prob05_B__Fliehkraftregler

Sage Notebook zur Aufgabe 5.B Der Fliehkraftregler

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/prob05_B__Fliehkraftregler__'
```

Pakete laden für Plotten und Numerik

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

Differentialgleichung

- als Funktion von t

Parameter

- ρ Stärke der Rotation
- γ Dissipation

```
rho = var('rho')
gamma = var('gamma')

t = var('t')
def dTheta_dt(X, t=0) :
    return [ X[1], -gamma * X[1] - sin( X[0] ) * ( 1-rho^2*cos(X[0]) ) ]
```

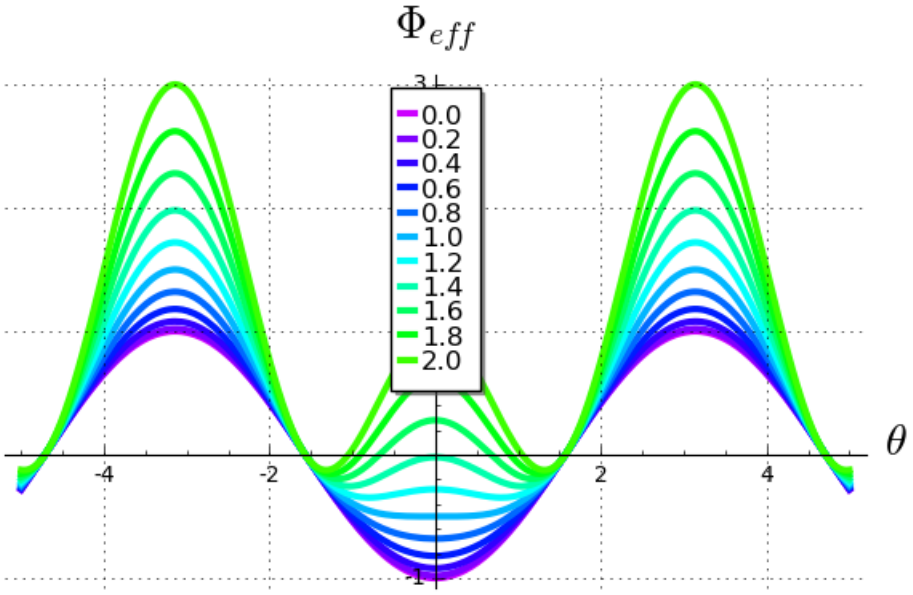
effektives Potential

Funktion definieren

```
def PhiEff(theta, rho) :
    return -cos(theta) + rho^2/2 * cos(theta)^2
```

...und plotten

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

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

p.axes_labels( [r'$\theta$', r'$\Phi_{eff}$'] )
p.legend('right')

p.axes_labels_size( 2 )
p.show( gridlines=True, figsize=[6,4] )

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



Parameterabhängigkeit des nicht-trivialen Fixpunktes

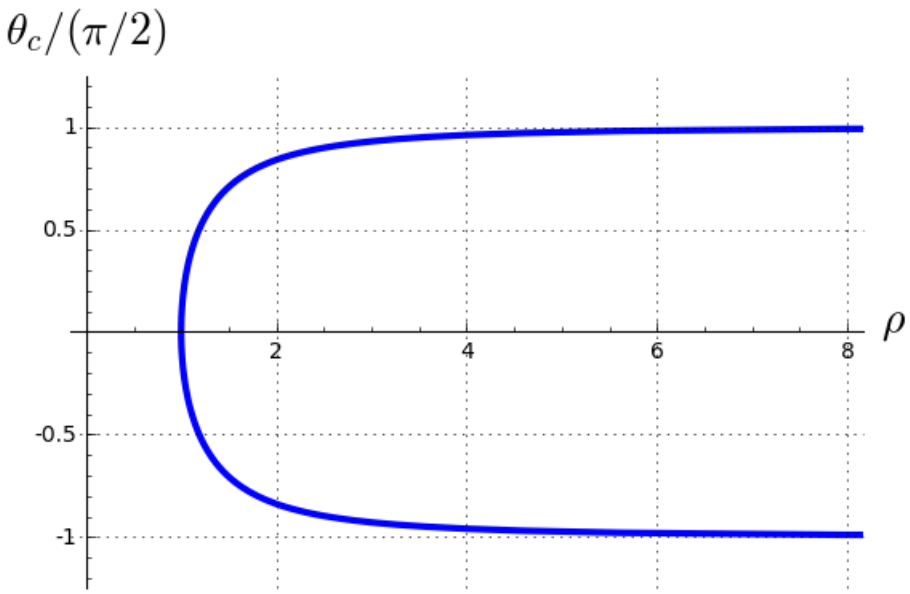
Funktion definieren

```
def theta_c(rho) :
    return arccos(1 / rho^2 )
```

...und plotten

```
theta = np.linspace(-np.pi/2, np.pi/2, 200)
X = zip( 1/sqrt(cos(theta)), 2*theta/pi )
```

```
p  = line( X, thickness=3 )

p.axes_labels( [r'$\rho$', r'$\theta_c/(\pi/2)$'] )
p.axes_labels_size( 2 )
p.show( gridlines=True, xmin=0, xmax=8, ymin=-1.2, 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 / 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]

$\rho = 0.7$ **und** $\gamma = 0$

```
rho=0.7
gamma = 0.0
```

```
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) :
```

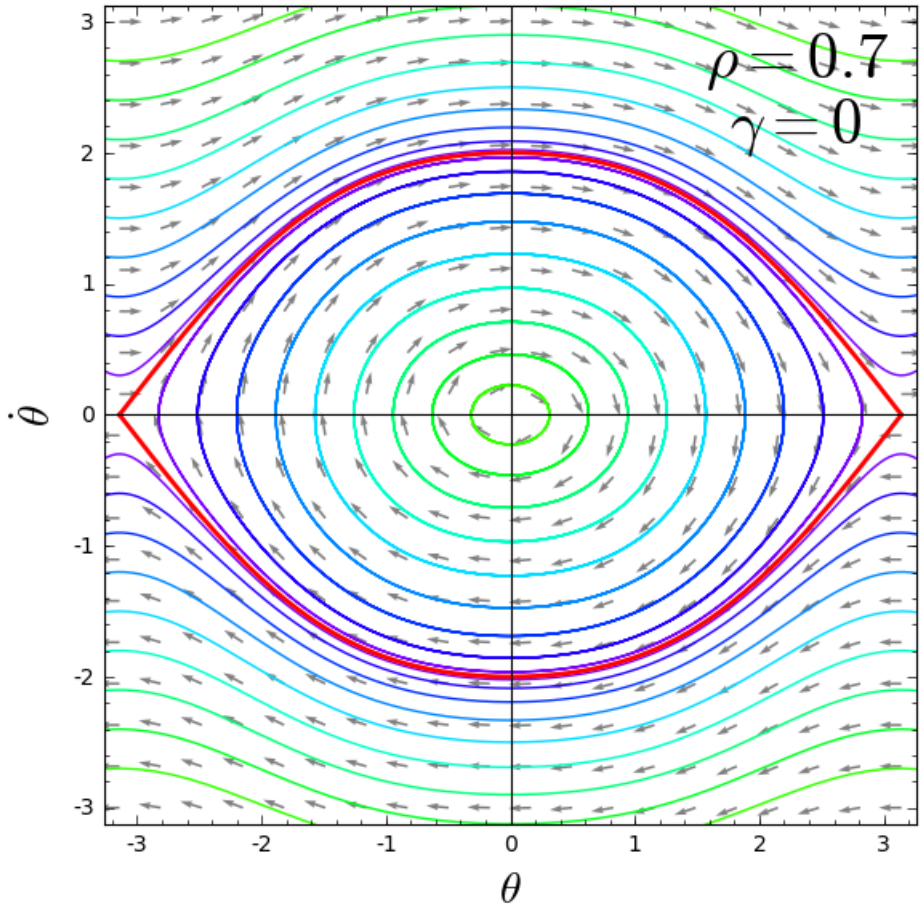
```
    Xo.append( integrate.odeint( dTheta_dt, IC[j], t ) )

# ... 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( dTheta_dt, IC[j], t ) )
```

homocline Trajektorien

```
pts = 1000
thetaVals = np.linspace(-np.pi, np.pi, 200 )
homoclin1 = zip( thetaVals,  sqrt( 2 + 2*cos(thetaVals) + rho^2 *
sin(thetaVals)^2 ) )
homoclin2 = zip( thetaVals, -sqrt( 2 + 2*cos(thetaVals) + rho^2 *
sin(thetaVals)^2 ) )
```

Graphik erstellen

```
x,y = var('x', 'y')
q = plot_vector_field( g(x,y), (x,-pi, pi), (y, -3, 3), color='gray' )
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', thickness=2 )
q += line( homoclin2, color='red', thickness=2 )

q += text( r'$\rho = 0.7$', (2.3,2.7), fontsize=28, color='black' )
q += text( r'$\gamma = 0$', (2.3,2.2), fontsize=28, color='black' )
q.axes_labels([ r'$\theta$', r'$\dot\theta$' ] )
q.axes_labels_size( 2 )

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

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



$\rho = 1.5$ **und** $\gamma = 0$

```
rho=1.5
gamma=0.0
```

```
t = srange(0, 30, 0.05 )
V = VectorSpace(RR, 2)

# ... für Oszillationen
no = 9
IC = srange(V([-pi, 0]), V([-1.2, 0]), step=V([(pi-1.2)/no,0]))
Xo = []
for j in range(no) :
    Xo.append( integrate.odeint( dTheta_dt, IC[j], t ) )

# ... für Rotation
nr = 8
IC = srange(V([-3*pi,0]), V([-3*pi,3]), step=V([0, 3/nr]))
Xr = []
for j in range(nr) :
    Xr.append( integrate.odeint( dTheta_dt, IC[j], t ) )
```

homocline Trajektorien

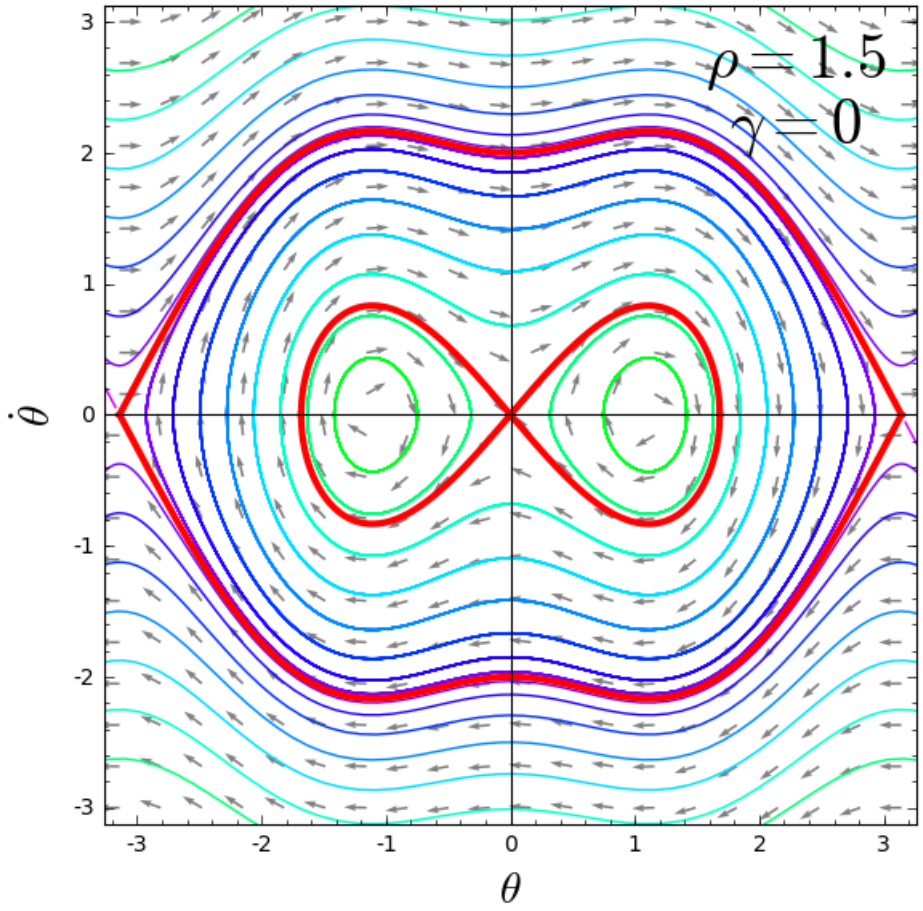
```
theta_crit = solve( -x+rho^2/2 * x^2 == -1+rho^2/2, x)
theta_crit
```

$$\left[x = 1, x = \left(-\frac{1}{9}\right)\right]$$

```
pts = 1000

thetaVals = np.linspace(-np.pi, np.pi, pts )
homoclin1 = zip( thetaVals,  sqrt( 2 + 2*cos(thetaVals) + rho^2 *
sin(thetaVals)^2 ) )
homoclin2 = zip( thetaVals, -sqrt( 2 + 2*cos(thetaVals) + rho^2 *
sin(thetaVals)^2 ) )


thetaVals = np.linspace( -float( acos(-1/9) ), float( acos(-1/9) ), pts )
homoclin3 = zip( thetaVals,  sqrt( -2 + 2*cos(thetaVals) + rho^2 *
sin(thetaVals)^2 ) )
homoclin4 = zip( thetaVals, -sqrt( -2 + 2*cos(thetaVals) + rho^2 *
sin(thetaVals)^2 ) )
```

Graphik erstellen

```
x,y = var('x', 'y')
q = plot_vector_field( g(x,y), (x,-pi, pi), (y, -3, 3), color='gray' )
for j in range(no) :
    q += line( Xo[j], color=hue(.8-float(j)/(1.8*n)) )
    q += line(-Xo[j], color=hue(.8-float(j)/(1.8*n)) )

for j in range(nr) :
    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', thickness=3 )
q += line( homoclin2, color='red', thickness=3 )
q += line( homoclin3, color='red', thickness=3 )
q += line( homoclin4, color='red', thickness=3 )

q += text( r'$\rho = 1.5$', (2.3,2.7), fontsize=28, color='black' )
q += text( r'$\gamma = 0$', (2.3,2.2), fontsize=28, color='black' )
q.axes_labels([ r'$\theta$', r'$\dot\theta$' ] )
q.axes_labels_size( 2 )

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

# q.save_image(baseName+'Phasenraum_15.svg', figsize=[6,6])
```



$\rho = 0.7$ **und** $\gamma = 0.2$

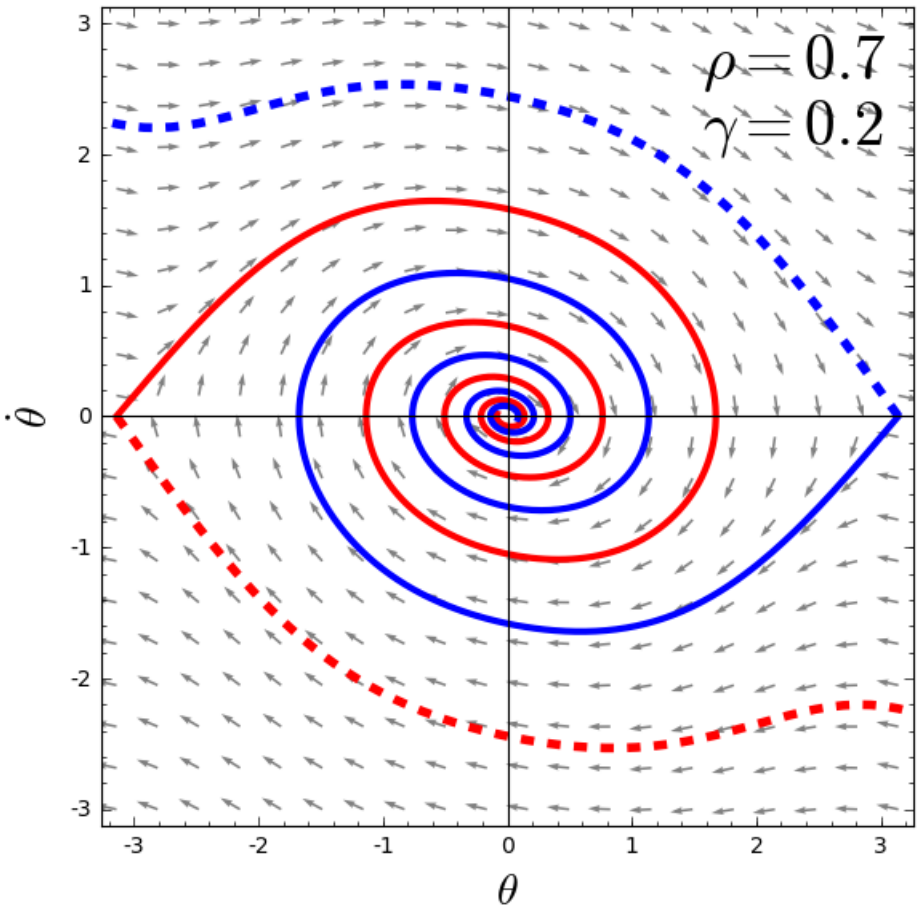
```
rho=0.7
gamma = 0.2
```

```
# instabile Mannigfaltigkeit
ti = srange(0, 40, 0.05 )
IC = V([-pi+1e-3 , 1e-3])
Mi,dMi = integrate.odeint( dTheta_dt, IC, ti ).T

# stabile  Mannigfaltigkeit
ts = srange(0, -20, -0.05 )
IC = V([pi-1e-5 , 1e-5])
Ms,dMs  = integrate.odeint( dTheta_dt, IC, ts ).T
```

Graphik erstellen

```
x,y = var('x', 'y')
q = plot_vector_field( g(x,y), (x,-pi, pi), (y, -3, 3), color='gray' )

q += line( zip( Mi,  dMi), color='red'  , thickness=3 )
q += line( zip(-Mi, -dMi), color='blue'  , thickness=3 )

q += line( zip( Ms,  dMs), color='blue'  , thickness=4, linestyle='--' )
q += line( zip(-Ms, -dMs), color='red'  , thickness=4, linestyle='--' )
```

```
q += text( r'$\rho = 0.7$', (2.3,2.7), fontsize=28, color='black' )
q += text( r'$\gamma = 0.2$', (2.3,2.2), fontsize=28, color='black' )
q.axes_labels([ r'$\theta$', r'$\dot\theta$' ] )
q.axes_labels_size( 2 )

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

# q.save_image(baseName+'Phasenraum_07__gamma_02.svg', figsize=[6,6])
```



$\rho = 1.5$ **und** $\gamma = 0.2$

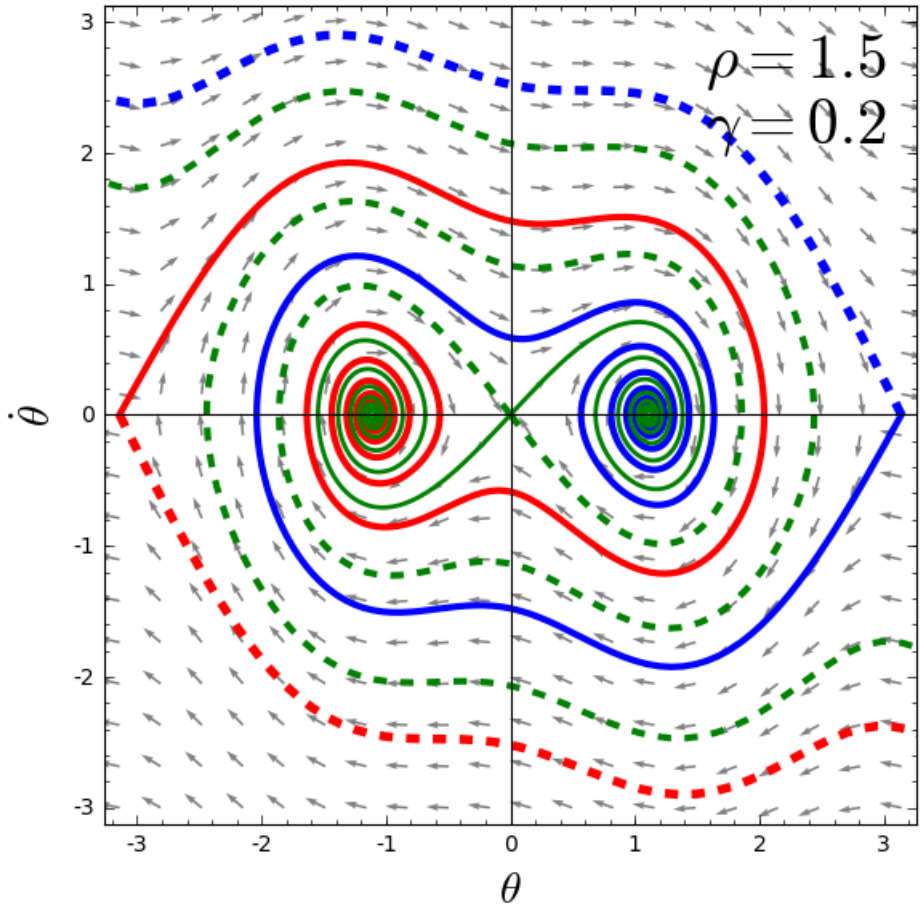
```
rho=1.5
gamma = 0.2
```

```
# instabile Mannigfaltigkeit von (-pi, 0)
ti = srange(0, 40, 0.05 )
IC = V([-pi+1e-3 , 1e-3])
MiPi,dMiPi = integrate.odeint( dTheta_dt, IC, ti ).T

# stabile  Mannigfaltigkeit (pi, 0)
ts = srange(0, -20, -0.05 )
IC = V([pi-1e-5 , 1e-5])
MsPi,dMsPi  = integrate.odeint( dTheta_dt, IC, ts ).T

# instabile Mannigfaltigkeit von (0, 0)
ti = srange(0, 40, 0.05 )
IC = V([1e-3 , 1e-3])
MiNull,dMiNull = integrate.odeint( dTheta_dt, IC, ti ).T
```

```
# stabile  Mannigfaltigkeit (0, 0)
ts = srange(0, -20, -0.05 )
IC = V([-1e-5 , 1e-5])
MsNull,dMsNull  = integrate.odeint( dTheta_dt, IC, ts ).T
```

Graphik erstellen

```
x,y = var('x', 'y')
q = plot_vector_field( g(x,y), (x,-pi, pi), (y, -3, 3), color='gray' )

q += line( zip( MiPi,  dMiPi), color='red'  , thickness=3 )
q += line( zip(-MiPi, -dMiPi), color='blue' , thickness=3 )

q += line( zip( MsPi,  dMsPi), color='blue' , thickness=4, linestyle='--' )
q += line( zip(-MsPi, -dMsPi), color='red'  , thickness=4, linestyle='--' )

q += line( zip( MiNull,  dMiNull), color='green' , thickness=2 )
q += line( zip(-MiNull, -dMiNull), color='green' , thickness=2 )

q += line( zip( MsNull,  dMsNull), color='green' , thickness=3, linestyle='--'
)
q += line( zip(-MsNull, -dMsNull), color='green' , thickness=3, linestyle='--'
)

q += text( r'$\rho = 1.5$', (2.3,2.7), fontsize=28, color='black' )
q += text( r'$\gamma = 0.2$', (2.3,2.2), fontsize=28, color='black' )
q.axes_labels([ r'$\theta$', r'$\dot\theta$' ] )
q.axes_labels_size( 2 )

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

# q.save_image(baseName+'Phasenraum_07__gamma_02.svg', figsize=[6,6])
```


$\rho = 1.5$
$\gamma = 0.2$
$\dot{\theta}$
θ
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3