prob01_3__mathematisches_Pendel

Sage Notebook zur Aufgabe
1.3 Das mathematische 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/prob01_3__mathematisches_Pendel__'
```

Pakete laden für Plotten und Numerik

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

Differentialgleichung als Funktion von t

```
t = var('t')
def dX_dt(X, t=0) :
    return [ X[1], - sin( X[0] ) ]
```

Trajektorien plotten

3 Trajektorien berechnen

```
t = srange(0, 30, 0.01 )
X1ini = [ -pi, 0.001 ]
X1 = integrate.odeint( dX_dt, X1ini, t )

X2ini = [ -pi+0.001, 0 ]
X2 = integrate.odeint( dX_dt, X2ini, t )
```

```
X3ini = [ -0.2*pi, 0 ]
X3 = integrate.odeint( dX_dt, X3ini, t )
```

...und plotten

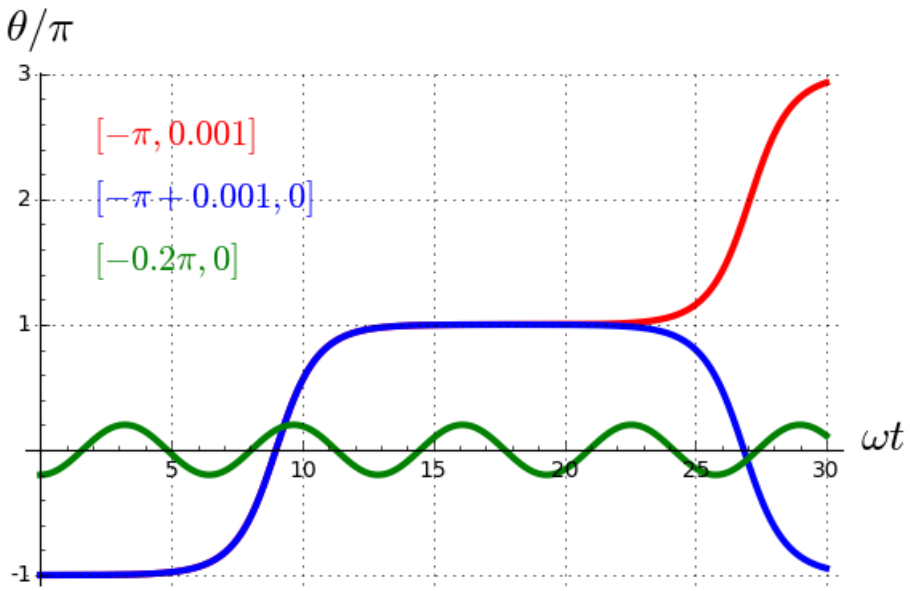
```
theta1, dtheta = X1.T
p  = line( zip( t, theta1/pi ), thickness=3, color='red' )
p += text( r'$[ -\pi, 0.001 ]$', (2,2.5), fontsize=16,
horizontal_alignment='left', color='red' )

theta2, dtheta = X2.T
p += line( zip( t, theta2/pi ), thickness=3, color='blue' )
p += text( r'$[ -\pi+0.001, 0 ]$', (2,2), fontsize=16,
horizontal_alignment='left', color='blue' )

theta3, dtheta = X3.T
p += line( zip( t, theta3/pi ), thickness=3, color='green' )
p += text( r'$[ -0.2 \pi, 0 ]$', (2,1.5), fontsize=16,
horizontal_alignment='left', color='green' )

p.axes_labels( [r'$\omega t$', r'$\theta/\pi$'] )
p.axes_labels_size( 2 )
p.show( gridlines=True, figsize=[6,4])

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



Phasenraumplot mit Vektorfeld

Definition des Vektorfeldes

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

n Trajektorien für Oszillationen und Rotationen

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

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

V = VectorSpace(RR, 2)

# ... für Oszillationen
IC = srange(V([-pi,0]), V([0,0]), step=V([pi/n,0]))
Xo = []
for j in range(n) :
    Xo.append( integrate.odeint( dX_dt, IC[j], t ) )

# ... für Rotation
IC = srange(V([-3*pi,0]), V([-3*pi,3]), step=V([0, 3/n]))
Xr = []
for j in range(n) :
    Xr.append( integrate.odeint( dX_dt, IC[j], t ) )
```

homocline Trajektorien

```
pts = 1000
thetaVals = np.linspace(-10, 10, 200 )
homoclin1 = zip( thetaVals, 2*cos(thetaVals/2) )
homoclin2 = zip( thetaVals,-2*cos(thetaVals/2) )
```

Graphik erstellen

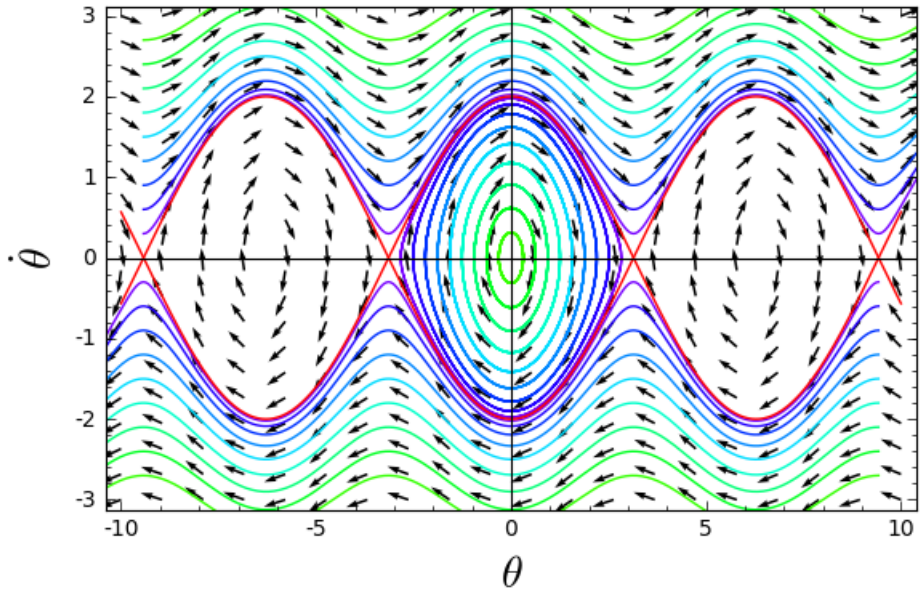
```
x,y = var('x', 'y')
q = plot_vector_field( g(x,y), (x,-10, 10), (y, -3, 3) )
for j in range(n) :
    q += line( Xo[j], color=hue(.8-float(j)/(1.8*n)) )
    q += line( Xr[j], color=hue(.8-float(j)/(1.8*n)) )
    q += line(-Xr[j], color=hue(.8-float(j)/(1.8*n)) )

q += line( homoclin1, color='red' )
q += line( homoclin2, color='red' )

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

q.show(xmin=-10, xmax=10, ymin=-3, ymax=3, figsize=[6,4])

# q.save_image(baseName+'Phasenraum.svg', figsize=[6,4])
```


3
2
1
0
-1
-2
-3
-10
-5
0
5
10
$\dot{\theta}$
θ