  Qpp[0]:=0;  Qp[0]:=0;                  {Anregung erfolgt später durch Input-Pulse am Kondensator}
  x[0]:=rupomag+0.00;{Meter}   xp[0]:=0;  xpp[0]:=0;  {Keine mechanische Anregung}
  For i:=1 to AnzP do  {Lsg. der Dgln.}
  begin
    xpp[i]:=(Fmech+WikFak*Fs(x[i-1]))/m; {Magnetkraft und Federkraft wirken,  WikFak beachten}
    xp[i]:=xp[i-1]+xpp[i]*dt;            K2[i]:=WikFak*Fs(x[i-1]);
    x[i]:=x[i-1]+xp[i]*dt;
{   Writeln(i,': Fs = ',WikFak*Fs(x[i-1]),' N, Fmech = ',Fmech,' N'); }
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    K1[i]:=-(Uindges*WikFak); Qpp[i]:=Qpp[i]+K1[i]/L;  {Induzierte Spannung nach *1 von S.51 in Spule 1 bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    K3[i]:=Uin(i);                    {Anregungs-Spannung auf dem Oszi anschauen}
    Q[i]:=Q[i-1]+Qp[i]*dt+C*K3[i];   {Hier: Die Anregungs-Spannung über den Kondensator einkoppeln}
    If (n*m*nk*mk>200)and((i mod 100)=0) then write('*');
  end;
  ExcelAusgabe('test_03.dat',9,x,xp,xpp,Q,Qp,Qpp,K1,K2,K3,x,x,x);

{ ------------------------------------------------------------------------------------------ }
  Wait;   Wait;       mussneurechnen:=true;
  If mussneurechnen then Spulengeometrie_aufschreiben('Spule_merk.dat');
End.
```

Es folgt Teil 4

Var_L_015

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
```

```
    end;
  end;
  Close(fout);
end;

Function Mx:Double;    {x-Komponente des Drehmoments}
begin
  Mx:=Bo*n*Qp[i]*A*sin(phi[i]);
end;

Function Ui:Double;    {Induzierte Spannung}
begin
  Ui:=n*Bo*A*sin(phi[i])*phip[i];
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=50000;   {Windungszahl der Spule}
  di:=0.1;    {Spulenkörper-Durchmesser}
  Dd:=0.0005; {Drahtdurchmesser}
  Bo:=1.8;    {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;   {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=750E-6;  {Farad} {Kapazität des Kondensators}
  rm:=0.035;  {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;   {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;        {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.005; {sec.}       {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
{ Hier beginnt das Rechenprogramm.}

{ Test der Drehmomentsberechnung:}
{ For i:=0 to 360 do
  begin
    Qp[i]:=1.0; {Ampere}
{   phi[i]:=i/360*2*pi;
    writeln(i:5,': phi => Mx : ',phi[i],' => ',Mx);
{   Wait; }
{ end; }

{ Test der Induzierten Spannung:}
{ For i:=0 to 360 do
  begin
    phi[i]:=i/360*2*pi;
    phip[i]:=7; {rad pro sec.}
{    writeln(i:5,': phi => Ui : ',phi[i],' => ',Ui);
    Wait;
  end; }

{ Teil_01: Harmonische Schwingung}
  Writeln(' -> Teil 1: Elektrischer Schwingkreis');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_02: Gedämpfte Schwingung}
```

```
  Writeln(' -> Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_03: Mechanische Drehung}
  Writeln(' -> Teil_03: Mechanische Drehung');
  phi[0]:=0;  phip[0]:=0.5;  phipp[0]:=0; {Startwerte für die Integrationskonst.}
  For i:=1 to AnzP do
  begin
    qp[i]:=0.0000004; {Ampere}  {Spulenstrom geben, damit eine Magnetkraft existiert.}
    phipp[i]:=Bo*n*qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i-1]*dt;
    phi[i]:=phi[i-1]+phip[i-1]*dt;
  end;
  ExcelAusgabe('test_03.dat',6,Q,Qp,Qpp,phi,phip,phipp,Q,Q,Q,Q,Q,Q);

  Wait;   Wait;
End.
```

Var_L_016

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Mx:Double;    {x-Komponente des Drehmoments}
begin
```

```
  Mx:=Bo*n*Qp[i]*A*sin(phi[i]);
end;

Function Ui:Double;    {Induzierte Spannung}
begin
  Ui:=n*Bo*A*sin(phi[i])*phip[i];
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=50000;   {Windungszahl der Spule}
  di:=0.1;    {Spulenkörper-Durchmesser}
  Dd:=0.0005; {Drahtdurchmesser}
  Bo:=0.0004; {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;   {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=750E-6;  {Farad} {Kapazität des Kondensators}
  rm:=0.035;  {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;   {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;        {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.005; {sec.}       {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
{ Hier beginnt das Rechenprogramm.}

{ Teil_01: Harmonische Schwingung}
  Writeln(' -> Teil 1: Elektrischer Schwingkreis');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_02: Gedämpfte Schwingung}
  Writeln(' -> Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_03: Mechanische Drehung}
  Writeln(' -> Teil_03: Mechanische Drehung');
  phi[0]:=0;  phip[0]:=0.5;  phipp[0]:=0; {Startwerte für die Integrationskonst.}
  For i:=1 to AnzP do
  begin
    qp[i]:=0.0000004; {Ampere}  {Spulenstrom geben, damit eine Magnetkraft existiert.}
    phipp[i]:=Bo*n*qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
  end;
  ExcelAusgabe('test_03.dat',6,Q,Qp,Qpp,phi,phip,phipp,Q,Q,Q,Q,Q,Q);
```

```
{ Teil_03: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;  Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;  phip[0]:=0.1;  phipp[0]:=0;        {Mechanische Startwerte}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i])*phip[i]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=Bo*n*qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
  end;
  ExcelAusgabe('test_04.dat',6,Q,Qp,Qpp,phi,phip,phipp,Q,Q,Q,Q,Q);

  Wait;   Wait;
End.
```

Var_L_017

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
end;

Function Mx:Double;    {x-Komponente des Drehmoments}
begin
```

```
  Mx:=Bo*n*Qp[i]*A*sin(phi[i]);
end;

Function Ui:Double;    {Induzierte Spannung}
begin
  Ui:=n*Bo*A*sin(phi[i])*phip[i];
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=50000;   {Windungszahl der Spule}
  di:=0.1;    {Spulenkörper-Durchmesser}
  Dd:=0.0005; {Drahtdurchmesser}
  Bo:=0.0005; {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;   {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=750E-6;  {Farad} {Kapazität des Kondensators}
  rm:=0.035;  {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;   {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;        {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.04; {sec.}       {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
{ Hier beginnt das Rechenprogramm.}

{ Teil_01: Harmonische Schwingung}
  Writeln(' -> Teil 1: Elektrischer Schwingkreis');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_02: Gedämpfte Schwingung}
  Writeln(' -> Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_03: Mechanische Drehung}
  Writeln(' -> Teil_03: Mechanische Drehung');
  phi[0]:=0;  phip[0]:=0.5;  phipp[0]:=0; {Startwerte für die Integrationskonst.}
  For i:=1 to AnzP do
  begin
    qp[i]:=0.0000004; {Ampere}  {Spulenstrom geben, damit eine Magnetkraft existiert.}
    phipp[i]:=Bo*n*qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
  end;
  ExcelAusgabe('test_03.dat',6,Q,Qp,Qpp,phi,phip,phipp,Q,Q,Q,Q,Q,Q);
```

```
{ Teil_04: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;  Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;  phip[0]:=0.1;  phipp[0]:=0;        {Mechanische Startwerte}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
  end;
  ExcelAusgabe('test_04.dat',6,Q,Qp,Qpp,phi,phip,phipp,Q,Q,Q,Q,Q);

  Wait;   Wait;
End.
```

Var_L_018

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,php,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}
    K0,K1,K2,K3,K4,K5 : Feld;  {Kontroll-Felder für Plot-Zwecke}
    EmA,EmE,siA,siE : Double;  {Energie: Mittelwerte und Sigma "Anfang" und "Ende"}
    delE,sigdelE    : Double;  {Veränderung der Energie-Mittelwerte "Anfang" zu "Ende"}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
  Close(fout);
```

```
end;

Function Mx:Double;    {x-Komponente des Drehmoments}
begin
  Mx:=Bo*n*Qp[i]*A*sin(phi[i]);
end;

Function Ui:Double;    {Induzierte Spannung}
begin
  Ui:=n*Bo*A*sin(phi[i])*phip[i];
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=2000;     {Windungszahl der Spule}
  di:=0.09;    {Spulenkörper-Durchmesser}
  Dd:=0.0010;  {Drahtdurchmesser}
  Bo:=0.0051;  {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;    {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=4070E-6;  {Farad} {Kapazität des Kondensators}
  rm:=0.035;   {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;    {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;         {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.005; {sec.}       {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R:8:2,' Ohm');                R:=0; {Achtung !!}
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
  Writeln('Gesamtdauer der Betrachtung: ',AnzP*dt,' sec.');
{ Hier beginnt das Rechenprogramm.}

{ Teil_01: Harmonische Schwingung}
  Writeln(' -> Teil 1: Elektrischer Schwingkreis');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_02: Gedämpfte Schwingung}
  Writeln(' -> Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_03: Mechanische Drehung}
  Writeln(' -> Teil_03: Mechanische Drehung');
  phi[0]:=0;  phip[0]:=0.5;  phipp[0]:=0; {Startwerte für die Integrationskonst.}
  For i:=1 to AnzP do
  begin
    qp[i]:=0.0000004; {Ampere}  {Spulenstrom geben, damit eine Magnetkraft existiert.}
    phipp[i]:=Bo*n*qp[i]*A/J*sin(phi[i-1]);
```

```
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
  end;
  ExcelAusgabe('test_03.dat',6,Q,Qp,Qpp,phi,phip,phipp,Q,Q,Q,Q,Q,Q);

{ Teil_04: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;   Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;  phip[0]:=2*pi;  phipp[0]:=0;          {Mechanische Startwerte}
  K0[0]:=0;
  K1[0]:=1/2*L*Sqrt(Qp[0]);  {Spulen-Energie}
  K2[0]:=1/2*C*Sqr(Q[0]/C);  {Kondensator-Energie}
  K3[0]:=1/2*J*Sqr(phip[0]); {Rotations-Energie}
  K4[0]:=K1[i]+K2[i]+K3[0];  {Gesamt-Energie}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
    K0[i]:=0;  K1[i]:=0;  K2[i]:=0;  K3[i]:=0;  K4[i]:=0;  K5[i]:=0;
    K1[i]:=1/2*L*Sqr(Qp[i]);   {Spulen-Energie}
    K2[i]:=1/2*C*Sqr(Q[i]/C);  {Kondensator-Energie}
    K3[i]:=1/2*J*Sqr(phip[i]); {Rotations-Energie}
    K4[i]:=K1[i]+K2[i]+K3[i];  {Gesamt-Energie}
  end;
{ Gesamt-Energie-Bilanz und Anzeige:}
  EmA:=0; EmE:=0; siA:=0; siE:=0;
  For i:=1 to 10 do  EmA:=EmA+K4[i]/10;              {Mittelwert zu Beginn}
  For i:=AnzP-9 to AnzP do EmE:=EmE+K4[i]/10;        {Mittelwert am Ende}
  For i:=1 to 10 do siA:=siA+Sqr(EmA-K4[i]);         {Varianz zu Beginn}
  For i:=AnzP-9 to AnzP do siE:=siE+Sqr(EmE-K4[i]);  {Varianz am Ende}
  siA:=Sqrt(siA)/10;  siE:=Sqrt(siE)/10;             {Standardabweichungen}
  Writeln('Energie-Werte:  E_Anfang = (',EmA:11:7,' +/- ',siA:11:7,') Joule');
  Writeln('                E_Ende   = (',EmE:11:7,' +/- ',siE:11:7,') Joule');
  delE:=EmE-EmA;  sigdelE:=Sqrt(Sqr(siE)+Sqr(siA));
  Writeln('=> Veraenderung: delta_E = (',delE:11:7,' +/- ',sigdelE:11:7,') Joule');
  Writeln('=> Konvertierte Leistung = (',1000*delE/(AnzP*dt):11:7,' +/- ',1000*sigdelE/(AnzP*dt):11:7,')
milliWatt');
  ExcelAusgabe('test_04.dat',11,Q,Qp,Qpp,phi,phip,phipp,K0,K1,K2,K3,K4,K0);

  Wait;   Wait;
End.
```

Var_L_019

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,php,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}
    K0,K1,K2,K3,K4,K5 : Feld;  {Kontroll-Felder für Plot-Zwecke}
    EmA,EmE,siA,siE : Double;  {Energie: Mittelwerte und Sigma "Anfang" und "Ende"}
    delE,sigdelE    : Double;  {Veränderung der Energie-Mittelwerte "Anfang" zu "Ende"}
    UmAn            : Double;  {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
```

```
  Close(fout);
end;

Function Mx:Double;    {x-Komponente des Drehmoments}
begin
  Mx:=Bo*n*Qp[i]*A*sin(phi[i]);
end;

Function Ui:Double;    {Induzierte Spannung}
begin
  Ui:=n*Bo*A*sin(phi[i])*phip[i];
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=1000;     {Windungszahl der Spule}
  di:=0.09;    {Spulenkörper-Durchmesser}
  Dd:=0.0010;  {Drahtdurchmesser}
  Bo:=0.0067;  {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;    {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=6.9E-6;   {Farad} {Kapazität des Kondensators}
  rm:=0.035;   {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;    {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;         {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  UmAn:=50;               {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.005; {sec.}       {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R:8:2,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
  Writeln('Gesamtdauer der Betrachtung: ',AnzP*dt,' sec.');
{ Hier beginnt das Rechenprogramm.}

{ Teil_01: Harmonische Schwingung}
  Writeln(' -> Teil 1: Elektrischer Schwingkreis');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_02: Gedämpfte Schwingung}
  Writeln(' -> Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_03: Mechanische Drehung}
  Writeln(' -> Teil_03: Mechanische Drehung');
  phi[0]:=0;  phip[0]:=0.5;  phipp[0]:=0; {Startwerte für die Integrationskonst.}
  For i:=1 to AnzP do
  begin
```

```
    qp[i]:=0.0000004; {Ampere}  {Spulenstrom geben, damit eine Magnetkraft existiert.}
    phipp[i]:=Bo*n*qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
  end;
  ExcelAusgabe('test_03.dat',6,Q,Qp,Qpp,phi,phip,phipp,Q,Q,Q,Q,Q,Q);

{ Teil_04: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;          Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;    phip[0]:=UmAn*2*pi;  phipp[0]:=0;          {Mechanische Startwerte}
  K0[0]:=0;
  K1[0]:=1/2*L*Sqrt(Qp[0]);  {Spulen-Energie}
  K2[0]:=1/2*C*Sqr(Q[0]/C);  {Kondensator-Energie}
  K3[0]:=1/2*J*Sqr(phip[0]); {Rotations-Energie}
  K4[0]:=K1[i]+K2[i]+K3[0];  {Gesamt-Energie}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
    K0[i]:=0;  K1[i]:=0;  K2[i]:=0;  K3[i]:=0;  K4[i]:=0;  K5[i]:=0;
    K1[i]:=1/2*L*Sqr(Qp[i]);   {Spulen-Energie}
    K2[i]:=1/2*C*Sqr(Q[i]/C);  {Kondensator-Energie}
    K3[i]:=1/2*J*Sqr(phip[i]); {Rotations-Energie}
    K4[i]:=K1[i]+K2[i]+K3[i];  {Gesamt-Energie}
  end;
{ Gesamt-Energie-Bilanz und Anzeige:}
  EmA:=0; EmE:=0; siA:=0; siE:=0;
  For i:=1 to 10 do  EmA:=EmA+K4[i]/10;              {Mittelwert zu Beginn}
  For i:=AnzP-9 to AnzP do EmE:=EmE+K4[i]/10;        {Mittelwert am Ende}
  For i:=1 to 10 do siA:=siA+Sqr(EmA-K4[i]);         {Varianz zu Beginn}
  For i:=AnzP-9 to AnzP do siE:=siE+Sqr(EmE-K4[i]);  {Varianz am Ende}
  siA:=Sqrt(siA)/10;  siE:=Sqrt(siE)/10;             {Standardabweichungen}
  Writeln('Energie-Werte:  E_Anfang = (',EmA:11:7,' +/- ',siA:11:7,') Joule');
  Writeln('                E_Ende   = (',EmE:11:7,' +/- ',siE:11:7,') Joule');
  delE:=EmE-EmA;  sigdelE:=Sqrt(Sqr(siE)+Sqr(siA));
  Writeln('=> Veraenderung: delta_E = (',delE:11:7,' +/- ',sigdelE:11:7,') Joule');
  Writeln('=> Konvertierte Leistung = (',delE/(AnzP*dt):11:7,' +/- ',sigdelE/(AnzP*dt):11:7,') Watt');
  ExcelAusgabe('test_04.dat',11,Q,Qp,Qpp,phi,phip,phipp,K0,K1,K2,K3,K4,K0);

  Wait;   Wait;
End.
```

Var_L_020

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}
    K0,K1,K2,K3,K4,K5 : Feld;  {Kontroll-Felder für Plot-Zwecke}
    EmA,EmE,siA,siE : Double;  {Energie: Mittelwerte und Sigma "Anfang" und "Ende"}
    delE,sigdelE    : Double;  {Veränderung der Energie-Mittelwerte "Anfang" zu "Ende"}
    UmAn            : Double;  {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
```

```
  Close(fout);
end;

Function Mx:Double;    {x-Komponente des Drehmoments}
begin
  Mx:=Bo*n*Qp[i]*A*sin(phi[i]);
end;

Function Ui:Double;    {Induzierte Spannung}
begin
  Ui:=n*Bo*A*sin(phi[i])*phip[i];
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=1000;     {Windungszahl der Spule}
  di:=0.09;    {Spulenkörper-Durchmesser}
  Dd:=0.0010;  {Drahtdurchmesser}
  Bo:=0.2000;  {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;    {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=6.9E-6;   {Farad} {Kapazität des Kondensators}
  rm:=0.035;   {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;    {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;         {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  UmAn:=50;               {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.001; {sec.}       {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R:8:2,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
  Writeln('Gesamtdauer der Betrachtung: ',AnzP*dt,' sec.');
{ Hier beginnt das Rechenprogramm.}

{ Teil_01: Harmonische Schwingung}
  Writeln(' -> Teil 1: Elektrischer Schwingkreis');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{ Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do  {Zunächst noch ohne Daempfung}
  begin
    UC:=Q[i-1]/C; UL:=-UC;
    Qpp[i]:=UL/L;
    Qp[i]:=Qp[i-1]+Qpp[i]*dt;
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_01.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_02: Gedämpfte Schwingung}
  Writeln(' -> Teil_02: Gedaempfte elektrische Schwingung: L,C,R');
  UC:=10;{Volt}  Q[0]:=C*UC;{Coulomb}  {Benötigt eine elektrische Spannung zum Starten}
  Qpp[0]:=0;     Qp[0]:=0;             {Startwerte für die Integrationskonstanten}
{  Writeln('  t/[sec.]  |  Uc/[V] | '); }
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
{   Writeln(i*dt:11:9,' | ',Q[i]/C:7:2,' |'); }
  end;
  ExcelAusgabe('test_02.dat',3,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp,Q,Qp,Qpp);

{ Teil_03: Mechanische Drehung}
  Writeln(' -> Teil_03: Mechanische Drehung');
  phi[0]:=0;  phip[0]:=0.5;  phipp[0]:=0; {Startwerte für die Integrationskonst.}
  For i:=1 to AnzP do
  begin
```

```
    qp[i]:=0.0000004; {Ampere}  {Spulenstrom geben, damit eine Magnetkraft existiert.}
    phipp[i]:=Bo*n*qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
  end;
  ExcelAusgabe('test_03.dat',6,Q,Qp,Qpp,phi,phip,phipp,Q,Q,Q,Q,Q,Q);

{ Teil_04: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;          Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;    phip[0]:=UmAn*2*pi;  phipp[0]:=0;          {Mechanische Startwerte}
  K0[0]:=0;
  K1[0]:=1/2*L*Sqrt(Qp[0]);  {Spulen-Energie}
  K2[0]:=1/2*C*Sqr(Q[0]/C);  {Kondensator-Energie}
  K3[0]:=1/2*J*Sqr(phip[0]); {Rotations-Energie}
  K4[0]:=K1[i]+K2[i]+K3[0];  {Gesamt-Energie}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]);
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
    K0[i]:=0;  K1[i]:=0;  K2[i]:=0;  K3[i]:=0;  K4[i]:=0;  K5[i]:=0;
    K1[i]:=1/2*L*Sqr(Qp[i]);   {Spulen-Energie}
    K2[i]:=1/2*C*Sqr(Q[i]/C);  {Kondensator-Energie}
    K3[i]:=1/2*J*Sqr(phip[i]); {Rotations-Energie}
    K4[i]:=K1[i]+K2[i]+K3[i];  {Gesamt-Energie}
  end;
{ Gesamt-Energie-Bilanz und Anzeige:}
  EmA:=0; EmE:=0; siA:=0; siE:=0;
  For i:=1 to 10 do  EmA:=EmA+K4[i]/10;              {Mittelwert zu Beginn}
  For i:=AnzP-9 to AnzP do EmE:=EmE+K4[i]/10;        {Mittelwert am Ende}
  For i:=1 to 10 do siA:=siA+Sqr(EmA-K4[i]);         {Varianz zu Beginn}
  For i:=AnzP-9 to AnzP do siE:=siE+Sqr(EmE-K4[i]);  {Varianz am Ende}
  siA:=Sqrt(siA)/10;  siE:=Sqrt(siE)/10;             {Standardabweichungen}
  Writeln('Energie-Werte:  E_Anfang = (',EmA:11:7,' +/- ',siA:11:7,') Joule');
  Writeln('                E_Ende   = (',EmE:11:7,' +/- ',siE:11:7,') Joule');
  delE:=EmE-EmA;  sigdelE:=Sqrt(Sqr(siE)+Sqr(siA));
  Writeln('=> Veraenderung: delta_E = (',delE:11:7,' +/- ',sigdelE:11:7,') Joule');
  Writeln('=> Konvertierte Leistung = (',delE/(AnzP*dt):11:7,' +/- ',sigdelE/(AnzP*dt):11:7,') Watt');
  ExcelAusgabe('test_04.dat',11,Q,Qp,Qpp,phi,phip,phipp,K0,K1,K2,K3,K4,K0);

  Wait;   Wait;
End.
```

Var_L_021

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}
    K0,K1,K2,K3,K4,K5 : Feld;  {Kontroll-Felder für Plot-Zwecke}
    EmA,EmE,siA,siE : Double;  {Energie: Mittelwerte und Sigma "Anfang" und "Ende"}
    delE,sigdelE    : Double;  {Veränderung der Energie-Mittelwerte "Anfang" zu "Ende"}
    UmAn            : Double;  {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
```

```
  Close(fout);
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=1000;     {Windungszahl der Spule}
  di:=0.09;    {Spulenkörper-Durchmesser}
  Dd:=0.0010;  {Drahtdurchmesser}
  Bo:=0.5000;  {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;    {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=1.60E-6;   {Farad} {Kapazität des Kondensators}
  rm:=0.035;   {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;    {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;         {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  UmAn:=50;               {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.0001; {sec.}       {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R:8:2,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
  Writeln('Gesamtdauer der Betrachtung: ',AnzP*dt,' sec.');
{ Hier beginnt das Rechenprogramm.}

{ Teil_04: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;          Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;    phip[0]:=UmAn*2*pi;  phipp[0]:=0;          {Mechanische Startwerte}
  K0[0]:=0;
  K1[0]:=1/2*L*Sqrt(Qp[0]);  {Spulen-Energie}
  K2[0]:=1/2*C*Sqr(Q[0]/C);  {Kondensator-Energie}
  K3[0]:=1/2*J*Sqr(phip[0]); {Rotations-Energie}
  K4[0]:=K1[0]+K2[0]+K3[0];  {Gesamt-Energie}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-R/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]); {Mechanisches Drehmoment, x-Komponente}
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
    K0[i]:=0;  K1[i]:=0;  K2[i]:=0;  K3[i]:=0;  K4[i]:=0;  K5[i]:=0;
    K1[i]:=1/2*L*Sqr(Qp[i]);   {Spulen-Energie}
    K2[i]:=1/2*C*Sqr(Q[i]/C);  {Kondensator-Energie}
    K3[i]:=1/2*J*Sqr(phip[i]); {Rotations-Energie}
    K4[i]:=K1[i]+K2[i]+K3[i];  {Gesamt-Energie}
  end;
{ Gesamt-Energie-Bilanz und Anzeige:}
  EmA:=0; EmE:=0; siA:=0; siE:=0;
  For i:=1 to 10 do  EmA:=EmA+K4[i]/10;              {Mittelwert zu Beginn}
  For i:=AnzP-9 to AnzP do EmE:=EmE+K4[i]/10;        {Mittelwert am Ende}
  For i:=1 to 10 do siA:=siA+Sqr(EmA-K4[i]);         {Varianz zu Beginn}
  For i:=AnzP-9 to AnzP do siE:=siE+Sqr(EmE-K4[i]);  {Varianz am Ende}
  siA:=Sqrt(siA)/10;  siE:=Sqrt(siE)/10;             {Standardabweichungen}
  Writeln('Energie-Werte:  E_Anfang = (',EmA:11:7,' +/- ',siA:11:7,') Joule');
  Writeln('                E_Ende   = (',EmE:11:7,' +/- ',siE:11:7,') Joule');
  delE:=EmE-EmA;  sigdelE:=Sqrt(Sqr(siE)+Sqr(siA));
  Writeln('=> Veraenderung: delta_E = (',delE:11:7,' +/- ',sigdelE:11:7,') Joule');
  Writeln('=> Konvertierte Leistung = (',delE/(AnzP*dt):11:7,' +/- ',sigdelE/(AnzP*dt):11:7,') Watt');
  ExcelAusgabe('test_04.dat',11,Q,Qp,Qpp,phi,phip,phipp,K0,K1,K2,K3,K4,K0);

  Wait;   Wait;
End.
```

Var_L_022

```
Program Konverter_mit_variabler_Spule;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}
    K0,K1,K2,K3,K4,K5 : Feld;  {Kontroll-Felder für Plot-Zwecke}
    EmA,EmE,siA,siE : Double;  {Energie: Mittelwerte und Sigma "Anfang" und "Ende"}
    delE,sigdelE    : Double;  {Veränderung der Energie-Mittelwerte "Anfang" zu "Ende"}
    UmAn            : Double;  {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
    Eent            : Double;  {Entnommene Energie, elektrisch}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
```

```
  end;
  Close(fout);
end;

Function Rlast:Double;
begin
  Rlast:=21;
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=1000;     {Windungszahl der Spule}
  di:=0.09;    {Spulenkörper-Durchmesser}
  Dd:=0.0010;  {Drahtdurchmesser}
  Bo:=0.5000;  {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;    {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=1.641E-6; {Farad} {Kapazität des Kondensators}
  rm:=0.035;   {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;    {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;         {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  UmAn:=50;               {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.0001; {sec.}       {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R:8:2,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
  Writeln('Gesamtdauer der Betrachtung: ',AnzP*dt,' sec.');
{ Hier beginnt das Rechenprogramm.}

{ Teil_04: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;          Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;    phip[0]:=UmAn*2*pi;  phipp[0]:=0;          {Mechanische Startwerte}
  Eent:=0;                              {Reset für: Entnommene Energie, elektrisch}
  K0[0]:=0;
  K1[0]:=1/2*L*Sqrt(Qp[0]);  {Spulen-Energie}
  K2[0]:=1/2*C*Sqr(Q[0]/C);  {Kondensator-Energie}
  K3[0]:=1/2*J*Sqr(phip[0]); {Rotations-Energie}
  K4[0]:=K1[0]+K2[0]+K3[0];  {Gesamt-Energie}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]); {Mechanisches Drehmoment, x-Komponente}
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
    K0[i]:=0;
    K1[i]:=1/2*L*Sqr(Qp[i]);  {Spulen-Energie}
    K2[i]:=1/2*C*Sqr(Q[i]/C); {Kondensator-Energie}
    K3[i]:=1/2*J*Sqr(phip[i]);{Rotations-Energie}
    K4[i]:=K1[i]+K2[i]+K3[i]; {Gesamt-Energie}
    K5[i]:=Rlast*Sqr(Qp[i]);  {Am Lastwiderstand entnommene Leistung}
    Eent:=Eent+K5[i]*dt;      {Am Lastwiderstand entnommene Energie}
  end;
{ Gesamt-Energie-Bilanz und Anzeige:}
  EmA:=0; EmE:=0; siA:=0; siE:=0;
  For i:=1 to 10 do  EmA:=EmA+K4[i]/10;              {Mittelwert zu Beginn}
  For i:=AnzP-9 to AnzP do EmE:=EmE+K4[i]/10;        {Mittelwert am Ende}
  For i:=1 to 10 do siA:=siA+Sqr(EmA-K4[i]);         {Varianz zu Beginn}
  For i:=AnzP-9 to AnzP do siE:=siE+Sqr(EmE-K4[i]);  {Varianz am Ende}
  siA:=Sqrt(siA)/10;  siE:=Sqrt(siE)/10;             {Standardabweichungen}
  Writeln('Energie-Werte:  E_Anfang = (',EmA:11:7,' +/- ',siA:11:7,') Joule');
  Writeln('                E_Ende   = (',EmE:11:7,' +/- ',siE:11:7,') Joule');
  delE:=EmE-EmA;  sigdelE:=Sqrt(Sqr(siE)+Sqr(siA));
  Writeln('=> Veraenderung: delta_E = (',delE:11:7,' +/- ',sigdelE:11:7,') Joule');
```

```
  Writeln('=> Konvertierte Leistung = (',delE/(AnzP*dt):11:7,' +/- ',sigdelE/(AnzP*dt):11:7,') Watt');
  Writeln('An Rlast entnom.Leistung =  ',Eent/(AnzP*dt):11:7,' Watt');
  ExcelAusgabe('test_04.dat',12,Q,Qp,Qpp,phi,phip,phipp,K0,K1,K2,K3,K4,K5);

  Wait;   Wait;
End.
```

Var_L_023

```
Program Magnetkonverter_mit_realer_Leistungsentnahme;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}
    K0,K1,K2,K3,K4,K5 : Feld;  {Kontroll-Felder für Plot-Zwecke}
    EmA,EmE,siA,siE : Double;  {Energie: Mittelwerte und Sigma "Anfang" und "Ende"}
    delE,sigdelE    : Double;  {Veränderung der Energie-Mittelwerte "Anfang" zu "Ende"}
    UmAn            : Double;  {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
    Eent            : Double;  {Entnommene Energie, elektrisch}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
```

```
  end;
  Close(fout);
end;

Function Rlast:Double;
begin
  Rlast:=21;  {Ohm Lastwiderstand}
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=1120;     {Windungszahl der Spule}
  di:=0.09;    {Spulenkörper-Durchmesser}
  Dd:=0.0010;  {Drahtdurchmesser}
  Bo:=0.5000;  {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;    {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=1.650E-6; {Farad} {Kapazität des Kondensators}
  rm:=0.039;   {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;    {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;         {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  UmAn:=50;               {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.0001; {sec.}      {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R:8:2,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
  Writeln('Gesamtdauer der Betrachtung: ',AnzP*dt,' sec.');
{ Hier beginnt das Rechenprogramm.}

{ Teil_04: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;          Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;    phip[0]:=UmAn*2*pi;  phipp[0]:=0;          {Mechanische Startwerte}
  Eent:=0;                              {Reset für: Entnommene Energie, elektrisch}
  K0[0]:=0;
  K1[0]:=1/2*L*Sqrt(Qp[0]);  {Spulen-Energie}
  K2[0]:=1/2*C*Sqr(Q[0]/C);  {Kondensator-Energie}
  K3[0]:=1/2*J*Sqr(phip[0]); {Rotations-Energie}
  K4[0]:=K1[0]+K2[0]+K3[0];  {Gesamt-Energie}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]); {Mechanisches Drehmoment, x-Komponente}
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
    K0[i]:=0;
    K1[i]:=1/2*L*Sqr(Qp[i]);  {Spulen-Energie}
    K2[i]:=1/2*C*Sqr(Q[i]/C); {Kondensator-Energie}
    K3[i]:=1/2*J*Sqr(phip[i]);{Rotations-Energie}
    K4[i]:=K1[i]+K2[i]+K3[i]; {Gesamt-Energie}
    K5[i]:=Rlast*Sqr(Qp[i]);  {Am Lastwiderstand entnommene Leistung}
    Eent:=Eent+K5[i]*dt;      {Am Lastwiderstand entnommene Energie}
  end;
{ Gesamt-Energie-Bilanz und Anzeige:}
  EmA:=0; EmE:=0; siA:=0; siE:=0;
  For i:=1 to 10 do  EmA:=EmA+K4[i]/10;              {Mittelwert zu Beginn}
  For i:=AnzP-9 to AnzP do EmE:=EmE+K4[i]/10;        {Mittelwert am Ende}
  For i:=1 to 10 do siA:=siA+Sqr(EmA-K4[i]);         {Varianz zu Beginn}
  For i:=AnzP-9 to AnzP do siE:=siE+Sqr(EmE-K4[i]);  {Varianz am Ende}
  siA:=Sqrt(siA)/10;  siE:=Sqrt(siE)/10;             {Standardabweichungen}
  Writeln('Energie-Werte:  E_Anfang = (',EmA:11:7,' +/- ',siA:11:7,') Joule');
  Writeln('                E_Ende   = (',EmE:11:7,' +/- ',siE:11:7,') Joule');
  delE:=EmE-EmA;  sigdelE:=Sqrt(Sqr(siE)+Sqr(siA));
  Writeln('=> Veraenderung: delta_E = (',delE:11:7,' +/- ',sigdelE:11:7,') Joule');
```

```
  Writeln('=> Konvertierte Leistung = (',delE/(AnzP*dt):11:7,' +/- ',sigdelE/(AnzP*dt):11:7,') Watt');
  Writeln('An Rlast entnom.Leistung =  ',Eent/(AnzP*dt):11:7,' Watt');
  ExcelAusgabe('test_04.dat',12,Q,Qp,Qpp,phi,phip,phipp,K0,K1,K2,K3,K4,K5);

  Wait;   Wait;
End.
```

Var_L_024

```
Program Magnetkonverter_mit_realer_Leistungsentnahme;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}
    K0,K1,K2,K3,K4,K5 : Feld;  {Kontroll-Felder für Plot-Zwecke}
    EmA,EmE,siA,siE : Double;  {Energie: Mittelwerte und Sigma "Anfang" und "Ende"}
    delE,sigdelE    : Double;  {Veränderung der Energie-Mittelwerte "Anfang" zu "Ende"}
    UmAn            : Double;  {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
    Eent            : Double;  {Entnommene Energie, elektrisch}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
```

```
  end;
  Close(fout);
end;

Function Rlast:Double;
begin
  Rlast:=22;  {Ohm Lastwiderstand}
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=1050;     {Windungszahl der Spule}
  di:=0.09;    {Spulenkörper-Durchmesser}
  Dd:=0.0010;  {Drahtdurchmesser}
  Bo:=0.6800;  {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;    {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=0.93E-6; {Farad} {Kapazität des Kondensators}
  rm:=0.039;   {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;    {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;         {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  UmAn:=100;               {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.0001; {sec.}      {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R:8:2,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
  Writeln('Gesamtdauer der Betrachtung: ',AnzP*dt,' sec.');
{ Hier beginnt das Rechenprogramm.}

{ Teil_04: Mechanische Drehung}
  Writeln(' -> Teil_04: Mechanisch und elektrisch gekoppelt');
  UC:=0;{Volt}  Q[0]:=C*UC;          Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;    phip[0]:=UmAn*2*pi;  phipp[0]:=0;          {Mechanische Startwerte}
  Eent:=0;                              {Reset für: Entnommene Energie, elektrisch}
  K0[0]:=0;
  K1[0]:=1/2*L*Sqrt(Qp[0]);  {Spulen-Energie}
  K2[0]:=1/2*C*Sqr(Q[0]/C);  {Kondensator-Energie}
  K3[0]:=1/2*J*Sqr(phip[0]); {Rotations-Energie}
  K4[0]:=K1[0]+K2[0]+K3[0];  {Gesamt-Energie}
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]); {Mechanisches Drehmoment, x-Komponente}
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
    K0[i]:=0;
    K1[i]:=1/2*L*Sqr(Qp[i]);  {Spulen-Energie}
    K2[i]:=1/2*C*Sqr(Q[i]/C); {Kondensator-Energie}
    K3[i]:=1/2*J*Sqr(phip[i]);{Rotations-Energie}
    K4[i]:=K1[i]+K2[i]+K3[i]; {Gesamt-Energie}
    K5[i]:=Rlast*Sqr(Qp[i]);  {Am Lastwiderstand entnommene Leistung}
    Eent:=Eent+K5[i]*dt;      {Am Lastwiderstand entnommene Energie}
  end;
{ Gesamt-Energie-Bilanz und Anzeige:}
  EmA:=0; EmE:=0; siA:=0; siE:=0;
  For i:=1 to 10 do  EmA:=EmA+K4[i]/10;              {Mittelwert zu Beginn}
  For i:=AnzP-9 to AnzP do EmE:=EmE+K4[i]/10;        {Mittelwert am Ende}
  For i:=1 to 10 do siA:=siA+Sqr(EmA-K4[i]);         {Varianz zu Beginn}
  For i:=AnzP-9 to AnzP do siE:=siE+Sqr(EmE-K4[i]);  {Varianz am Ende}
  siA:=Sqrt(siA)/10;  siE:=Sqrt(siE)/10;             {Standardabweichungen}
  Writeln('Energie-Werte:  E_Anfang = (',EmA:11:7,' +/- ',siA:11:7,') Joule');
  Writeln('                E_Ende   = (',EmE:11:7,' +/- ',siE:11:7,') Joule');
  delE:=EmE-EmA;  sigdelE:=Sqrt(Sqr(siE)+Sqr(siA));
  Writeln('=> Veraenderung: delta_E = (',delE:11:7,' +/- ',sigdelE:11:7,') Joule');
```

```
  Writeln('=> Konvertierte Leistung = (',delE/(AnzP*dt):11:7,' +/- ',sigdelE/(AnzP*dt):11:7,') Watt');
  Writeln('An Rlast entnom.Leistung =  ',Eent/(AnzP*dt):11:7,' Watt');
  ExcelAusgabe('test_04.dat',12,Q,Qp,Qpp,phi,phip,phipp,K0,K1,K2,K3,K4,K5);

  Wait;   Wait;
End.
```

Var_L_025

```
Program Magnetkonverter_mit_realer_Leistungsentnahme;
{$APPTYPE CONSOLE}
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

Const AnzPmax=10000;   {Anzahl der Zeitschritte zur Lsg. der Dgl.}

Type Feld = Array[0..AnzPmax] of Double;

Var epo,muo    : Double;   {Naturkonstanten}
    lichtgesch : Double;   {Lichtgeschwindigkeit}
    n          : LongInt;  {Windungszahl der Spule}
    A          : Double;   {Querschnittsfläche der Spule}
    Bo         : Double;   {Magnetfeld (Amplitude) des Dauermagneten}
    ls         : Double;   {Länge der zylindrischen Spule}
    di         : Double;   {Durchmesser des Spulenkörpers}
    Dd         : Double;   {Drahtdurchmesser}
    rm         : Double;   {Radius des Dauermagneten}
    L          : Double;   {Induktivität der Spule}
    C          : Double;   {Kapazität des Kondensators}
    R          : Double;   {Ohm`scher Widerstand des Spulendrahtes}
    rho        : Double;   {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
    phi,phip,phipp : Feld; {Drehwinkel und dessen Ableitungen}
    Q,Qp,Qpp       : Feld; {Ladung und deren Ableitungen}
    i          : LongInt;  {Laufvariable}
    AnzP       : LongInt;  {Anzahl der tatsächlich berechneten Zeit-Schritte}
    dt         : Double;   {Dauer der Zeitschritte zur Lsg. der Dgl.}
    Abstd      : Integer;  {Jeder wievielte Punkte soll geplottet werden}
    omo        : Double;   {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
    T          : Double;   {Schwingungsdauer des elektrischen Schwingkreises}
    UC,UL      : Double;   {Kondensatorspannung und Spulenspannung}
    rhom       : Double;   {Dichte des Magnetmaterials}
    dm         : Double;   {Dicke des zylindrischen Dauermagneten}
    mt         : Double;   {Träge Masse des zylindrischen Dauermagneten}
    J          : Double;   {Trägheitsmoment des zylindrischen Dauermagneten}
    K0,K1,K2,K3,K4,K5 : Feld;  {Kontroll-Felder für Plot-Zwecke}
    EmA,EmE,siA,siE : Double;  {Energie: Mittelwerte und Sigma "Anfang" und "Ende"}
    delE,sigdelE    : Double;  {Veränderung der Energie-Mittelwerte "Anfang" zu "Ende"}
    UmAn            : Double;  {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
    Eent            : Double;  {Entnommene Energie, elektrisch}
    Rlast           : Double;  {Ohm'scher Lastwiderstand}

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer;KA,KB,KC,KD,KE,KF,KG,KH,KI,KJ,KK,KL:Feld);
Var fout   : Text;    {Bis zu 12 Spalten zum Aufschreiben der Ergebnisse}
    lv,j,k : Integer; {Laufvariable}
    Zahl   : String;  {Die ins Excel zu druckenden Zahlen}
begin
  Assign(fout,Name); Rewrite(fout); {File öffnen}
  For lv:=0 to AnzP do  {von "plotanf" bis "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {Kolumnen drucken}
        If j=1  then Str(KA[lv]:19:14,Zahl);
        If j=2  then Str(KB[lv]:19:14,Zahl);
        If j=3  then Str(KC[lv]:19:14,Zahl);
        If j=4  then Str(KD[lv]:19:14,Zahl);
        If j=5  then Str(KE[lv]:19:14,Zahl);
        If j=6  then Str(KF[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {Keine Dezimalpunkte verwenden, sondern Kommata}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Daten-Trennung, Tabulator}
      end;
      Writeln(fout,'');    {Zeilen-Trennung}
    end;
  end;
```

```
  Close(fout);
end;

Begin {Hauptprogramm}
{ Initialisierung - Vorgabe der Werte: }        {Wir arbeiten in SI-Einheiten}
  Writeln('Raumenergie-Konverter mit Rotation.');
{ Vorgabe der Werte -> Input-Parameter:}
  epo:=8.854187817E-12{As/Vm}; {Magnetische Feldkonstante}
  muo:=4*pi*1E-7{Vs/Am};       {Elektrische Feldkonstante}
  lichtgesch:=Sqrt(1/muo/epo){m/s};   Writeln('Lichtgeschwindigkeit c = ',lichtgesch, ' m/s');
{ Spule, Magnet, Kondensator:}
  n:=1600;     {Windungszahl der Spule}
  di:=0.09;    {Spulenkörper-Durchmesser}
  Dd:=0.0010;  {Drahtdurchmesser}
  Bo:=0.700;   {Tesla} {Magnetfeld (Amplitude) des Dauermagneten}
  ls:=0.01;    {Meter} {Länge des zylindrischen Spulenkörpers}
  C:=0.23E-6;  {Farad} {Kapazität des Kondensators}
  rm:=0.039;   {Meter} {Radius des zylindrischen Dauermagneten}
  dm:=0.01;    {Meter} {Dicke des zylindrischen Dauermagneten}
  rhom:=7.8E3;         {Dichte des Magnet-Materials, Eisen, Kohlrausch Bd.3}
{ Abgeleitete Parameter, keine Eingabe möglich:}
  A:=di*di; {Meter * Meter} {Querschnittsfläche der Spule}
  L:=muo*a*n*n/ls;  {Induktivität der Spule}
  omo:=1/Sqrt(L*C); {Kreis-Eigenfrequenz des elektrischen Schwingkreises}
  T:=2*pi/omo;      {Schwingungsdauer des elektrischen Schwingkreises}
  rho:=1.7E-8; {Ohm*m}  {Spez. Widerstand von Kupfer, je nach Temperatur, Kohlrausch,T193}
  R:=rho*(2*pi*di*n)/(pi*(Dd/2)*(Dd/2)); {Ohm} {Ohm`scher Widerstand des Spulendrahtes}
{ Sonstige:}
  UmAn:=100;              {Startwert: Umdrehungen pro Sekunde bei Anlaufen des Rotors}
  Rlast:=28;              {Ohm'scher Lastwiderstand}
  AnzP:=AnzPmax;          {Anzahl der Zeitschritte insgesamt}
  dt:=0.0001; {sec.}      {Größe der Zeitschritte}
  Abstd:=1;               {Jeder wievielte Punkte soll geplottet werden}
  mt:=pi*rm*rm*dm*rhom;   {Träge Masse des zylindrischen Dauermagneten}
  J:=1/2*mt*rm*rm;        {Trägheitsmoment des zylindrischen Dauermagneten}
{ Anzeige der Werte:}
  Writeln('Induktivitaet der Luft-Spule: L = ',L,' Henry');
  Writeln('Eigen-Kreisfreq harmon.el.Osz.: omo = ',omo:8:4,' Hz => T = ',T:15,'sec.');
  Writeln('Laenge des Spulendrahts: ',(2*pi*di*n),' m');
  Writeln('Ohm`scher Widerstand des Spulendrahts: R = ',R:8:2,' Ohm');
  Writeln('Traege Masse des zylindrischen Dauermagneten: mt = ',mt,' kg');
  Writeln('Traegheitsmoment des Dauermagneten: J = ',J,' kg*m^2');
  Writeln('Gesamtdauer der Betrachtung: ',AnzP*dt,' sec.');

{ Hier beginnt das Rechenprogramm.}
  Writeln('Mechanisch und elektrisch gekoppelte Schwingung.');
  UC:=0;{Volt}  Q[0]:=C*UC;          Qpp[0]:=0;  Qp[0]:=0; {Elektrische Startwerte}
  phi[0]:=0;    phip[0]:=UmAn*2*pi;  phipp[0]:=0;          {Mechanische Startwerte}
  Eent:=0;                              {Reset für: Entnommene Energie, elektrisch}
  K0[0]:=0;
  K1[0]:=1/2*L*Sqrt(Qp[0]);  {Spulen-Energie}
  K2[0]:=1/2*C*Sqr(Q[0]/C);  {Kondensator-Energie}
  K3[0]:=1/2*J*Sqr(phip[0]); {Rotations-Energie}
  K4[0]:=K1[0]+K2[0]+K3[0];  {Gesamt-Energie}
  K5[0]:=0;
  For i:=1 to AnzP do
  begin
    Qpp[i]:=-1/L/C*Q[i-1]-(R+Rlast)/2/L*Qp[i-1];      {nach *5 von S.6}
    Qpp[i]:=Qpp[i]+n*Bo*A*sin(phi[i-1])*phip[i-1]/L; {Induzierte Spannung in Spule bringen.}
    Qp[i]:=Qp[i-1]+(Qpp[i]-R/2/L*Qp[i-1])*dt; {nach *3 & *4 von S.6}
    Q[i]:=Q[i-1]+Qp[i]*dt;
    phipp[i]:=-Bo*n*Qp[i]*A/J*sin(phi[i-1]); {Mechanisches Drehmoment, x-Komponente}
    phip[i]:=phip[i-1]+phipp[i]*dt;
    phi[i]:=phi[i-1]+phip[i]*dt;
    K0[i]:=0;
    K1[i]:=1/2*L*Sqr(Qp[i]);  {Spulen-Energie}
    K2[i]:=1/2*C*Sqr(Q[i]/C); {Kondensator-Energie}
    K3[i]:=1/2*J*Sqr(phip[i]);{Rotations-Energie}
    K4[i]:=K1[i]+K2[i]+K3[i]; {Gesamt-Energie}
    K5[i]:=Rlast*Sqr(Qp[i]);  {Am Lastwiderstand entnommene Leistung}
    Eent:=Eent+K5[i]*dt;      {Am Lastwiderstand entnommene Energie}
  end;
{ Gesamt-Energie-Bilanz und Anzeige:}
  EmA:=0; EmE:=0; siA:=0; siE:=0;
  For i:=1 to 10 do  EmA:=EmA+K4[i]/10;              {Mittelwert zu Beginn}
  For i:=AnzP-9 to AnzP do EmE:=EmE+K4[i]/10;        {Mittelwert am Ende}
  For i:=1 to 10 do siA:=siA+Sqr(EmA-K4[i]);         {Varianz zu Beginn}
  For i:=AnzP-9 to AnzP do siE:=siE+Sqr(EmE-K4[i]);  {Varianz am Ende}
  siA:=Sqrt(siA)/10;  siE:=Sqrt(siE)/10;             {Standardabweichungen}
  Writeln('Energie-Werte:  E_Anfang = (',EmA:11:7,' +/- ',siA:11:7,') Joule');
  Writeln('                E_Ende   = (',EmE:11:7,' +/- ',siE:11:7,') Joule');
  delE:=EmE-EmA;  sigdelE:=Sqrt(Sqr(siE)+Sqr(siA));
  Writeln('=> Veraenderung: delta_E = (',delE:11:7,' +/- ',sigdelE:11:7,') Joule');
  Writeln('=> Konvertierte Leistung = (',delE/(AnzP*dt):11:7,' +/- ',sigdelE/(AnzP*dt):11:7,') Watt');
  Writeln('An Rlast entnom.Leistung =  ',Eent/(AnzP*dt):11:7,' Watt');
  ExcelAusgabe('test_04.dat',12,Q,Qp,Qpp,phi,phip,phipp,K0,K1,K2,K3,K4,K5);
  Wait;   Wait;
End.
```