prob07_3__Symmetrien_Erhaltungsgroessen

Sage Notebook zur Aufgabe
7.3 Symmetrien und Erhaltungsgrößen

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/prob07_3__Symmetrien_Erhaltungsgroessen__'
```

Pakete laden für Plotten und Numerik

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

Differentialgleichung

- als Funktion von t

Parameter

- $\omega^2 = g/a\Delta_m^2$ wird in Zeitskala absorbiert

```
t = var('t')
def dDelta_dt(X, t=0) :
    return [ X[1], -X[0] * (1+X[1]^2) / (1 + X[0]^2) ]
```

effektives Potential

Funktion definieren

```
DeltaMax = var('DeltaMax')
def PhiEff(Delta, DeltaMax) :
    return 0.5 * ( 1. - (1.+DeltaMax^2)/(1+Delta^2) )
```

...und plotten

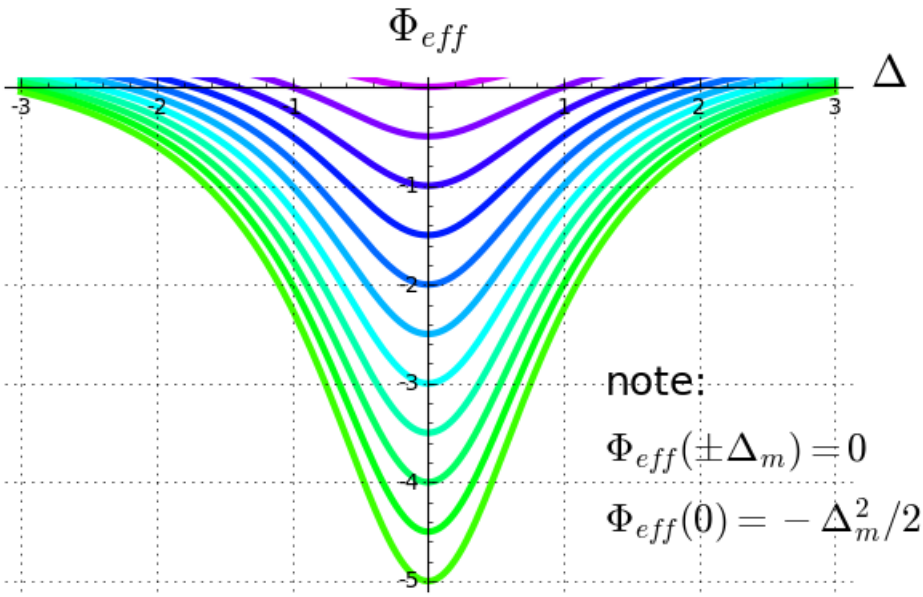
```
n = 11
p = plot( [] )

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

p += text( r'note:',                          (1.3,-3.0), fontsize=18,
color='black', horizontal_alignment='left' )
p += text( r'$\Phi_{eff}(\pm\Delta_m) = 0$',   (1.3,-3.7), fontsize=18,
color='black', horizontal_alignment='left' )
p += text( r'$\Phi_{eff}(0) = -\Delta_m^2/2$', (1.3,-4.4), fontsize=18,
color='black', horizontal_alignment='left' )

p.axes_labels( [r'$\Delta$', r'$\Phi_{eff}$'] )
p.axes_labels_size( 2 )
p.show( gridlines=True, figsize=[6,4], ymax=0 )

# p.save_image(baseName+'effektives_Potential.svg', gridlines=True, figsize=
[6,4], ymax=0)
```



Phasenraumplot mit Vektorfeld

Definition des Vektorfeldes

```
def g(x,y) :
    v = vector( dDelta_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]

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

V = VectorSpace(RR, 2)

# ... für Oszillationen
IC = srange(V([-4,0]), V([0,0]), step=V([4/n,0]))
Xo = []
for j in range(n) :
    Xo.append( integrate.odeint( dDelta_dt, IC[j], t ) )
```

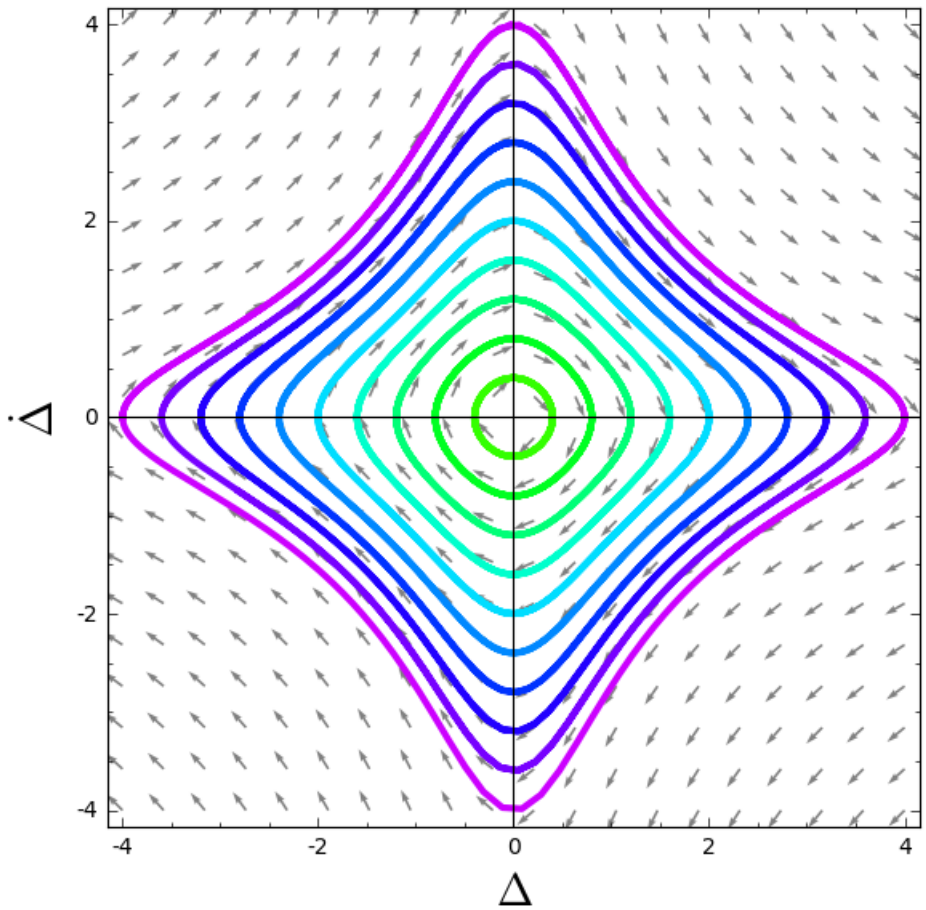
Graphik erstellen

```
x,y = var('x', 'y')
q = plot_vector_field( g(x,y), (x, -4, 4), (y, -4, 4), color='gray' )
for j in range(n) :
    q += line( Xo[j].T[0:2].T, color=hue(.8-float(j)/(1.8*n)), thickness=3 )

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

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

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



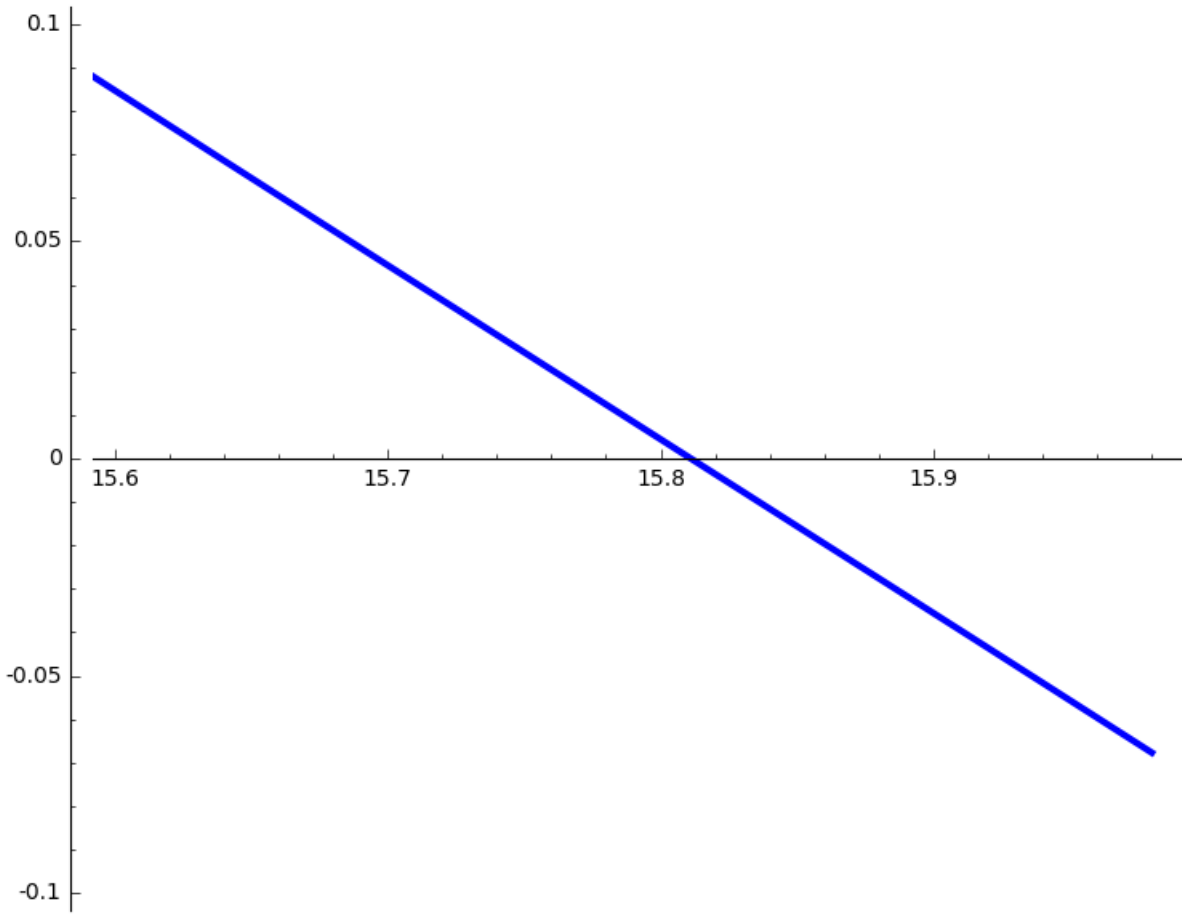
Trajektorien im Konfigurationsraum

```
DeltaMax = 2.0
Tcrit   = asin(1./DeltaMax)
DeltaCrit = DeltaMax * sin(Tcrit)
```

```
t = srange(0, 16, 0.02 )
IC = V([-DeltaMax, 0.0])
Xo = integrate.odeint( dDelta_dt, IC, t )

Theta, dTheta = Xo.T
```

```
Tperiod = 15.85 * 2/3
line( zip(t, dTheta), color='blue', thickness=3, xmin=15.6, xmax=16, ymin=-.1,
ymax=.1 )
```



Bahn plotten

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

q  = line( zip(t, Theta) , color='red' , thickness=3 )
q += line( zip(t, dTheta), color='blue', thickness=3 )

q += plot(  DeltaCrit, (x, 0, 16), color='gray')
q += plot( -DeltaCrit, (x, 0, 16), color='gray')

q += plot( -sqrt( DeltaMax^2 -  x^2              ),    (x, 0           ,
0.19*Tperiod ), color='black', thickness=4, linestyle='--' )
```

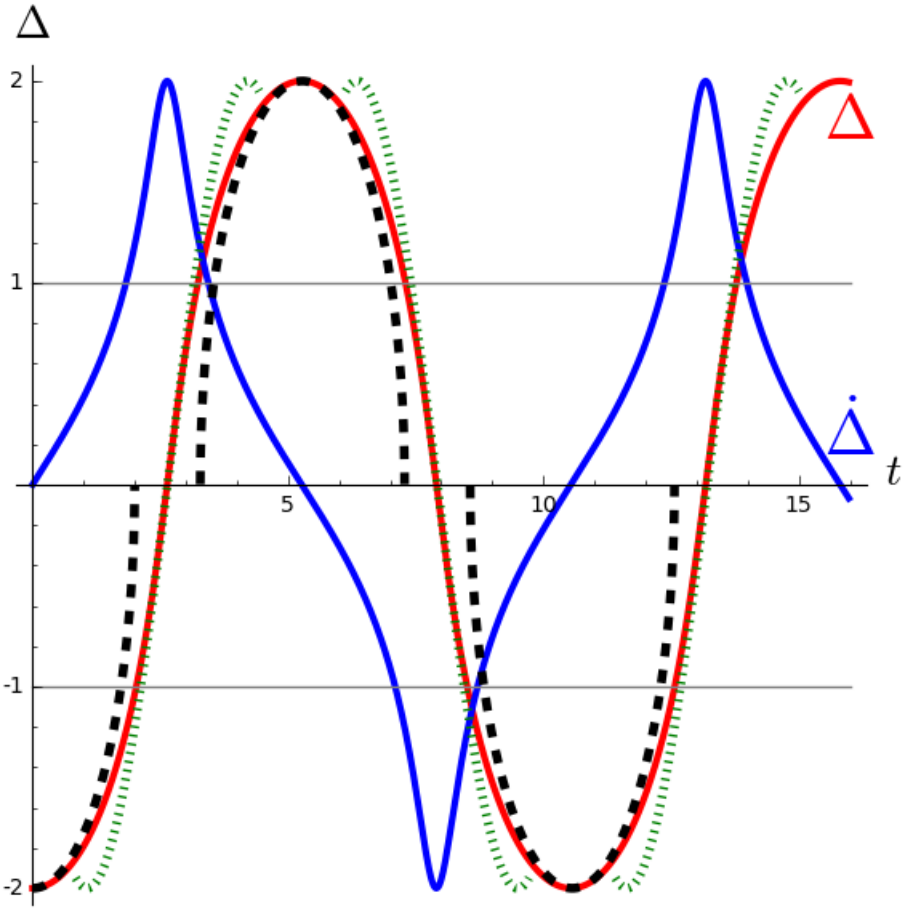
```
q += plot(  DeltaMax * sin(     x - 0.25*Tperiod ),    (x, 0.08*Tperiod,
0.42*Tperiod ), color='green', thickness=4, linestyle=':' )
q += plot(  sqrt( DeltaMax^2 - (x - 0.5 *Tperiod)^2 ), (x, 0.31*Tperiod,
0.69*Tperiod ), color='black', thickness=4, linestyle='--' )
q += plot( -DeltaMax * sin(     x - 0.75*Tperiod    ), (x, 0.58*Tperiod,
0.92*Tperiod ), color='green', thickness=4, linestyle=':' )
q += plot( -sqrt( DeltaMax^2 - (x -      Tperiod)^2 ), (x, 0.81*Tperiod,
1.19*Tperiod ), color='black', thickness=4, linestyle='--' )
q += plot(  DeltaMax * sin(     x - 1.25*Tperiod ),    (x, 1.08*Tperiod,
1.42*Tperiod ), color='green', thickness=4, linestyle=':' )

q += text( r'$\Delta$',     (16, 1.8 ), fontsize=28, color='red'  )
q += text( r'$\dot\Delta$', (16, 0.25), fontsize=28, color='blue' )

q.axes_labels([ r'$t$', r'$\Delta$' ] )
q.axes_labels_size( 2 )

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

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




```
factor = 0.85

q  = plot(  DeltaCrit, (x, 0, 16), color='gray')
q += plot( -DeltaCrit, (x, 0, 16), color='gray')

q += plot( -sqrt( DeltaMax^2 -  x^2                      ),    (x, 0
, 0.23*factor*Tperiod ), color='black', thickness=4, linestyle='--' )
q += plot(  DeltaMax * sin(     x - 0.25*factor*Tperiod ),    (x,
0.05*factor*Tperiod, 0.45*factor*Tperiod ), color='green', thickness=2,
linestyle='-' )
```

```
q += plot(  sqrt( DeltaMax^2 - (x - 0.5 *factor*Tperiod)^2 ), (x,
0.27*factor*Tperiod, 0.73*factor*Tperiod ), color='black', thickness=4,
linestyle='--' )
q += plot( -DeltaMax * sin(     x - 0.75*factor*Tperiod    ), (x,
0.55*factor*Tperiod, 0.95*factor*Tperiod ), color='green', thickness=2,
linestyle='-' )
q += plot( -sqrt( DeltaMax^2 - (x -      factor*Tperiod)^2 ), (x,
0.77*factor*Tperiod, 1.23*factor*Tperiod ), color='black', thickness=4,
linestyle='--' )
q += plot(  DeltaMax * sin(     x - 1.25*factor*Tperiod ),    (x,
1.05*factor*Tperiod, 1.45*factor*Tperiod ), color='green', thickness=2,
linestyle='-' )

q += text( r'Periode um', (10, 1.7), fontsize=26, color='black',
horizontal_alignment='left' )
q += text( r'$85$% kuerzer!', (10, 1.3), fontsize=26, color='black',
horizontal_alignment='left' )

q.axes_labels([ r'$t$', r'$\Delta$' ] )
q.axes_labels_size( 2 )

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

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