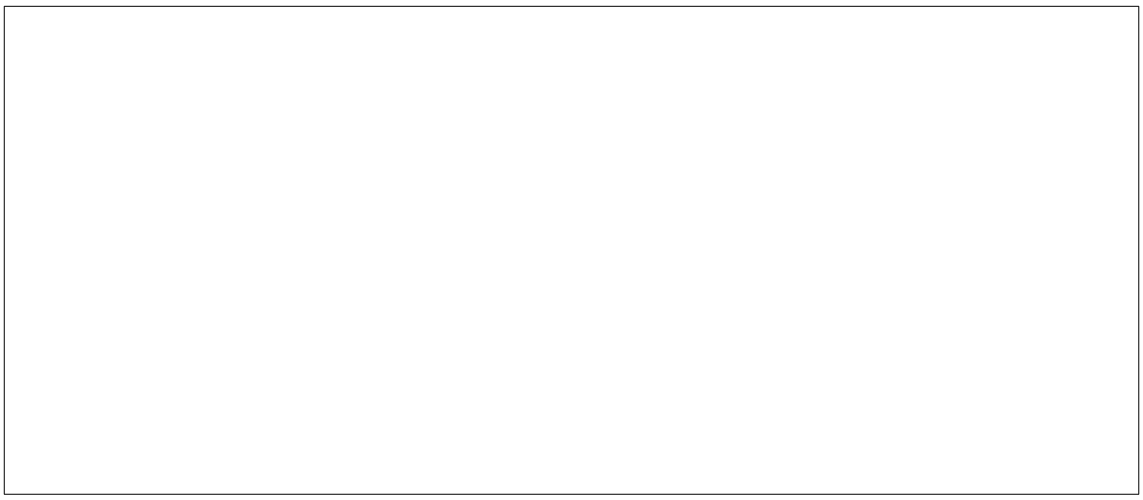
Das Doppelpendel

Im Lagrange-Formalismus mit Scilab



Im April 2025

Bernd Ragutt

Inhaltsverzeichnis

1 Das mathematische Doppelpendel

1.1 Die Newtonschen Bewegungsgleichungen

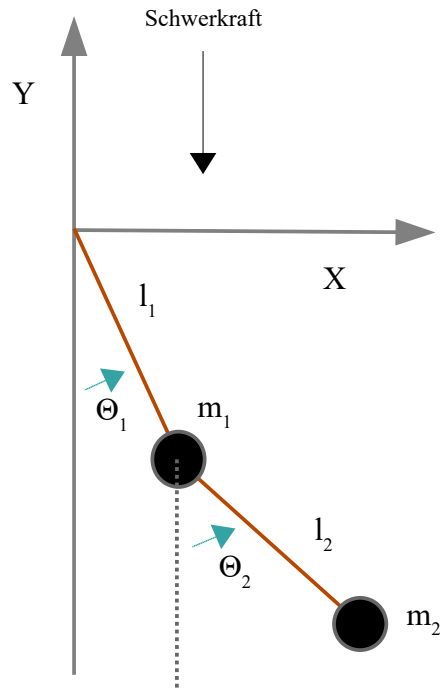


Abbildung 1: Bestimmungsgrößen für das Doppelpendel

Die Newton'schen Bewegungsgleichungen sind:	$m_1\ \ddot{x}_1 = 0$ $m_1\ \ddot{y}_1 = -m_1 g$ $m_2\ \ddot{x}_2 = 0$ $m_2\ \ddot{y}_2 = -m_2 g$
Die kartesischen Koordinaten erfüllen die zwei Zwangsbedingungen:	$x_1^2 + x_2^2 = l_1^2$ $(x_2 - x_1)^2 + (y_2 - y_1)^2 = l_2^2$
Es wirkt hier nur die Schwerkraft, die Corioliskraft, die durch die Erdrotation entsteht, und die Reibungskräfte, die durch die Luft entstehen, werden vernachlässigt. Das Problem wird als ebenes Problem behandelt – obwohl es durchaus spannend wäre, die Bewegungsvielfalt in drei Dimensionen zu betrachten.	

1.2 Die Lagrangeschen Bewegungsgleichungen

Als generalisierte Koordinaten werden die Auslenkwinkel θ_1 und θ_2 und deren Differenz $\Delta\theta = \theta_1 - \theta_2$ verwendet.	$x_1 = l_1 \sin\theta_1$ $y_1 = -l_1 \cos\theta_1$ $x_2 = x_1 + l_2 \sin\theta_2$ $y_2 = y_1 - l_2 \cos\theta_2$ $\Delta\theta = \theta_1 - \theta_2$
Die kinetische Energie ergibt sich wie folgt aus den Geschwindigkeiten:	$v_1^2 = \dot{x}_1^2 + \dot{y}_1^2$ $v_2^2 = \dot{x}_2^2 + \dot{y}_2^2$ $T = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2$
Die potentielle Energie ist:	$V = m_1 g\ y_1 + m_2 g\ y_2$
$\dot{x}_1 = l_1 \cos\theta_1\ \dot{\theta}_1$ $\dot{y}_1 = l_1 \sin\theta_1\ \dot{\theta}_1$	$\dot{x}_2 = \dot{x}_1 + l_2 \cos\theta_2\ \dot{\theta}_2$ $\dot{y}_2 = \dot{y}_1 + l_2 \sin\theta_2\ \dot{\theta}_2$
$T = \frac{1}{2}(m_1+m_2) l_1^2\ \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2\ \dot{\theta}_2^2 + m_2 l_1 l_2\ \dot{\theta}_1 \dot{\theta}_2 \cos\Delta\theta$	
$V = -(m_1+m_2) g\, l_1 \cos\theta_1 - m_2 g\, l_2 \cos\theta_2$	
$\mu = \frac{m_2}{m_1+m_2}$ $\lambda = \frac{l_2}{l_1}$ $\omega = \frac{g}{l_1}$	$m_2 \neq 0$ $l_1 \neq 0$
$T = \gamma_T \left(\dot{\theta}_1^2 + \mu\lambda^2\ \dot{\theta}_2^2 + 2\mu\lambda\ \dot{\theta}_1 \dot{\theta}_2 \cos\Delta\theta\right)$	$\gamma_T = \frac{1}{2}(m_1+m_2) l_1^2$
$V = \gamma_V \left(\cos\theta_1 + \mu\lambda\ \cos\theta_2\right)$	$\gamma_V = -(m_1+m_2) l_1\ g$
Die beiden Bewegungsgleichungen im Lagrange-Formalismus ergeben sich dann aus der Langrange-Funktion L und den Lagrange'schen Gleichungen.	$L \stackrel{\text{def}}{=} T - V$ $\frac{\partial L}{\partial \theta_1} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_1} = 0$ $\frac{\partial L}{\partial \theta_2} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_2} = 0$
$(m_1+m_2) l_1\ \ddot{\theta}_1 + m_2 l_2\ \ddot{\theta}_2 \cos\Delta\theta + m_2 l_2\ \dot{\theta}_2^2 \sin\Delta\theta + (m_1+m_2) g\ \sin\theta_1 = 0$	
$m_2 l_1\ \ddot{\theta}_1 \cos\Delta\theta + m_2 l_2\ \ddot{\theta}_2 - m_2 l_1\ \dot{\theta}_1^2 \sin\Delta\theta + m_2 g\ \sin\theta_2 = 0$	
$\ddot{\theta}_1 + \mu\lambda \cos\Delta\theta\ \ddot{\theta}_2 + \mu\lambda \sin\Delta\theta\ \dot{\theta}_2^2 + \omega \sin\theta_1 = 0$ $\cos\Delta\theta\ \ddot{\theta}_1 + \lambda\ \ddot{\theta}_2 - \sin\Delta\theta\ \dot{\theta}_1^2 + \omega \sin\theta_2 = 0$	

$p_1 \stackrel{\text{def}}{=} \dot{\theta}_1$ $\dot{p}_1 = \ddot{\theta}_1$	$p_2 \stackrel{\text{def}}{=} \dot{\theta}_2$ $\dot{p}_2 = \ddot{\theta}_2$
$\dot{p}_1 + \mu\lambda \cos\Delta\theta\ \dot{p}_2 + \mu\lambda \sin\Delta\theta\ p_2^2 + \omega \sin\theta_1 = 0$ $\cos\Delta\theta\ \dot{p}_1 + \lambda\ \dot{p}_2 - \sin\Delta\theta\ p_1^2 + \omega \sin\theta_2 = 0$	
$\begin{pmatrix} 1 & \mu\lambda \cos\Delta\theta \\ \cos\Delta\theta & \lambda \end{pmatrix} \begin{pmatrix} \dot{p}_1 \\ \dot{p}_2 \end{pmatrix} = \begin{pmatrix} -\mu\lambda \sin\Delta\theta\ p_2^2 - \omega \sin\theta_1 \\ \sin\Delta\theta\ p_1^2 - \omega \sin\theta_2 \end{pmatrix}$	
$\boldsymbol{p} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$ $\dot{\boldsymbol{p}} = \begin{pmatrix} \dot{p}_1 \\ \dot{p}_2 \end{pmatrix}$ $\boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$	$\boldsymbol{\theta} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$ $\dot{\boldsymbol{\theta}} = \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix}$

$\boldsymbol{A} = \boldsymbol{A}(\theta) \qquad \boldsymbol{b} = \boldsymbol{b}(\theta, \boldsymbol{p})$	$\boldsymbol{A}\,\dot{\boldsymbol{p}} - \boldsymbol{b} = 0 \qquad \dot{\boldsymbol{p}} = \boldsymbol{A}^{-1}\boldsymbol{b}$
Die Lagrangeschen Bewegungsgleichungen sind ausgeschrieben als Differentialgleichungen 1. Ordnung in Matrixform:	$\dot{\theta} = \boldsymbol{p}$ $\dot{\boldsymbol{p}} = \boldsymbol{A}^{-1}(\theta)\,\boldsymbol{b}(\theta, \boldsymbol{p})$
$\boldsymbol{A} = \begin{pmatrix} 1 & \mu\lambda\cos\Delta\theta \\ \cos\Delta\theta & \lambda \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$	$b_1 = -\mu\lambda\sin\Delta\theta\ p_2^2 - \omega\sin\theta_1$ $b_2 = \sin\Delta\theta\ p_1^2 - \omega\sin\theta_2$
Determinante von $\boldsymbol{A}$ $\lvert\boldsymbol{A}\rvert = \lambda\,(1 - \mu\cos^2\Delta\theta)$	*Die inverse Matrix von* $\boldsymbol{A}$ $\boldsymbol{A}^{-1} = \frac{1}{\lvert\boldsymbol{A}\rvert}\begin{pmatrix} \lambda & -\mu\lambda\cos\Delta\theta \\ -\cos\Delta\theta & 1 \end{pmatrix}$
In dem wisschenschaftlichen Werkzeugkasten Scilab kann man sich auch den Umweg über die inverse Matrix $\mathbf{A}^{-1}$ sparen - wenn man sich die zusätzlichen Rechenzeit leisten will und kann :-)	*Matrixgleichung in Scilab* $\boldsymbol{A}\,\dot{\boldsymbol{p}} = \boldsymbol{b} \qquad \lvert\boldsymbol{A}\rvert \neq 0$ *Lösungsvektor der Matrixgleichung in Scilab* $\dot{\boldsymbol{p}} = \boldsymbol{A} \setminus \boldsymbol{b}$

Anhang

A. Der Scilab-Code

Doppelpendel-Lagrange.sce

```
0001  // ###########################################################################
0002
0003  // Pendelparameter
0004  // Längen (l1,l2) und Massem (m1,m2) der Pendel
0005  // Länge in Meter [m] , Masse in Kilogramm [kg]
0006
0007  // Oberes Pendel 1
0008  l1=1.0; m1=1.0;
0009  // Unteres Pendel 2
0010  l2=1.0; m2=1.0;
0011
0012  // ###########################################################################
0013  // Anfangsbedingungen für die Differentialgleichnungen
0014
0015  // Auslenkungen (wi_0, i=1,2) und Winkelgeschwindigkeiten (widot_0, i=1,2) der beiden Massen
0016  // Auslenkungen in Bogengrad [°]
0017  // Winkelgeschwindigkeiten in Bogengrad pro Zeiteinheit [°/s]
0018
0019  // Auslenkungen
0020  w1_0=90.0; w2_0=90.0;
0021
0022  // Winkelgeschwindigkeiten
0023  w1dot_0=0.0; w2dot_0=0.0;
0024
0025  // Zeitliche Simulationsdauer
0026  tEnd=11; // Dauer in Zeiteinheiten
0027
0028  // ###########################################################################
0029  funcprot(0);
0030  // ###########################################################################
0031  // Import sci-Module
0032
0033  pathdir = get_absolute_file_path('Doppelpendel-Lagrange.sce');
0034  exec(pathdir + 'Module/dpLagrange-DGLAlgorithmen.sci');
0035  exec(pathdir + 'Module/dpLagrange-DGLSystem.sci');
0036  exec(pathdir + 'Module/dpLagrange-EnergieAlgorithmen.sci');
0037  exec(pathdir + 'Module/dpLagrange-OrtsAlgorithmen.sci');
0038  exec(pathdir + 'Module/dpLagrange-OrtsSimulation.sci');
0039  exec(pathdir + 'Module/dpLagrange-EnergieAusgaben.sci');
0040
0041  // ###########################################################################
0042
0043  // Umrechnungsfunktion Bogengrad nach Radian
0044  function y=fd2r(x),y=x*%pi/180,endfunction;
0045      // Umrechnungsfunktion Radian nach Bogengrad
0046  function y=fr2d(x),y=x*180.0/%pi,endfunction;
0047
0048  // ###########################################################################
0049  // Zeitliche Simulationsdaten
0050
0051  tBegin=0; t0=tBegin;
0052  stepTime=1/50;  // Bilder pro Zeiteinheit
0053  tSteps=(tEnd-tBegin)/stepTime;
0054  tData=linspace(tBegin,tEnd,tSteps);
0055
0056  // Anfangsbedingungen nach Radian konvertieren und als Liste packen
0057  w1_0=fd2r(w1_0); w2_0=fd2r(w2_0);
0058  w1dot_0=fd2r(w1dot_0); w2dot_0=fd2r(w2dot_0);
0059  y0=[w1_0;w2_0;w1dot_0;w2dot_0];
0060
0061  // ###########################################################################
0062  // ###########################################################################
0063
0064  // Lösen der lagrangeschen Bewgungsgleichungen
0065  yData=dpLagrange_Solver(y0,t0,tData);
0066
0067  // ###########################################################################
0068
0069  // Ausgaben der Energie-Daten
0070  if %T then fdpLagrange_Ausgabe(tData,yData) end;
0071
0072  // Simulation des Doppelpendels im Ortsraum
0073  if %F then fdpLagrange_Simulation(yData) end;
0074
0075  // ###########################################################################
0076  disp('Beendet: Doppelpendel-Lagrange.sce');
```

dpLagrange-DGLAlgorithmen.sci

```
0001  // Algorithmen für das Doppelpendel im Lagrange-Formalismus
0002
0003  // ###########################################################################
0004  // Gravitationskonstante
0005
0006  gc=9.81; // [m/s^2]
0007
0008  // ###########################################################################
0009  // Abgeleitete Konstanten
0010
0011  mue=m2/(m1+m2);
0012  lbd=l2/l1;
0013  omg=gc/l1;
0014  gmt=0.5*(m1+m2)*l1^2;
0015  gmv=-(m1+m2)*l1*gc;
0016
0017  // ###########################################################################
0018  // ###########################################################################
0019
0020  function rwdot=fwdot(w1, w2, p1, p2)
0021      rwdot=[p1,p2];
0022  endfunction;
0023
0024  // ###########################################################################
0025
0026  function rA=fA(w1, w2, p1, p2)
0027      rA(1,1)=1.0;
0028      rA(1,2)=mue*lbd*cos(w1-w2);
0029      rA(2,1)=cos(w1-w2);
0030      rA(2,2)=lbd;
0031  endfunction;
0032
0033  function rb1=fb1(w1, w2, p1, p2)
0034      rb1=-mue*lbd*sin(w1-w2)*p2^2-omg*sin(w1);
0035  endfunction;
0036
0037  function rb2=fb2(w1, w2, p1, p2)
0038      rb2=sin(w1-w2)*p1^2-omg*sin(w2);
0039  endfunction;
0040
0041  function rb=fb(w1, w2, p1, p2)
0042      rb(1)=fb1(w1,w2,p1,p2);
0043      rb(2)=fb2(w1,w2,p1,p2);
0044  endfunction;
0045
0046  function rpdot=fpdot(w1, w2, p1, p2)
0047      A=fA(w1,w2,p1,p2);
0048      b=fb(w1,w2,p1,p2);
0049      rpdot=A\ b;
0050  endfunction;
0051
0052  // ###########################################################################
0053  // ###########################################################################
0054  disp('Geladen: dpLagrange-DGLAlgorithmen.sci');
```

dpLagrange-DatenAufbereitung.sci

```
0001  // ###########################################################################
0002  // Berechnung der Energien aus den Anfangsbedingungen über kartesische
0003  // Koordinaten - als Vergleichswerte genutzt.
0004
0005  // Kinetische Energie
0006  function rekin_kart=fekin_kart(w1, w2, w1d, w2d)
0007      x1d=l1*cos(w1)*w1d;
0008      y1d=l1*sin(w1)*w1d;
0009      v1q=x1d^2+y1d^2;
0010      x2d=x1d+l2*cos(w2)*w2d;
0011      y2d=y1d+l2*sin(w2)*w2d;
0012      v2q=x2d^2+y2d^2;
0013      rekin_kart=0.5*(m1*v1q+m2*v2q);
0014  endfunction;
0015
0016  // Potentielle Energie
0017  function repot_kart=fepot_kart(w1, w2)
0018      y1=-l1*cos(w1);
0019      y2=y1-l2*cos(w2);
0020      repot_kart=m1*gc*y1+m2*gc*y2;
0021  endfunction;
0022
0023  // Totale Energie
0024  function retot_kart=fetot_kart(w1, w2, w1d, w2d)
0025      retot_kart=fekin_kart(w1,w2,w1d,w2d)+fepot_kart(w1,w2);
0026  endfunction;
0027
0028  // ###########################################################################
0029  // Berechnung der Energien aus den Winkelkkoordinaten
0030
0031  // Funktion für die kinetische Energie
0032  function rekin=fekin(w1, w2, w1d, w2d)
0033      rekin=gmt*(w1d^2+mue*lbd^2*w2d^2+2.0*mue*lbd*w1d*w2d*cos(w1-w2));
0034  endfunction;
0035
0036  // Funktion für die potentielle Energie
0037  function repot=fepot(w1, w2)
0038      repot=gmv*(cos(w1)+mue*lbd*cos(w2));
0039  endfunction;
0040
0041  // Funktion für die totale Energie
0042  function retot=fetot(w1, w2, w1d, w2d)
0043      retot=fekin(w1,w2,w1d,w2d)+fepot(w1,w2);
0044  endfunction;
0045
0046  // ###########################################################################
0047
0048  function rekindata=fekindata(ydata)
0049      for i=1:size(ydata,"c")
0050          w1=ydata(1,i);
0051          w2=ydata(2,i);
0052          p1=ydata(3,i);
0053          p2=ydata(4,i);
0054          rekindata(i)=fekin(w1,w2,p1,p2);
0055      end
0056  endfunction;
0057
0058  function repotdata=fepotdata(ydata)
0059      for i=1:size(ydata,"c")
0060          w1=ydata(1,i);
0061          w2=ydata(2,i);
0062          repotdata(i)=fepot(w1,w2);
0063      end
0064  endfunction;
0065
0066  function retot=fetot(w1, w2, p1, p2)
0067      retot=fekin(w1,w2,p1,p2)+fepot(w1,w2)
0068  endfunction
0069
0070  function retotdata=fetotdata(ydata)
0071      for i=1:size(ydata,"c")
0072          w1=ydata(1,i);
0073          w2=ydata(2,i);
0074          p1=ydata(3,i);
0075          p2=ydata(4,i);
0076          retotdata(i)=fetot(w1,w2,p1,p2);
0077      end
0078  endfunction;
0079
0080  // ###########################################################################
0081  disp('Geladen: dpLagrange-DatenAufbereitung.sci');
```

dpLagrange-OrtsAlgorithmen.sci

```
0001  // ###########################################################################
0002  // Berechnung der kartesischen Koordinaten der beiden Pendelmassen aus den
0003  // berechneten Winkeldaten
0004
0005  function rposdata=fposdata(yData)
0006      for i=1:size(yData,"c")
0007          w1=yData(1,i);
0008          w2=yData(2,i);
0009          x1=l1*sin(w1);
0010          y1=(-1)*l1*cos(w1);
0011          x2=x1+l2*sin(w2);
0012          y2=y1-l2*cos(w2);
0013          rposdata(1,i)=x1;
0014          rposdata(2,i)=y1;
0015          rposdata(3,i)=x2;
0016          rposdata(4,i)=y2;
0017      end;
0018  endfunction;
0019
0020  // ###########################################################################
0021  // ###########################################################################
0022  disp('Geladen: dpLagrange-OrtsAlgorithmen.sci');
```

dpLagrange-DGLSystem.sci

```
0001  // ###########################################################################
0002
0003  function yData=dpLagrange_Solver(y0, t0, tD)
0004
0005      // Lagrangesches Differentialgleichungssystem
0006      function ydot=fdpLagrange(t, y)
0007          w1=y(1); w2=y(2);
0008          p1=y(3); p2=y(4);
0009          wd=fwdot(w1,w2,p1,p2);
0010          pd=fpdot(w1,w2,p1,p2);
0011          ydot=wd(1)*[1;0;0;0] ..
0012              +wd(2)*[0;1;0;0] ..
0013              +pd(1)*[0;0;1;0] ..
0014              +pd(2)*[0;0;0;1];
0015      endfunction;
0016
0017      // Lösung der Differntialgleichungen
0018      // Scilab-Solver-Methode:
0019      // "lsoda solver of package ODEPACK is called by default.
0020      // It automatically selects between nonstiff predictor-corrector Adams
0021      // method and stiff Backward Differentiation Formula (BDF) method.
0022      // It uses nonstiff method initially and dynamically monitors data in
0023      // order to decide which method to use."
0024      fdp=fdpLagrange;
0025      yData=ode(y0,t0,tData,fdp);
0026
0027      // // Scilab-Solver-Methode: Adaptive Runge-Kutta of order 4 (RK4)
0028      // yData=ode("rk",y0,t0,tData,fdp);
0029  endfunction;
0030
0031  // ###########################################################################
0032  disp('Geladen: dpLagrange-DGLSystem.sci');
```

dpLagrange-EnergieAusgaben.sci

```
0001  // ###########################################################################
0002  // Ausgabe der Daten
0003
0004  // ###########################################################################
0005
0006  function fdpLagrange_Ausgabe(tData, yData)
0007
0008      etotdata=fetotdata(yData);
0009      ekindata=fekindata(yData);
0010      epotdata=fepotdata(yData);
0011
0012      etot0=etotdata(1); // Gesamtenergie, Konstante der Bewegung
0013      px_etot=[tBegin;tEnd];
0014
0015      delta_etot=etotdata-etot0;
0016      delta_etot0=delta_etot(1);
0017
0018      // ###########################################################################
0019
0020      my_handle1=scf(1);
0021      clf(my_handle1,"reset");
0022
0023      a=gca();
0024      a.x_label.text = "Zeit [s]";
0025      a.y_label.text = "Energien [Joule]";
0026
0027      xgrid(color("lightgrey"));
0028      xtitle('Doppelpendel - Energieverhalten über die Zeit');
0029
0030      plot(tData,etotdata,'red');
0031      plot(tData,ekindata,'blue');
0032      plot(tData,epotdata,'green');
0033
0034      py_etot=[etot0;etot0];
0035      e=xpoly(px_etot,py_etot);
0036      e.foreground=color('scilabmagenta3');
0037
0038      legend(['Etot(t)';'Ekin(t)';'Epot(t)';'Etot'],"lower_caption",%f);
0039
0040      // ###########################################################################
0041
0042      my_handle2=scf(2);
0043      clf(my_handle2,"reset");
0044      a=gca();
0045
0046      a.x_label.text = "Zeit [s]";
0047      a.y_label.text = "Abweichungen vom physikalischen Wert Etot  [Joule]";
0048
0049      xgrid(color("lightgrey"));
0050      xtitle('Doppelpendel - Energieverhalten über die Zeit');
0051
0052      plot(tData,delta_etot,'red');
0053
0054      py_etot=[delta_etot0;delta_etot0];
0055      e=xpoly(px_etot,py_etot);
0056      e.foreground=color('scilabmagenta3');
0057
0058      legend(['Etot(t) - Etot';'Etot'],"lower_caption",%f);
0059
0060      // ###########################################################################
0061
0062      mean_etotdata=mean(etotdata);
0063      mean_ekindata=mean(ekindata);
0064      mean_epotdata=mean(epotdata);
0065
0066      max_etotdata=max(etotdata);
0067      max_ekindata=max(ekindata);
0068      max_epotdata=max(epotdata);
0069
0070      min_etotdata=min(etotdata);
0071      min_ekindata=min(ekindata);
0072      min_epotdata=min(epotdata);
0073
0074      init_etot=etotdata(1);
0075      init_ekin=ekindata(1);
0076      init_epot=epotdata(1);
0077
0078      // Zur Kontrolle der Algorithmen
0079      kart_ekin=fekin_kart(w1_0,w2_0,w1dot_0,w2dot_0);
0080      kart_epot=fepot_kart(w1_0,w2_0);
0081      kart_etot=fetot_kart(w1_0,w2_0,w1dot_0,w2dot_0);
0082
0083      disp('   ');
0084      disp('---');
0085      disp('kart_ekin    : '+ string(kart_ekin));
0086      disp('init_ekin    : '+ string(init_ekin));
0087      disp('mean_ekindata: '+ string(mean_ekindata));
0088      disp('max_ekindata : '+ string(max_ekindata));
0089      disp('min_ekindata : '+ string(min_ekindata));
0090      disp('---');
0091      disp('kart_epot    : '+ string(kart_epot));
0092      disp('init_epot    : '+ string(init_epot));
```

```
0093      disp('mean_epotdata: '+ string(mean_epotdata));
0094      disp('max_epotdata : '+ string(max_epotdata));
0095      disp('min_epotdata : '+ string(min_epotdata));
0096      disp('---');
0097      disp('kart_etot    : '+ string(kart_etot));
0098      disp('init_etot    : '+ string(init_etot));
0099      disp('mean_etotdata: '+ string(mean_etotdata));
0100      disp('max_etotdata : '+ string(max_etotdata));
0101      disp('min_etotdata : '+ string(min_etotdata));
0102      disp('---');
0103
0104     // disp((etotdata-etot1)/etot1);
0105
0106  endfunction;
0107
0108  // ##############################################################################
0109  disp('Geladen: dpLagrange-EnergieAusgaben.sci');
```

dpLagrange-OrtsSimulation.sci

```
0001  // ###########################################################################
0002  // Simulieren der Orts-Daten
0003
0004  // ###########################################################################
0005  // Zeichnerische Ausgabe der Ortsdaten der beiden Pendelmassen
0006
0007  function fdpLagrange_Simulation(yData)
0008
0009      posdata=fposdata(yData);
0010      nbpionts=size(posdata,"c");
0011
0012      xmax=(l1+l2)*1.05;
0013      xmin=(-1)*xmax;
0014      ymax=xmax;
0015      ymin=xmin;
0016
0017      whiteid=color("white");
0018
0019      my_handle = scf(100001);
0020      clf(my_handle,"reset");
0021
0022      drawlater;
0023
0024      // Achsen zeichnen
0025      plot2d(0,0,-1,rect=[xmin,ymin,xmax,ymax],frameflag=7);
0026      drawnow();
0027
0028      // Nur keine Hektik
0029      sleep(3000);
0030      realtimeinit(stepTime);
0031      realtime(0);
0032
0033      // Schrittweise Ausgabe der Simulation
0034      for i=1:nbpionts
0035          drawlater;
0036
0037          if i>1 then
0038              // Gezeichnete Kreuze und Stangen der letzten Ausgabe
0039              // werden in weiß überschrieben.
0040
0041              plot2d(posdata(1,i-1),posdata(2,i-1),-1,rect=[xmin,ymin,xmax,ymax],frameflag=7)
0042              p = get("hdl");
0043              p.children.mark_mode = "on";
0044              p.children.mark_foreground = whiteid;
0045
0046              plot2d(posdata(3,i-1),posdata(4,i-1),-1,rect=[xmin,ymin,xmax,ymax],frameflag=7)
0047              p = get("hdl");
0048              p.children.mark_mode = "on";
0049              p.children.mark_foreground = whiteid;
0050
0051              px=[0;posdata(1,i-1)];
0052              py=[0;posdata(2,i-1)];
0053              e=xpoly(px,py);
0054              e.foreground=whiteid;
0055
0056              px=[posdata(1,i-1);posdata(3,i-1)];
0057              py=[posdata(2,i-1);posdata(4,i-1)];
0058              e=xpoly(px,py);
0059              e.foreground=whiteid;
0060          end;
0061
0062          // Zeichnet das untere Kreuz der oberen Stange das mittlere Kreuz), in schwarz
0063          plot2d(posdata(1,i),posdata(2,i),-1,rect=[xmin,ymin,xmax,ymax],frameflag=7);
0064
0065          // Zeichnet das untere Kreuz der unteren Stange, in schwarz
0066          plot2d(posdata(3,i),posdata(4,i),-1,rect=[xmin,ymin,xmax,ymax],frameflag=7);
0067
0068          // Zeichnet die obere Stange
0069          px=[0;posdata(1,i)];
0070          py=[0;posdata(2,i)];
0071          xpoly(px,py);
0072
0073          // Zeichnet die untere Stange
0074          px=[posdata(1,i);posdata(3,i)];
0075          py=[posdata(2,i);posdata(4,i)];
0076          xpoly(px,py);
0077
0078          // Zeichnet das fixe obere Kreuz der oberen Stange, in schwarz
0079          plot2d(0,0,-1,rect=[xmin,ymin,xmax,ymax],frameflag=7);
0080
0081          // Warten bis zum nächsten Zyklus
0082          realtime(i);
0083          drawnow();
0084      end;
0085
0086  endfunction;
0087
0088  // ###########################################################################
0089  disp('Geladen: dpLagrange-OrtsSimulation.sci');
```