prob06_3__sphaerisches_Pendel

Sage Notebook zur Aufgabe 6.3 Sphärisches Pendel

der Vorlesung

Theoretische Physik 1. Mechanik
Uni Leipzig, Wintersemester 2018/19
Autor: Jürgen Vollmer (2018)

Sage ist ein OpenSource Projekt, das viele Methoden der computerbasierten Mathematik und Computer-Algebra in einem Python-basierten Notebook anbietet.
Dokumentation und Informationen zur **Installation** findet man auf
https://sagemath.org
Eine hervorragende Einführung in das Arbeiten mit Sage bietet das Buch
Paul Zimmermann, u.a.: "Computational Mathematics with SageMath"
http://sagebook.gforge.inria.fr/english.html

Allgemeine Definitionen, Variablen, Konstanten

Pfad und Stammname für Abbildungen

Bitte den Pfad editiert und die Kommentarzeichen vor den "save_image()"-Befehlen entfernen, um die erstellten Dateien zu speichern.

```
baseName = 'XXX--bitte editieren--XXX/2018W_Mechanik/Uebungen
/Sage/prob06_3__sphaerisches_Pendel__'
```

Pakete laden für Plotten und Numerik

```
import scipy; from scipy import integrate
import numpy as np
```

Differentialgleichung

- als Funktion von t

Parameter

- $\Lambda = L_z/MR^2\omega$ dimensionsloser Drehimpuls
- $\omega^2 = g/L$ wird in Zeitskala absorbiert

```
Lambda = var('Lambda')

t = var('t')
def dTheta_dt(X, t=0) :
    return [ X[1], Lambda^2 * cos( X[0] ) / sin( X[0] )^3 + sin( X[0] ),
Lambda/sin( X[0] )^2 ]
```

effektives Potential

Funktion definieren

```
def PhiEff(theta, Lambda) :
    return cos(theta) + Lambda^2 / sin(theta)^2
```

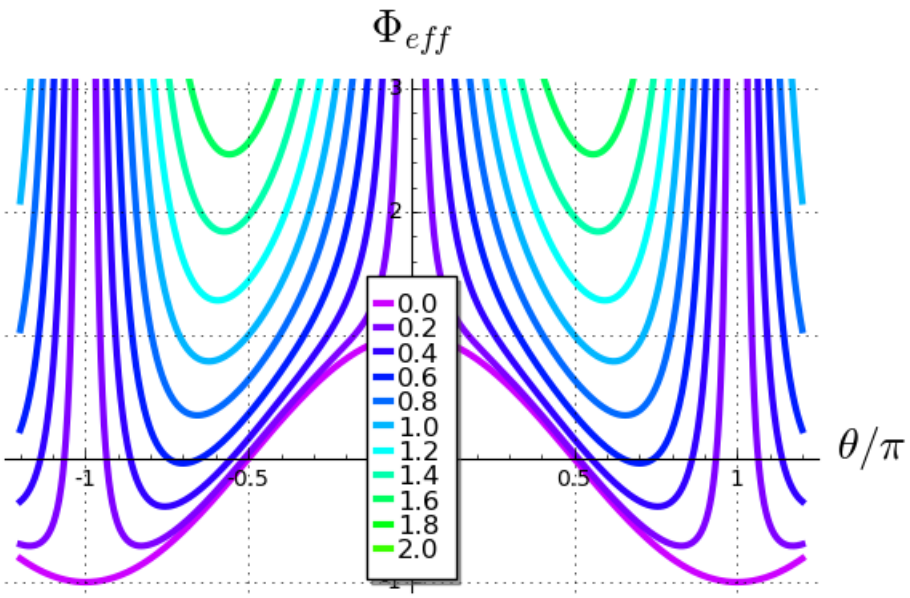
...und plotten

```
n = 11
p = plot( [] )

for j in range(n) :
    p += plot( PhiEff(x*pi, j/5), (-1.2,1.2), thickness=3, color=hue(.8-
float(j)/(1.8*n)), legend_label=j/5 )

p.axes_labels( [r'$\theta/\pi$', r'$\Phi_{eff}$'] )
p.axes_labels_size( 2 )
p.show( gridlines=True, ymax=3, figsize=[6,4] )

# p.save_image(baseName+'effektives_Potential.svg', gridlines=True, figsize=
[6,4])
```



Parameterabhängigkeit des nicht-trivialen Fixpunktes

implizite Darstellung auswerten

```
theta = np.linspace(-np.pi, np.pi, 200)
X = zip( sin(theta)^2/sqrt(-cos(theta)), theta/pi )
```

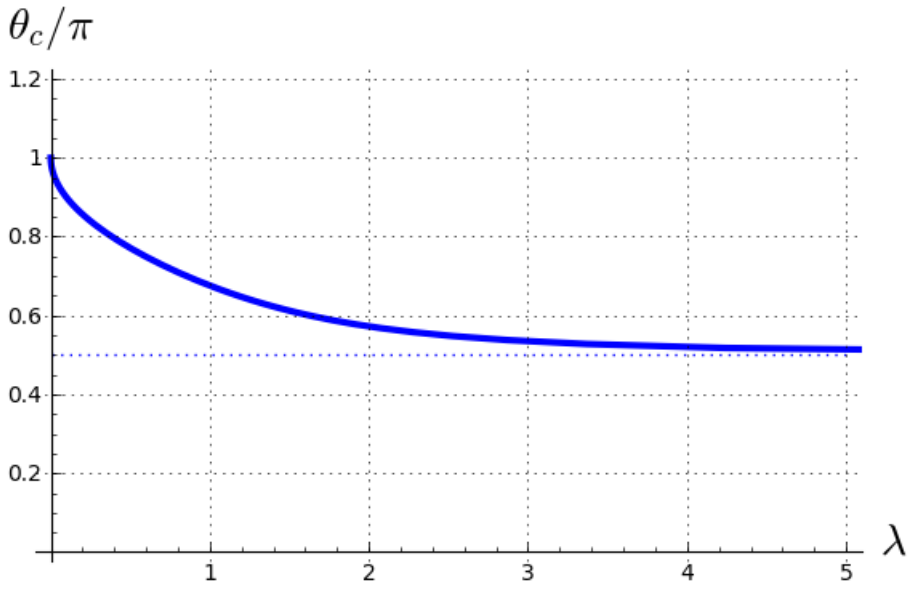
```
/usr/lib/python2.7/dist-packages/sage/functions/other.py:1995:
RuntimeWarning: invalid value encountered in sqrt
  return sqrt(x)
```

...und plotten

```
p  = line( X, thickness=3 )
p += plot( 0.5, (0,5), linestyle=':' )
```

```
p.axes_labels( [r'$\lambda$', r'$\theta_c/\pi$'] )
p.axes_labels_size( 2 )
p.show( gridlines=True, xmin=0, xmax=5, ymin=0, ymax=1.2, figsize=[6,4] )

# p.save_image(baseName+'theta_c.svg', gridlines=True, figsize=[6,4])
```



Phasenraumplot mit Vektorfeld

Definition des Vektorfeldes

```
def g(x,y) :
    v = vector( dTheta_dt([x,y]) )
    return v[0:2] / v[0:2].norm()
```

n Trajektorien für Oszillationen und Rotationen

- IC: Anfangsbedingungen ("Initial Conditions")
- Xo: Trajektorien, die oszillieren
- Xr: Trajektorien, die frei rotieren -- für Intervall [thetaL, thetaR]

$$\Lambda = 0.7$$

```
Lambda=0.7
```

```
n = 10
t = srange(0, 30, 0.05 )

V = VectorSpace(RR, 3)

# ... für Oszillationen
IC = srange(V([0.2,0,0]), V([0.9*pi,0,0]), step=V([0.7*pi/n,0,0]))
Xo = []
for j in range(n) :
    Xo.append( integrate.odeint( dTheta_dt, IC[j], t ) )
```

Graphik erstellen

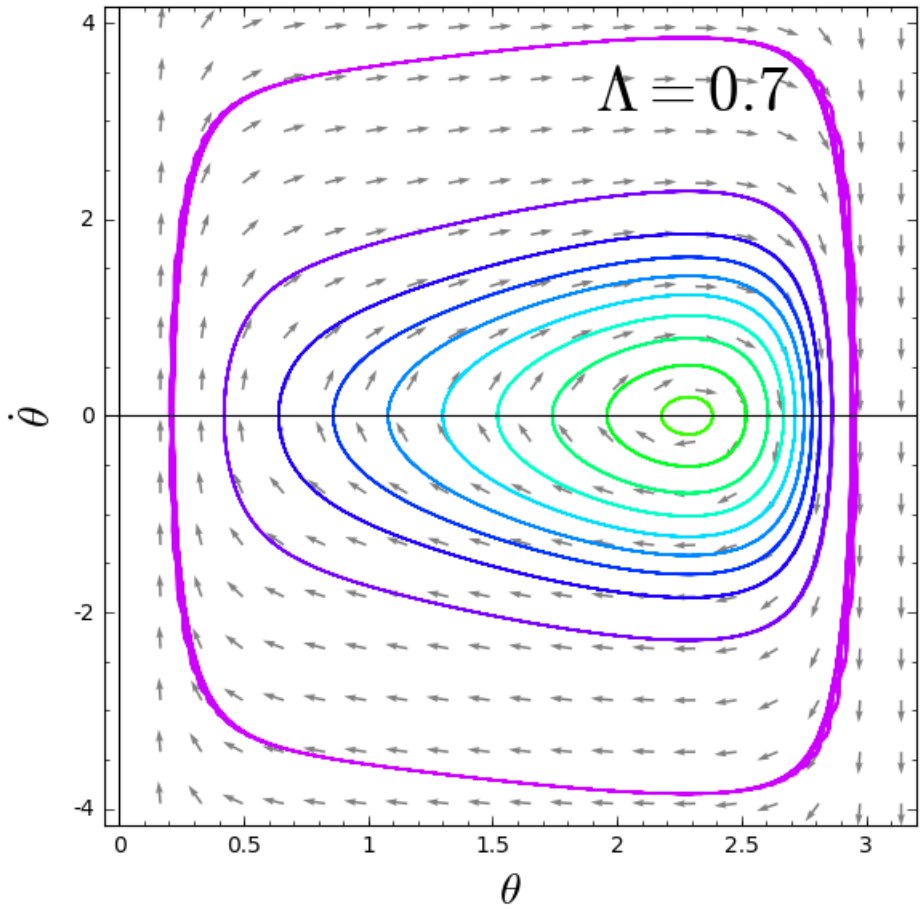
```
x,y = var('x', 'y')
q = plot_vector_field( g(x,y), (x, 0, pi), (y, -5, 5), color='gray' )
for j in range(n) :
    q += line( Xo[j].T[0:2].T, color=hue(.8-float(j)/(1.8*n)) )

q += text( r'$\Lambda = 0.7$', (2.3,3.3), fontsize=28, color='black' )

q.axes_labels( [r'$\theta$', r'$\dot\theta$'] )
q.axes_labels_size( 2 )

q.show(xmin=0, xmax=pi, ymin=-4, ymax=4, figsize=[6,6])

# q.save_image(baseName+'Phasenraum_07.svg', figsize=[6,6])
```



Trajektorien im Konfigurationsraum

```
Lambda=0.7
```

```
t = srange(0, 80, 0.02 )
IC = V([1.3,0,0])
Xo = integrate.odeint( dTheta_dt, IC, t )

Theta, dTheta, Phi = Xo.T
```

Bahn plotten

```
x,y = var('x', 'y')

q = line( zip(cos(Theta)*cos(Phi), cos(Theta)*sin(Phi)), color=hue(.8-
float(j)/(1.8*n)) )

q += text( r'$\Lambda = 0.7$',  (0.7,0.9), fontsize=28, color='black' )
q += text( r'$\dot\theta_0 = 0$', (0.75,0.75), fontsize=28, color='black' )

q.axes_labels([ r'$x$', r'$y$' ] )
q.axes_labels_size( 2 )

q.show(figsize=[6,6])

# q.save_image(baseName+'Bahn_07.svg', figsize=[6,6])
```