Papilduzdevumi_teor_fiz

Unknown Author

December 20, 2013

Contents

In [1]:
```
!ipython3 nbconvert --to=latex Papilduzdevumi_teor_fiz.ipynb --template="latex/toc_latex.tplx"
```

```
[NbConvertApp] Using existing profile dir:
'/home/akels/.config/ipython/profile_default'
[NbConvertApp] Converting notebook Papilduzdevumi_teor_fiz.ipynb to
latex
[NbConvertApp] Support files will be in Papilduzdevumi_teor_fiz_files/
[NbConvertApp] Loaded template latex/toc_latex.tplx
[NbConvertApp] Writing 82517 bytes to Papilduzdevumi_teor_fiz.tex
```

In [2]:
```
!xelatex Papilduzdevumi_teor_fiz.tex > /dev/null ;\
rm -rf Papilduzdevumi_teor_fiz.out Papilduzdevumi_teor_fiz.log Papilduzdevumi_teor_fiz.idx Papilduzd
```

In [3]:
```
%config InlineBackend.figure_format = 'svg'
pylab.rcParams['figure.figsize'] = (10.0, 8.0)
```

In [5]:
```
rc('axes',grid=True)
rc('text',usetex=False)
rc('axes',titlesize=10)
rc('axes',labelsize=12)

from pylab import *
from monochrome import setFigLinesBW

def legend(fontsize=None,**kw):
    from pylab import legend as Oldlegend
    setFigLinesBW(gcf())
    Oldlegend(fontsize=8,**kw)
```

1 11. Oktobris

1.1 Dzīvais kompass

$$\dot{x} = \cos(\gamma\tau - \beta)\dot{y} = \sin(\gamma\tau - \beta)\dot{\beta} = \gamma - \sin\beta$$

Kur γ:

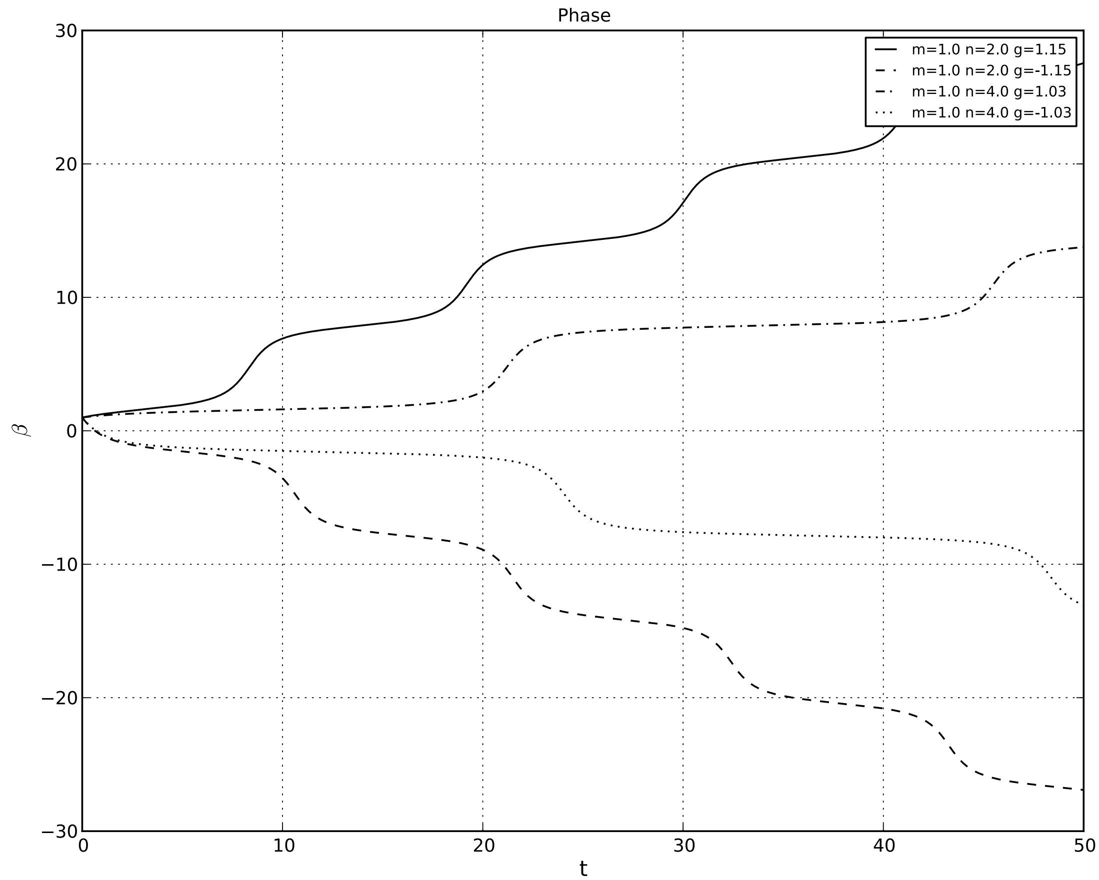
$\gamma = \frac{1}{\sqrt{1-(m/n)^2}}, \; m, n = 1, 2, \ldots$

In [6]:
```
from scipy.integrate import odeint

def f(r,tau,gamma):
    beta,x,y = r
    Dx = cos(gamma*tau - beta)
    Dy = sin(gamma*tau - beta)
    Dbeta = gamma - sin(beta)
    Dr = [Dbeta, Dx, Dy]
    return Dr
```

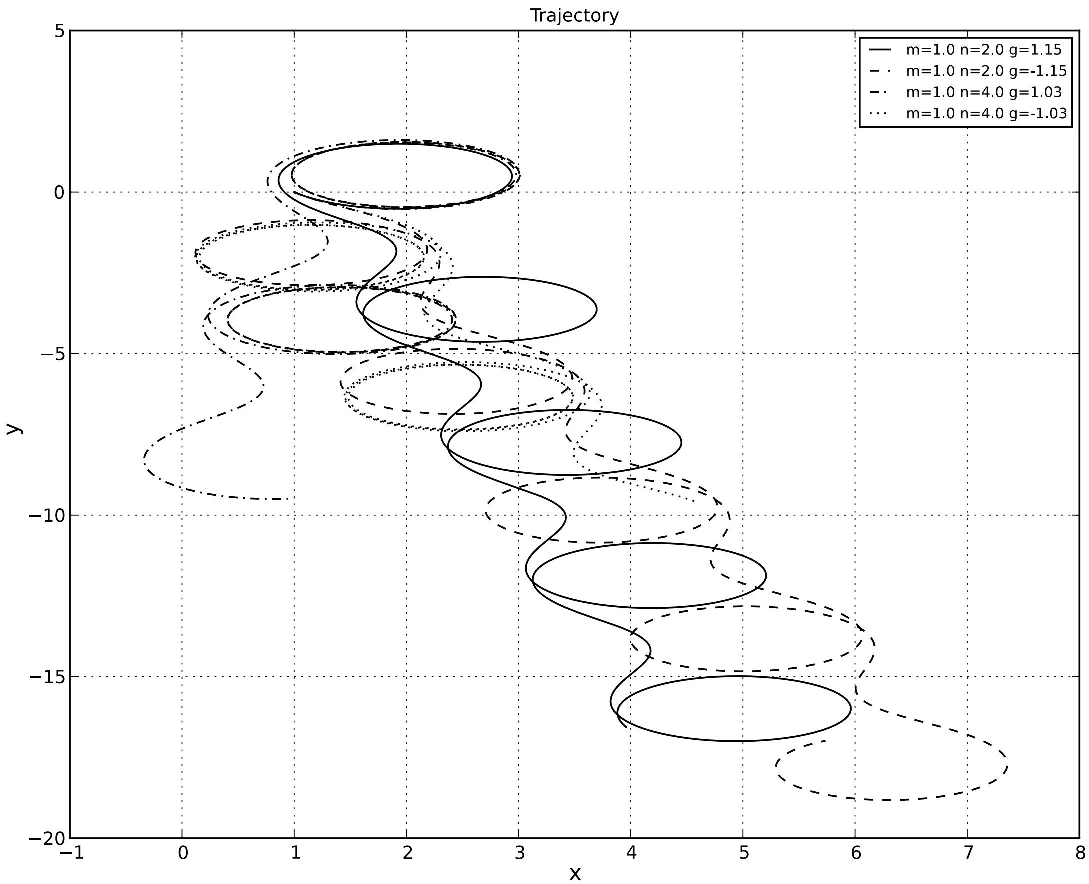
In [7]:
```
def plotpar(gammap):

    phase_fig = figure('Phase')
    title('Phase')
    xlabel('t')
    ylabel(r'$\beta$')


    traj_fig = figure('Trajectory')
    title('Trajectory')
    xlabel('x')
    ylabel('y')


    #gammap.append((1.,1.,1e6))

    for m,n,gamma in gammap:

        tpoints = linspace(0,50,500)
        y0 = [1,1,0]
        data = odeint(f,y0,tpoints,args=(gamma,))

        figure('Phase')
        plot(tpoints,data[:,0],label='m={0} n={1} g={2:.3}'.format(m,n,gamma))

        figure('Trajectory')
        plot(data[:,1],data[:,2],label='m={0} n={1} g={2:.3}'.format(m,n,gamma))


    figure('Phase')
    legend(fontsize=7)

    figure('Trajectory')
    legend(fontsize=7)

    return traj_fig
```

m=1, n=2,4

In [8]:
```
mpoints = [1.]
npoints = [2.,4.]
gammap = []
for m in mpoints:
    for n in npoints:
        gamma = 1/sqrt(1 -(m/n)**2)
        gammap.append((m,n,+gamma))
        gammap.append((m,n,-gamma))

traj_fig=plotpar(gammap)
traj_fig.savefig('PU6.pdf')
```

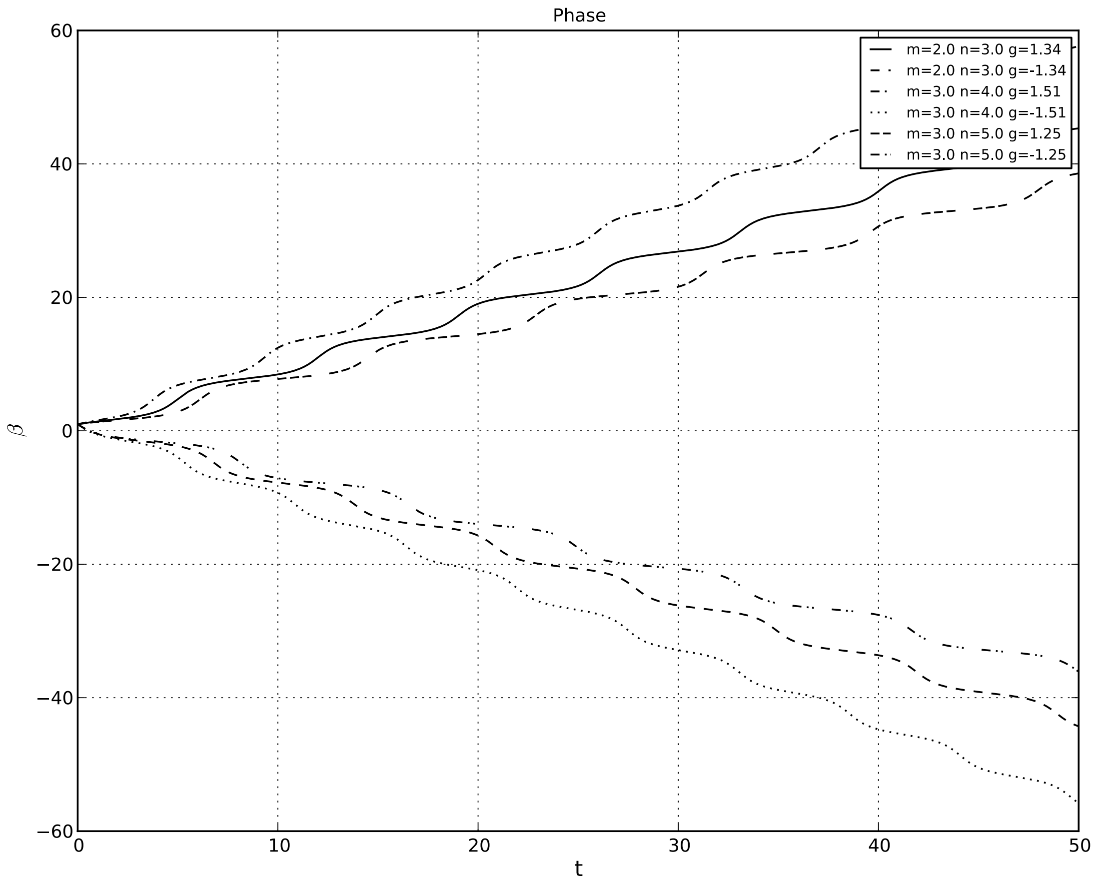


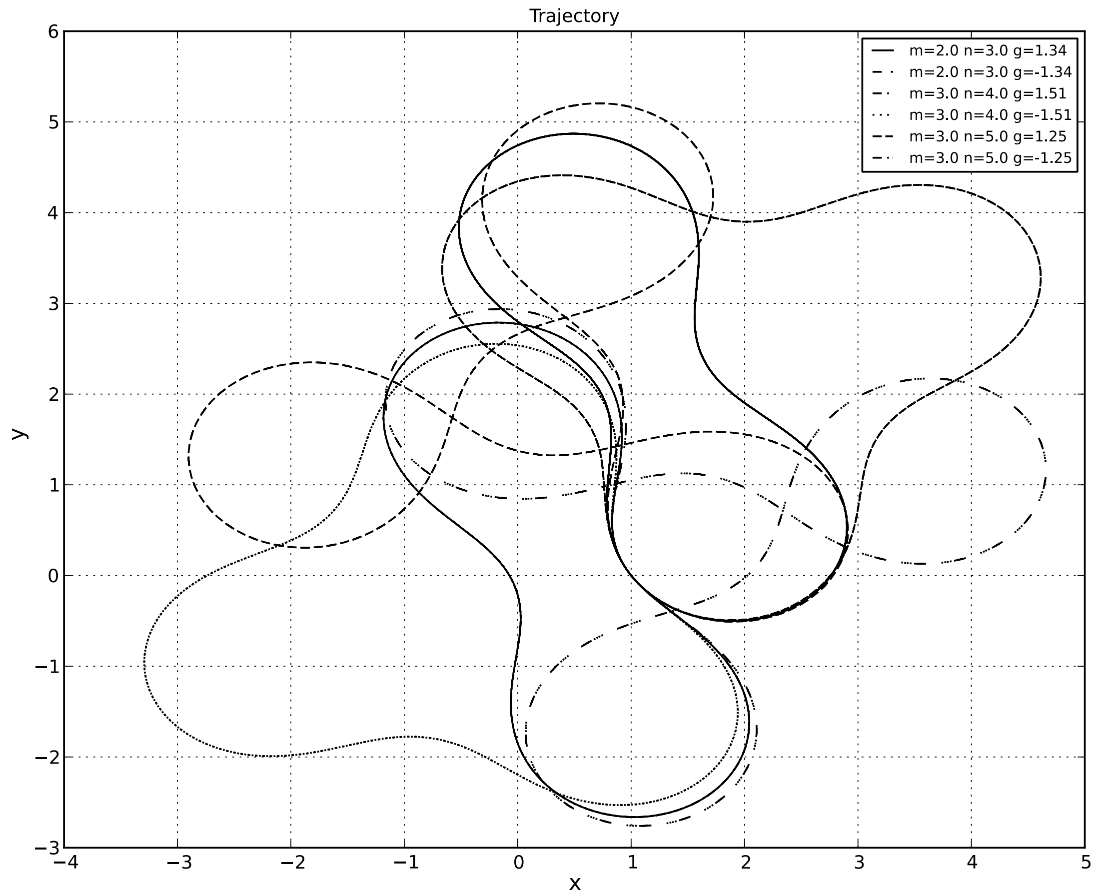
Phase
m=1.0 n=2.0 g=1.15
m=1.0 n=2.0 g=-1.15
m=1.0 n=4.0 g=1.03
m=1.0 n=4.0 g=-1.03
30
20
10
0
−10
−20
−30
β
0
10
20
30
40
50
t



m=2,3,4, n=1

In [9]:
```
points = [(2.,3.),(3.,4.),(3.,5.)]

gammap = []

for m,n in points:
    gamma = 1/sqrt(1 -(m/n)**2)
    gammap.append((m,n,+gamma))
    gammap.append((m,n,-gamma))

traj_fig = plotpar(gammap)
traj_fig.savefig('PU6b.pdf')
```


Phase
m=2.0 n=3.0 g=1.34
m=2.0 n=3.0 g=-1.34
m=3.0 n=4.0 g=1.51
m=3.0 n=4.0 g=-1.51
m=3.0 n=5.0 g=1.25
m=3.0 n=5.0 g=-1.25
60
40
20
0
−20
−40
−60
β
0
10
20
30
40
50
t



1.2 Kustība pie plaknes

$$\dot{\beta} = 1 - p \sin \beta \dot{x} = - \sin \beta \dot{y} = - \cos \beta$$

In [10]:

```
from scipy.integrate import odeint

def f(r,tau,p):
    beta,x,y = r
    Dx = - sin(beta)
    Dy = - cos(beta)
    Dbeta = 1 - p*sin(beta)
    Dr = [Dbeta, Dx, Dy]
    return Dr
```

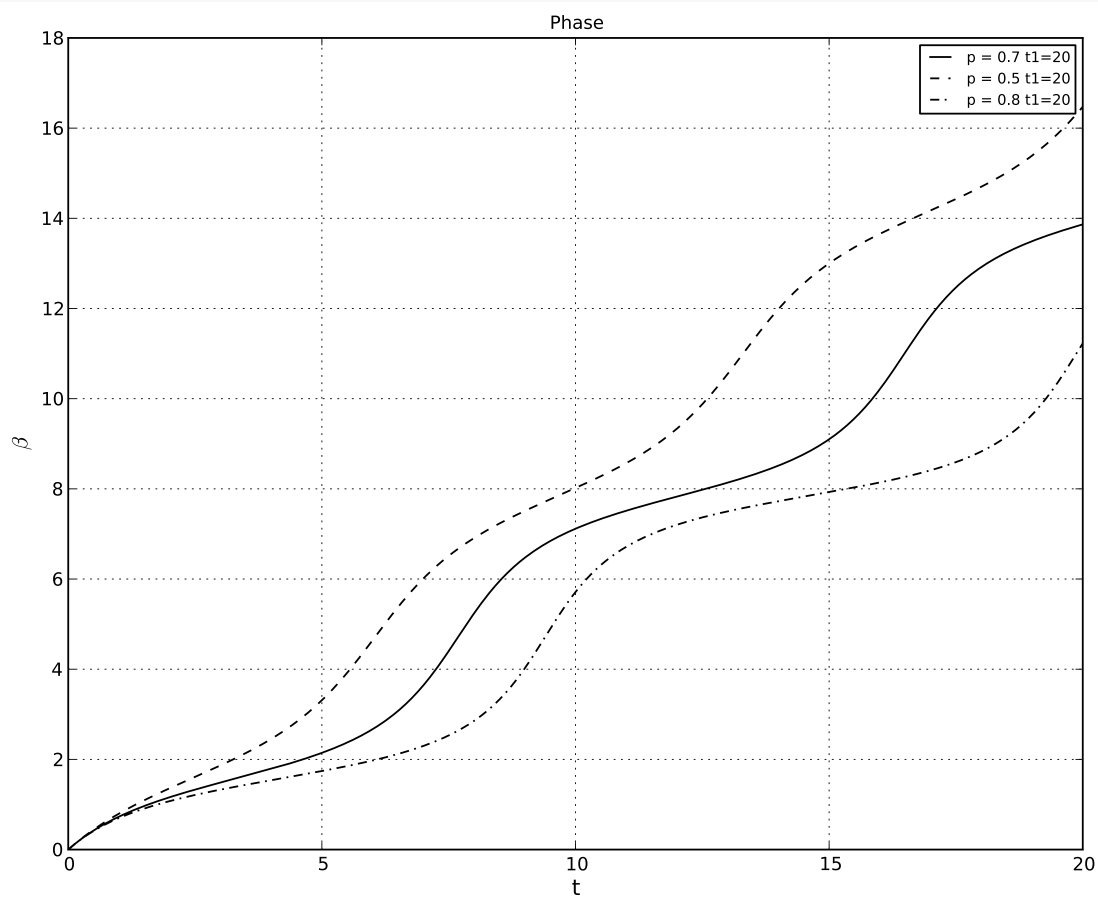
In [11]:

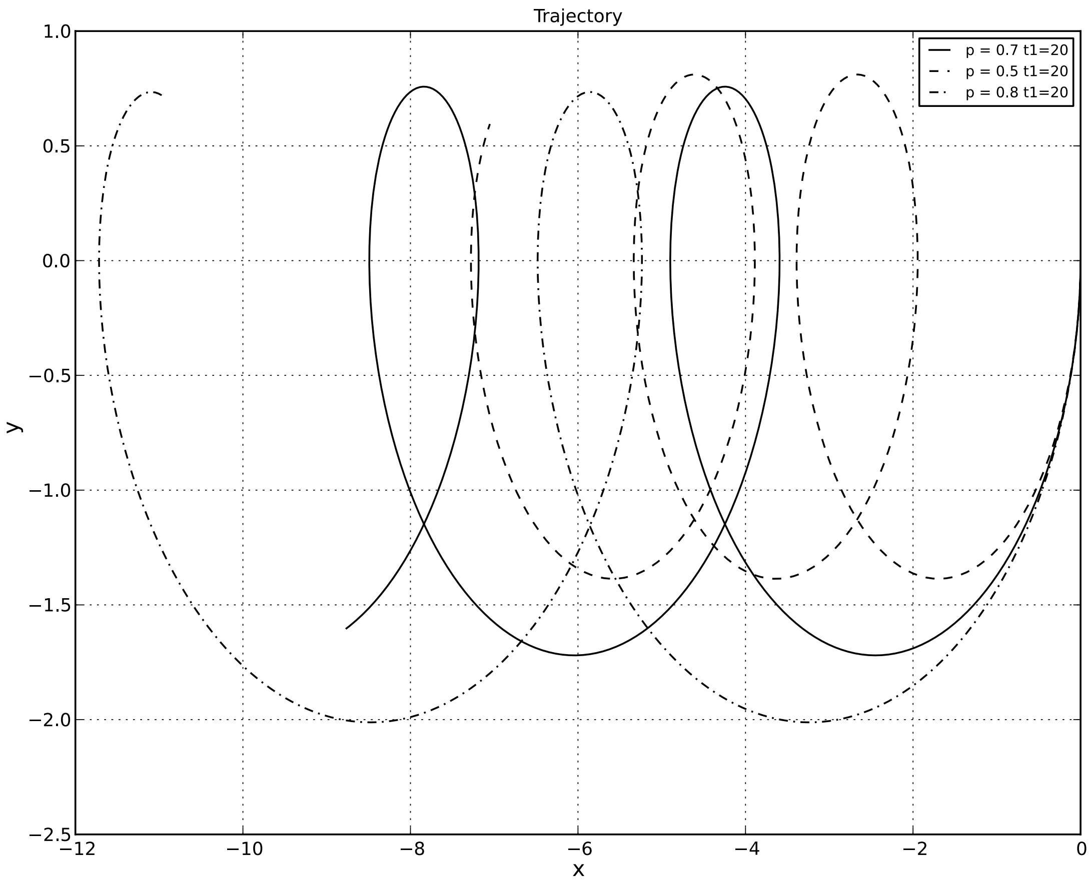
```
def plotpar(ppoints):

    phase_fig = figure('Phase')
    title('Phase')
    xlabel('t')
    ylabel(r'$\beta$')


    traj_fig = figure('Trajectory')
    title('Trajectory')
    xlabel('x')
    ylabel('y')
```

```python
    #gammap.append((1.,1.,1e6))

    for p,t1 in ppoints:

        tpoints = linspace(0,t1,500)
        y0 = [0,0,0]
        data = odeint(f,y0,tpoints,args=(p,))

        figure('Phase')
        plot(tpoints,data[:,0],label='p = {} t1={}'.format(p,t1))

        figure('Trajectory')
        plot(data[:,1],data[:,2],label='p = {} t1={}'.format(p,t1))


    figure('Phase')
    legend(fontsize=7)

    figure('Trajectory')
    legend(fontsize=7)
    return traj_fig
```

In [12]:

```python
p = [0.7,0.5,0.8]
t1 = 20
ppoints = [(pi,t1) for pi in p] # + [(pi,-50) for pi in p]
traject = plotpar(ppoints)
```





1.3 Dipols mainīgā magnētiskā laukā

$$\dot{\beta} = \gamma - \sin\beta\theta = \tau - \beta(\tau)$$

In [13]:

```
from scipy.integrate import odeint

def f(r,tau,p):
    beta = r
    Dbeta = 1 - p*sin(beta)
    Dr = Dbeta
    return Dr
```

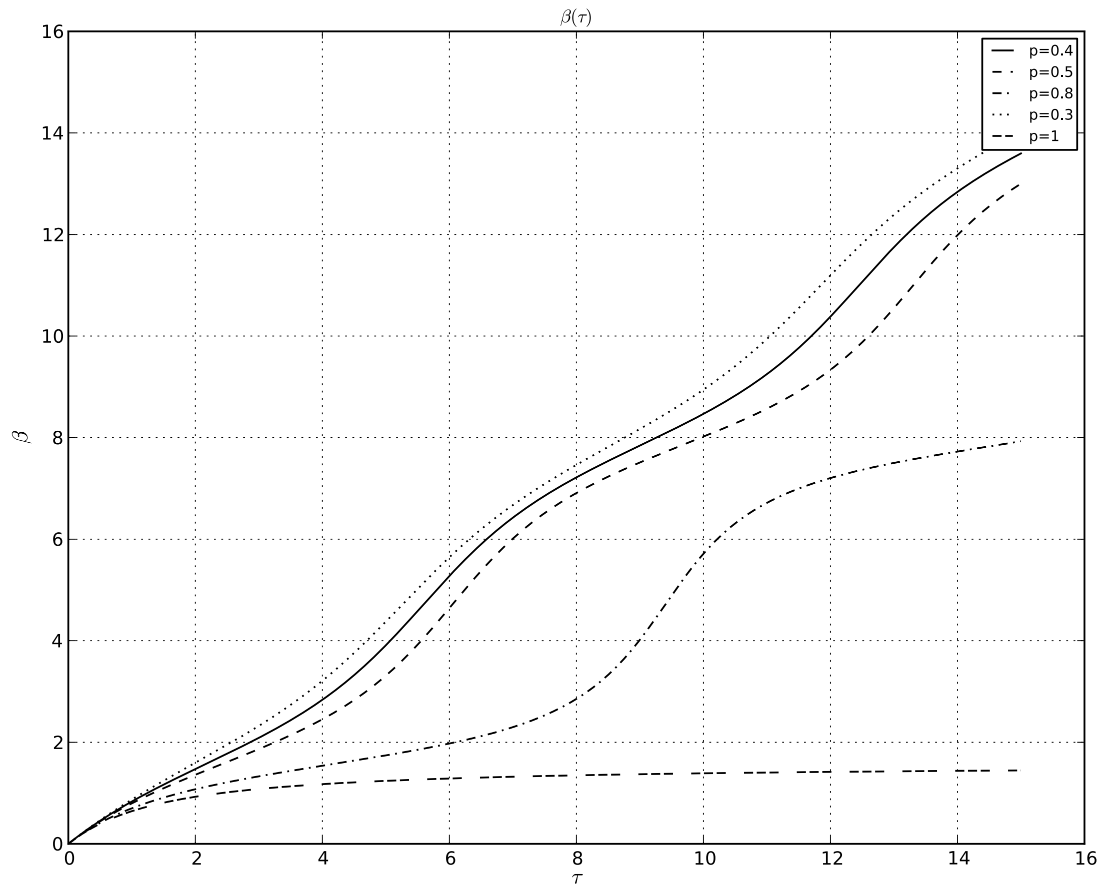
In [14]:

```
def plotpar(ppoints):

    beta_fig = figure('beta')
    title(r'$\beta(\tau)$')
    xlabel(r'$\tau$')
    ylabel(r'$\beta$')


    theta_fig = figure('theta')
    title(r'$\theta(\tau)$')
    xlabel(r'$\tau$')
    ylabel(r'$\theta$')

    tpol_fig = figure('polar')
    #title(r'Polar plot')
```

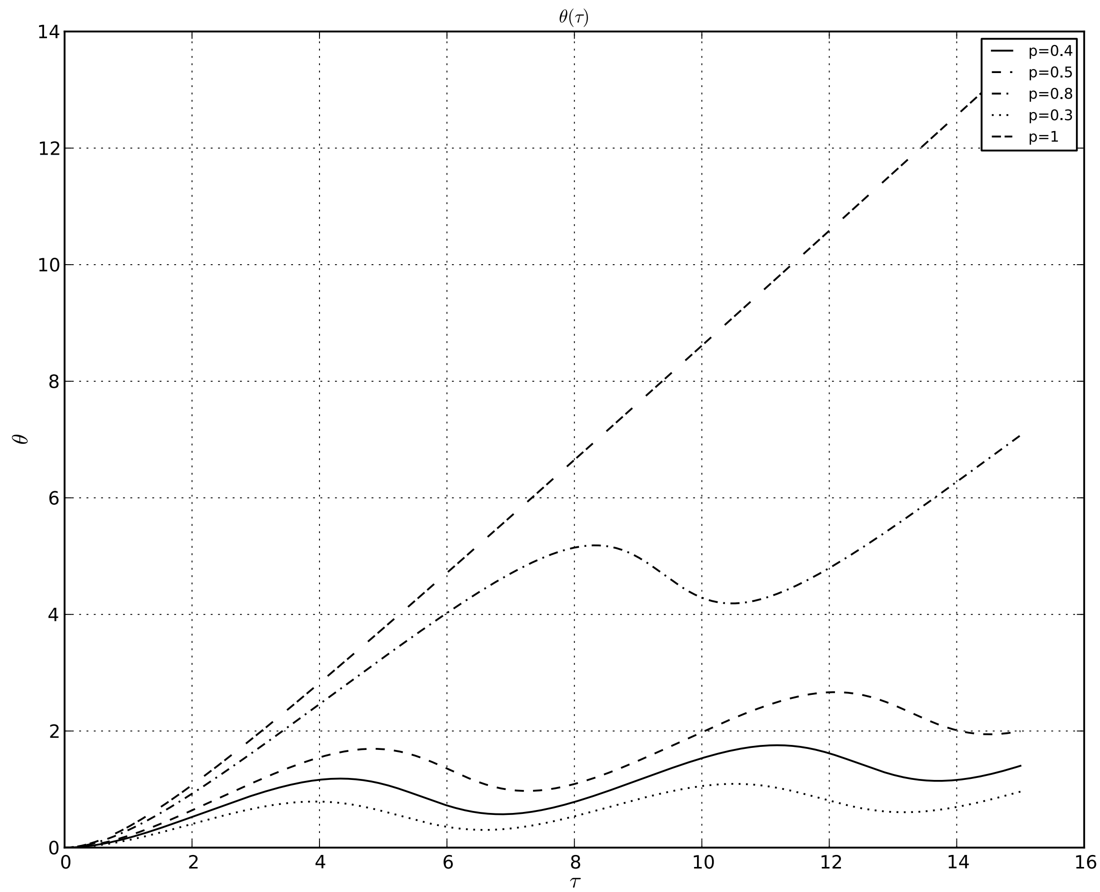
```
    #gammap.append((1.,1.,1e6))

    for p in ppoints:

        tpoints = linspace(0,15,500)
        y0 = 0
        data = odeint(f,y0,tpoints,args=(p,))

        label='p={}'.format(p)
        figure('beta')
        beta = data[:,0]
        plot(tpoints,beta,label=label)

        figure('theta')
        theta = tpoints - beta
        plot(tpoints,theta,label=label)

        figure('polar')
        polar(theta,tpoints,label=label)

    figure('theta')
    legend(fontsize=8)

    figure('beta')
    legend(fontsize=8)

    figure('polar')
    legend(loc=2,fontsize=8)
    #return thetafig
```

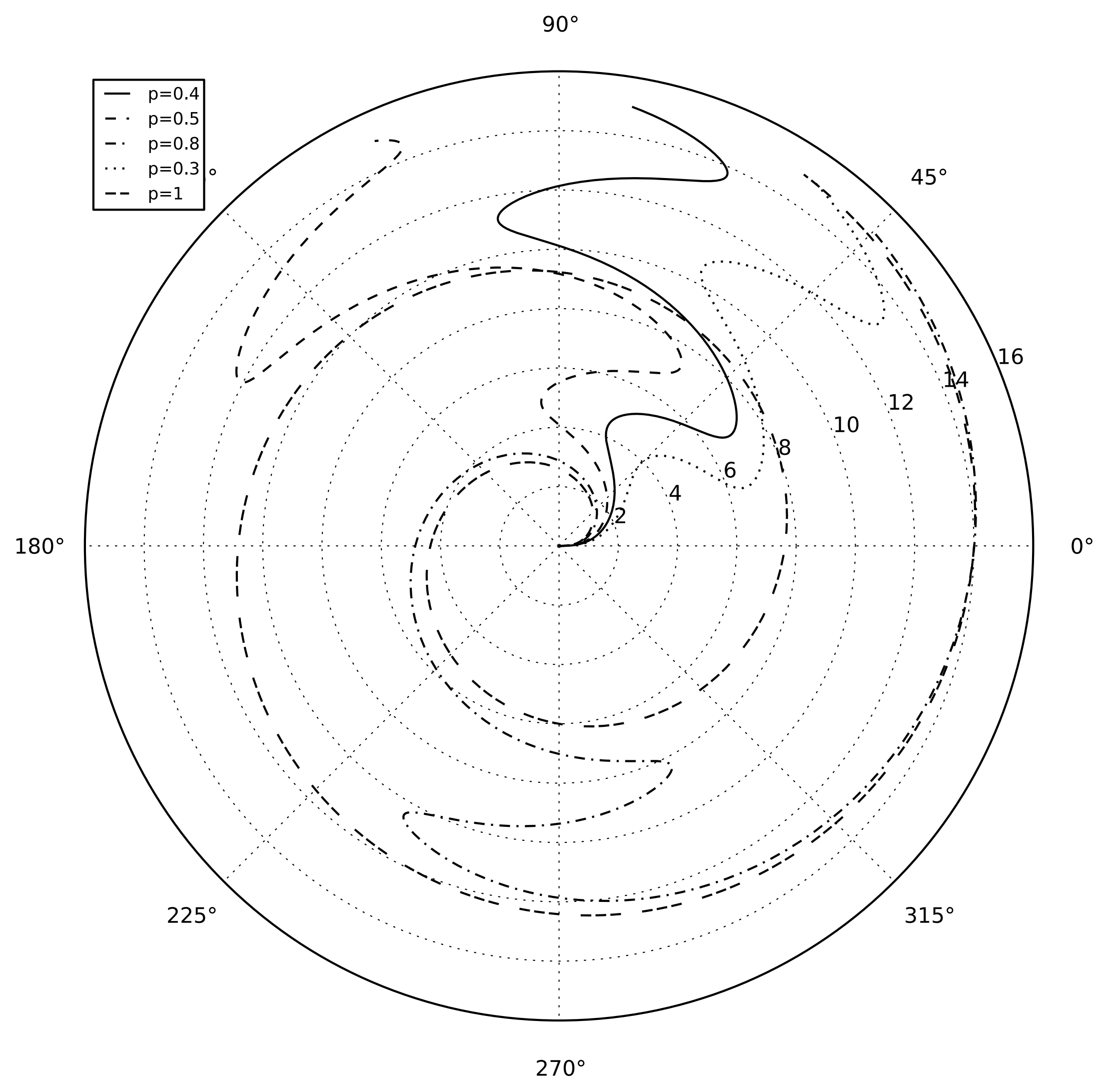
In [15]:

```
ppoints = [0.4,0.5,0.8,0.3,1]
thetafig = plotpar(ppoints)
```



$\beta(\tau)$
p=0.4
p=0.5
p=0.8
p=0.3
p=1
β
τ
0
2
4
6
8
10
12
14
16


$\theta(\tau)$
θ
τ
p=0.4
p=0.5
p=0.8
p=0.3
p=1
0
2
4
6
8
10
12
14
16

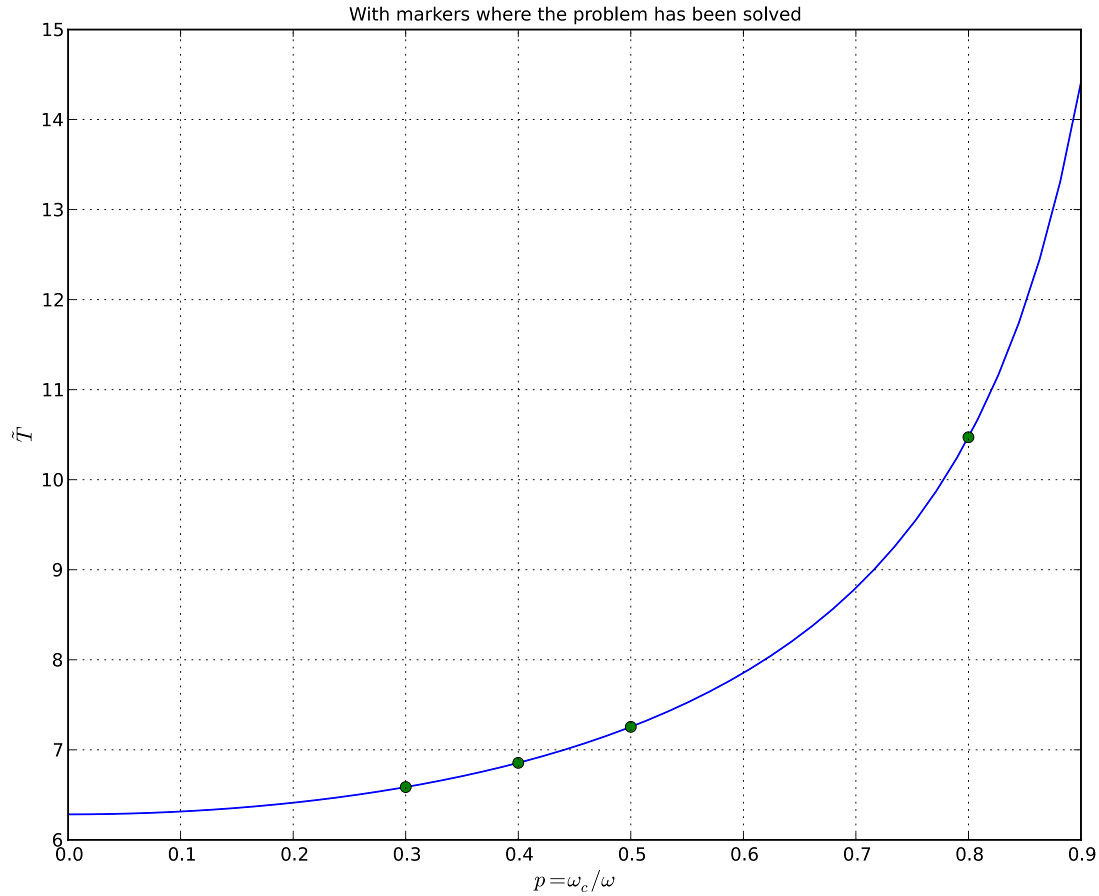


Dipola periods

$$\tilde{T} = \frac{2\pi}{\sqrt{1-p^2}}$$

In [16]:
```
def T(p):
    return 2*pi/sqrt(1 - p**2)

p = linspace(0,0.9)
plot(p,T(p))

plot(ppoints,T(array(ppoints)),'o',label='Computed solutions')
xlabel('$p = \omega_c/\omega$')
ylabel(r'$\tilde T$')

title('With markers where the problem has been solved')
#legend();
```

```
-c:2: RuntimeWarning: divide by zero encountered in true_divide
```



2 18. Oktobris

2.1 Lorenza vienādojumi $\sigma = 10$,$b = 8/3$

In [17]:

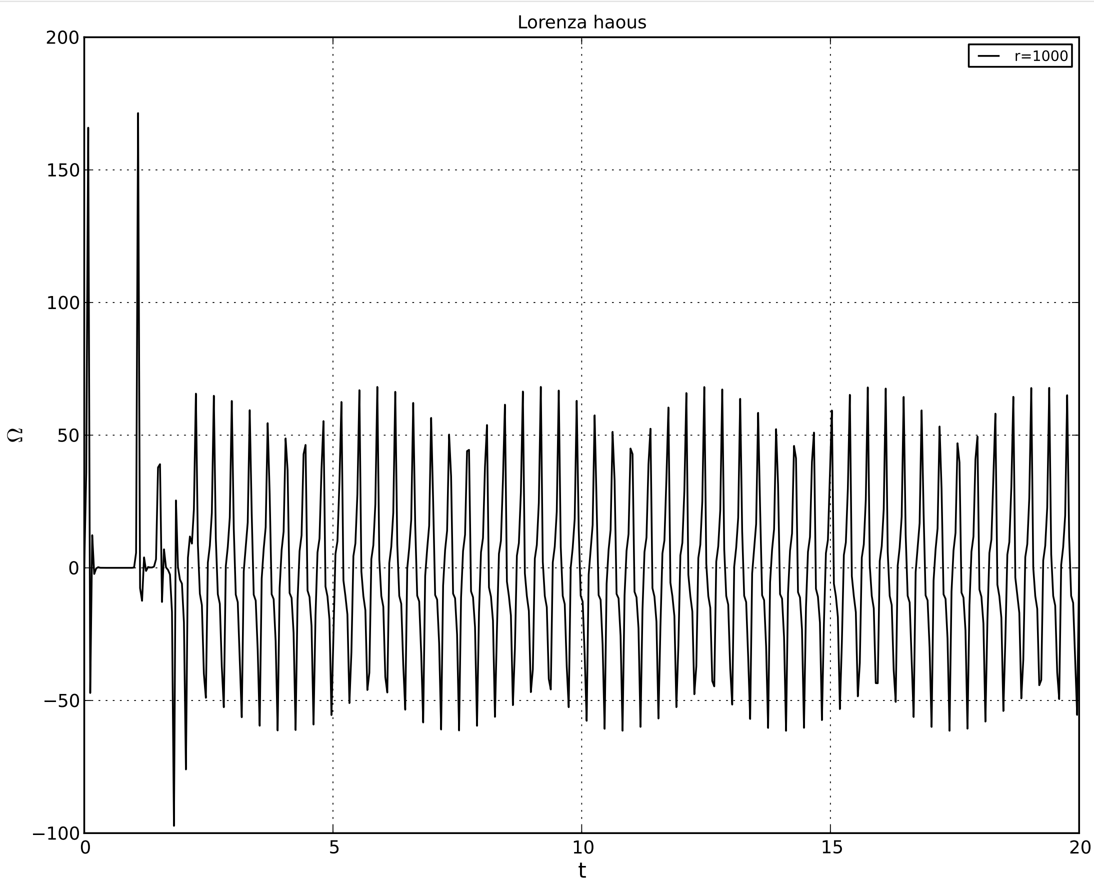
```
from scipy.integrate import odeint

tau = 0.125
taur = 8.2
sigma = taur/ tau
b = 1

def f(r_vect,t,r):
    x,y,z = r_vect
    Dx = -sigma*x + sigma*y
    Dy = -x*z + r*x -y
    Dz = x*y - b*z
    return [Dx,Dy,Dz]
```

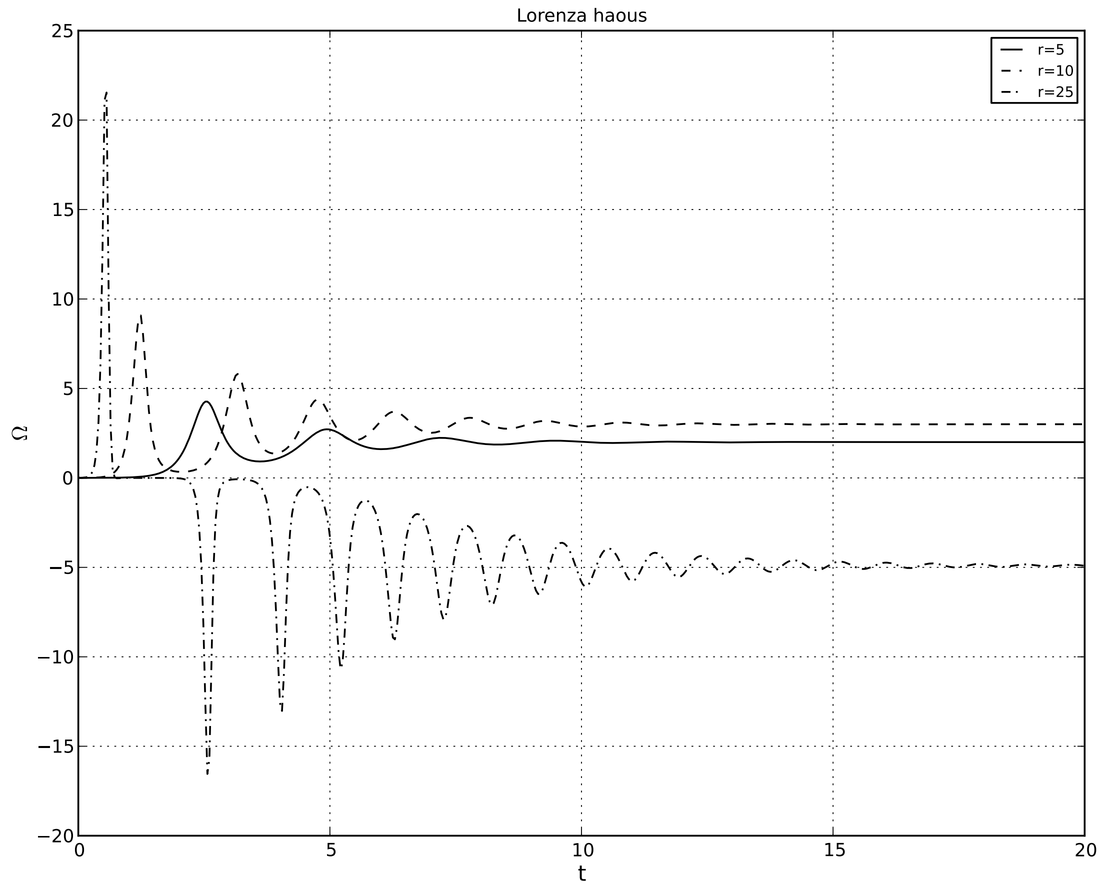
In [18]:

```
def plotlorentz(r_list):

    figure()
    title('Lorenza haous')
    xlabel('t')
    ylabel('$\Omega$')
```

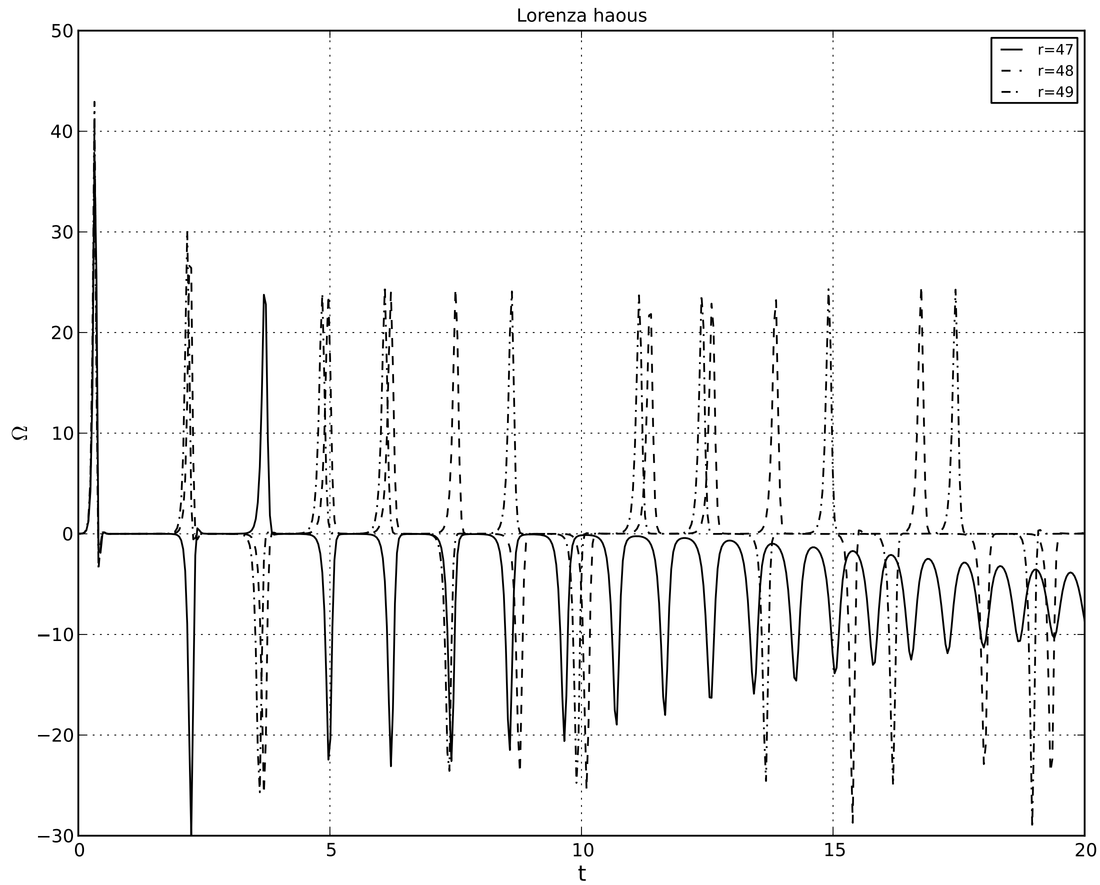
```
    for r in r_list:

        t = linspace(0,20,500)
        y0 = [0.01,0,0]
        data = odeint(f,y0,t,args=(r,))

        x= data[:,0]

        plot(t,x,label='r={}'.format(r))

    legend(fontsize=10)
    #
    #return traj_fig
```

In [19]:
```
plotlorentz([1000])#r_list)
r_list = [5,10,25]
plotlorentz(r_list)

# For chaous

r_list = arange(47,50)
plotlorentz(r_list)
```





Lorenza haous
r=5
r=10
r=25
Ω
t
25
20
15
10
5
0
−5
−10
−15
−20
0
5
10
15
20



3 25. Oktobris

3.1 Lorenza sistēmas stabilitāte

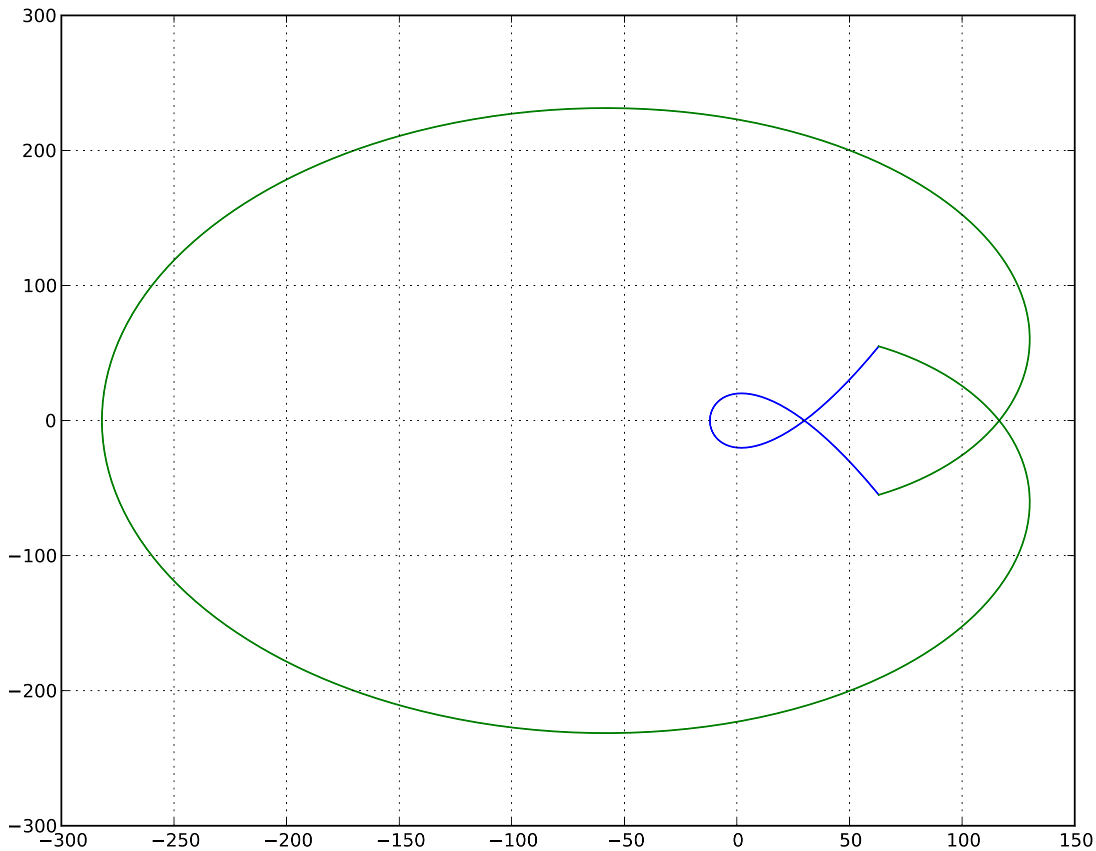
$$\dot{x} = -\sigma(x + y)\dot{y} = -xz - rx - y\dot{z} = xy - bz$$

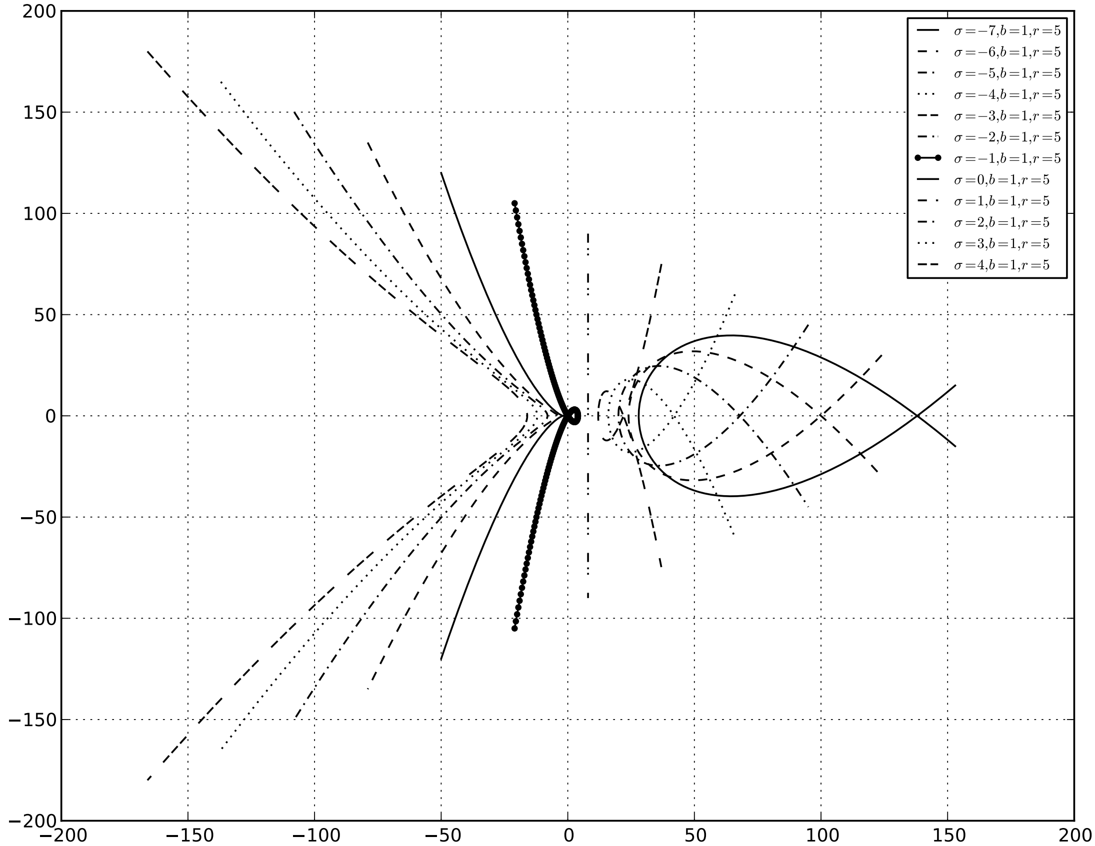
Lai punkts $(x, y, z) = (0, 0, 0)$ būtu stabils sistēmas visām īpašvērtībām ir jābūt negatīvām, kuras atrod atrisinot pret z:

$\Delta = det \begin{bmatrix} \sigma+z & -\sigma & 0 \\ -r & z+1 & 0 \\ 0 & 0 & b+z \end{bmatrix}$

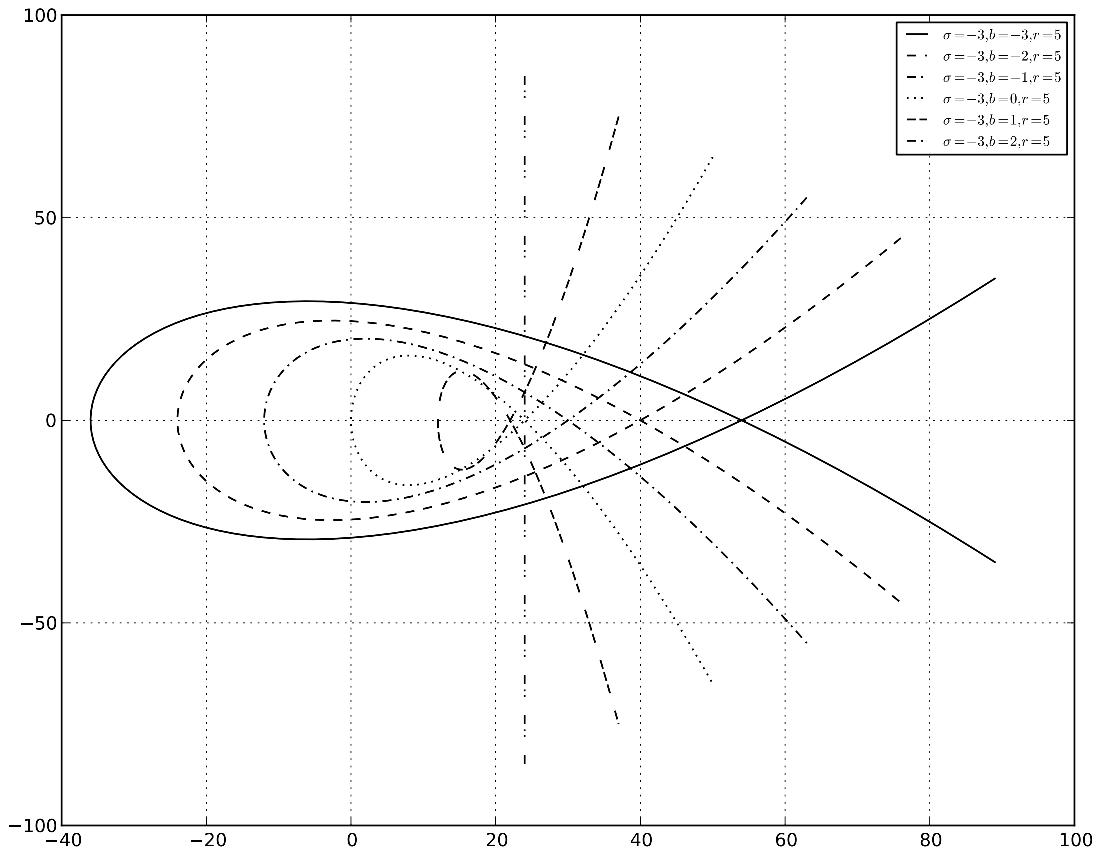
$\Delta = 0$

In [20]:
```
run -i Lorenza_stabilitate.py
```

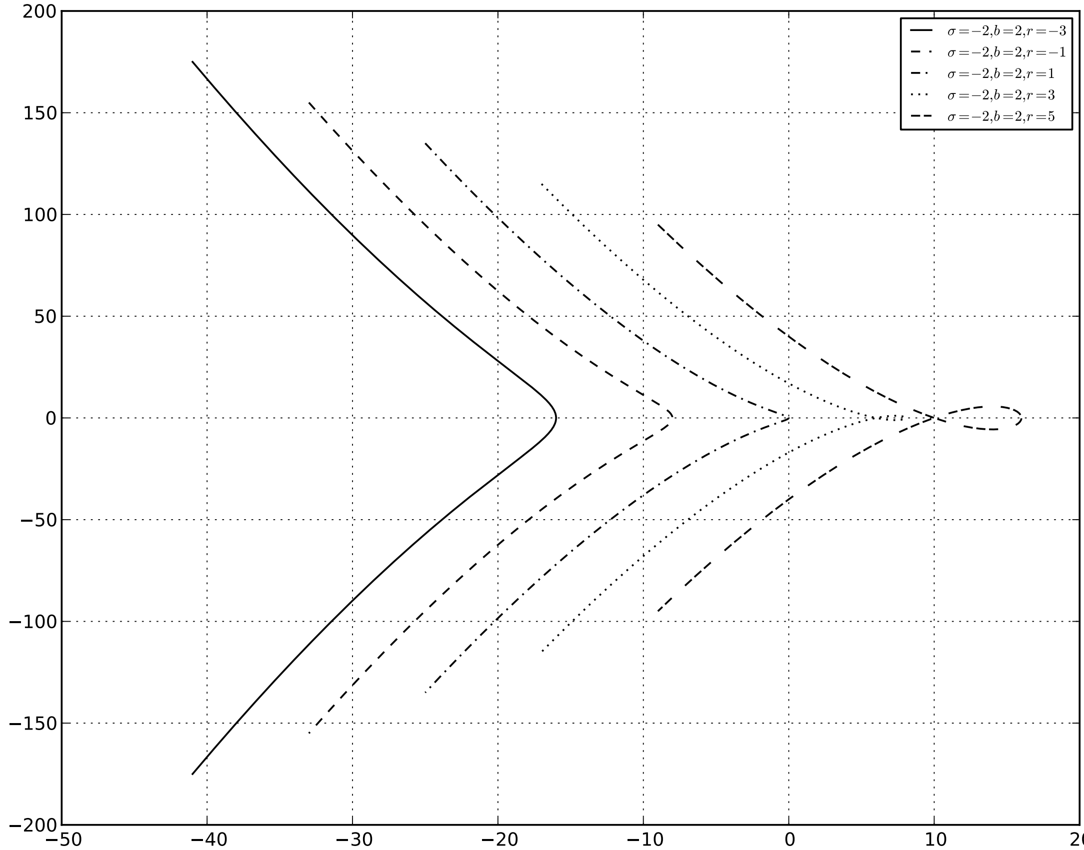

300
200
100
0
−100
−200
−300
−300
−250
−200
−150
−100
−50
0
50
100
150



σ=−7,b=1,r=5
σ=−6,b=1,r=5
σ=−5,b=1,r=5
σ=−4,b=1,r=5
σ=−3,b=1,r=5
σ=−2,b=1,r=5
σ=−1,b=1,r=5
σ=0,b=1,r=5
σ=1,b=1,r=5
σ=2,b=1,r=5
σ=3,b=1,r=5
σ=4,b=1,r=5
200
150
100
50
0
−50
−100
−150
−200
−200
−150
−100
−50
0
50
100
150
200



100
50
0
−50
−100
−40
−20
0
20
40
60
80
100
$\sigma=-3, b=-3, r=5$
$\sigma=-3, b=-2, r=5$
$\sigma=-3, b=-1, r=5$
$\sigma=-3, b=0, r=5$
$\sigma=-3, b=1, r=5$
$\sigma=-3, b=2, r=5$



3.2 Autosvārstību ģenerators

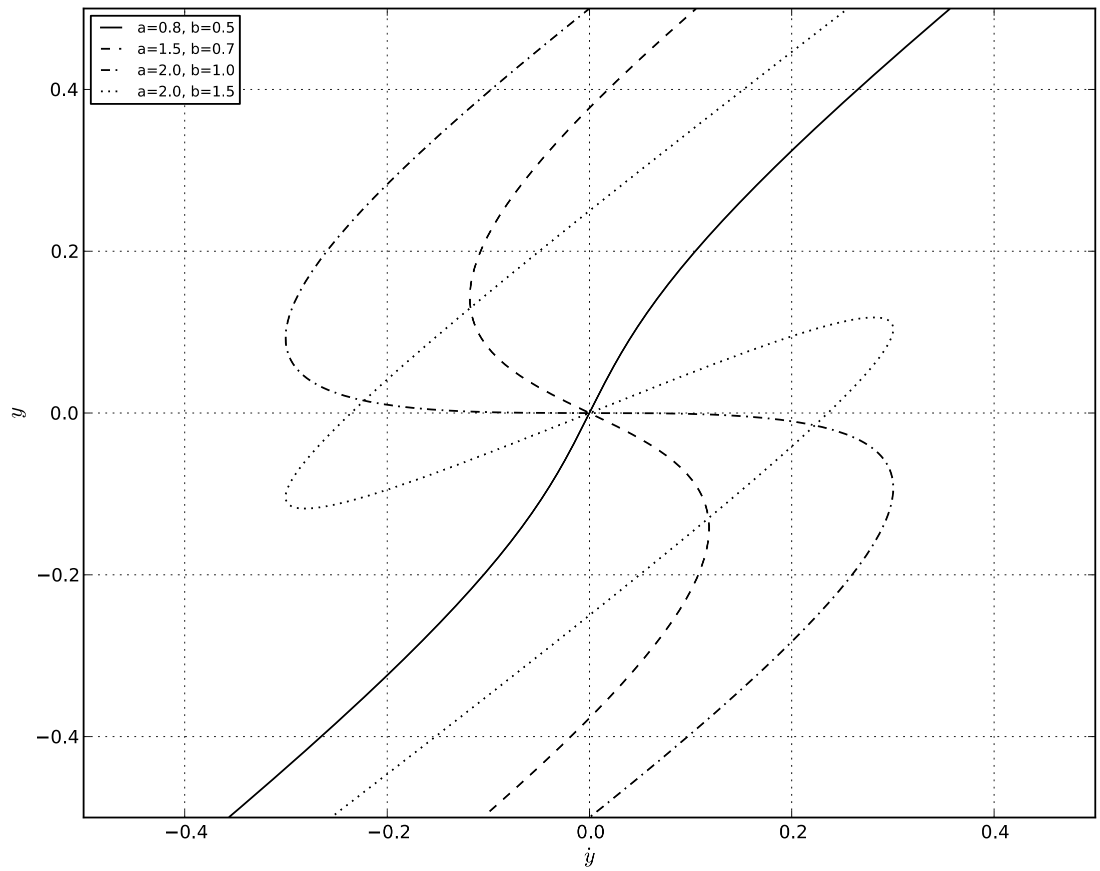
Ja ir negatīvā viskozitāte, tad:

$$\dot{y} = t\left(\frac{a}{t^2+1} - 1\right)$$

$$y = t\left(\frac{b}{t^2+1} - 1\right)$$

kur $a > b$

In [21]:
```
run -i autosvarstibu_gen.py
```



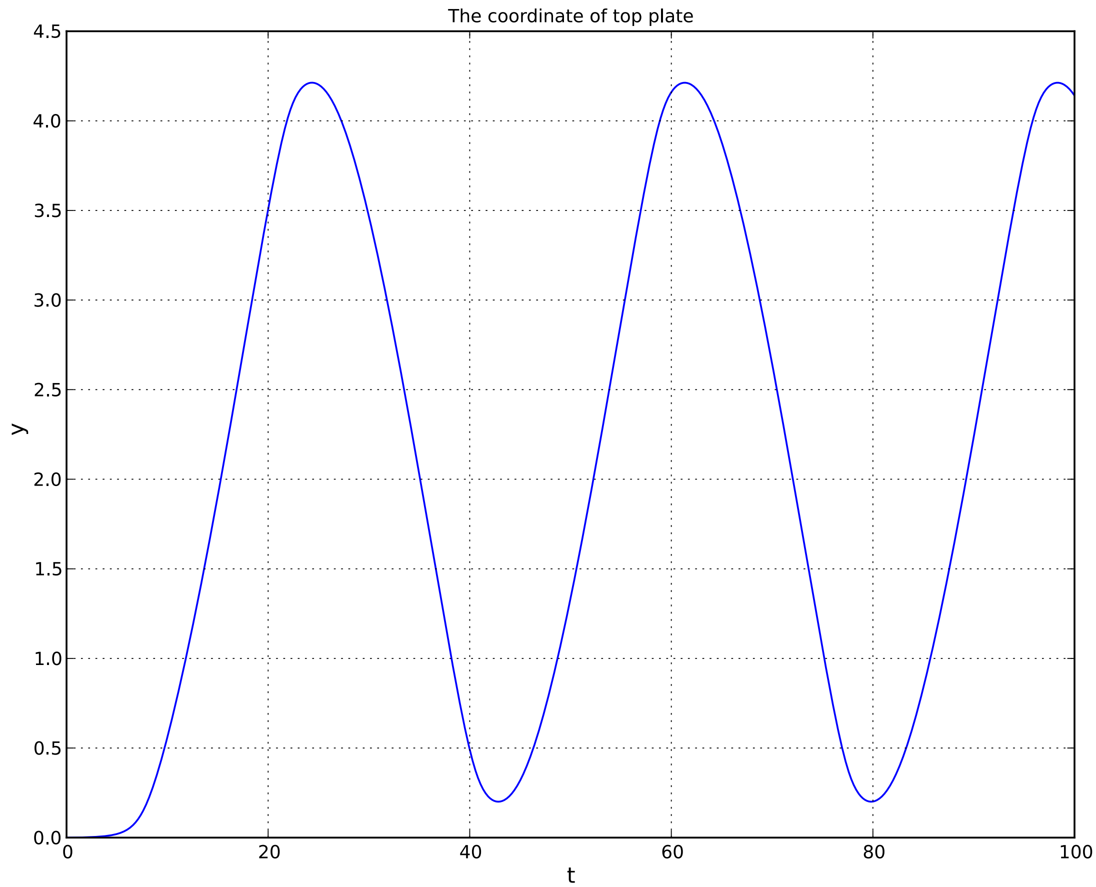
Autosvārstību ģeneratora risinājums

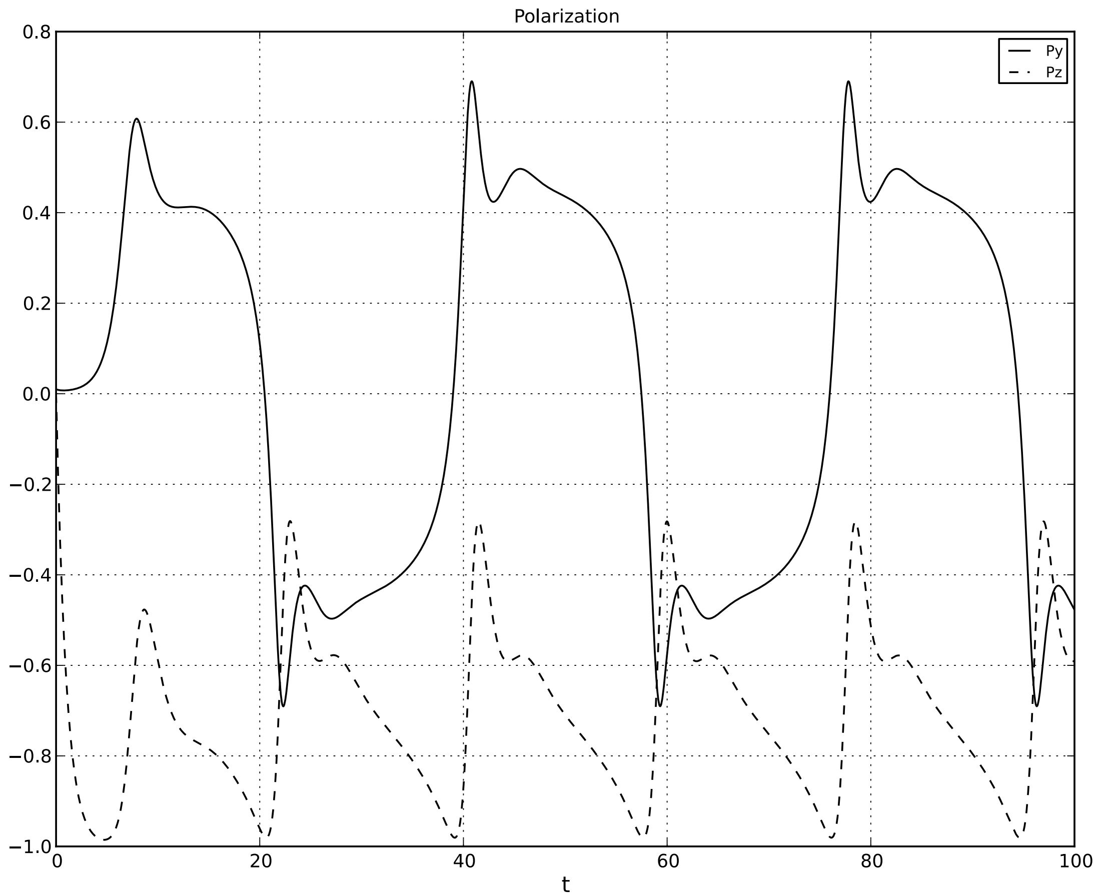
$$\Omega = -\dot{y} + \xi P_y \dot{P}_y = -\Omega P_z - P_y \dot{P}_z = \Omega P_y - P_z - 1\ddot{y} = -\frac{\tau}{\tau_r}(\dot{y} + \xi\gamma P_y) - \omega_0^2\tau^2 y$$

Pieņemot, ka $\omega_0 \to 0$ un $x = \dot{y}$, tad pēdējais vienādojums:

$$\dot{y} = x\dot{x} = -\frac{\tau}{\tau_r}(x + \xi\gamma P_y) - \omega_0^2\tau^2 y$$

In [22]:
```
run -i autosvarstibu_gen_ode.py
```


The coordinate of top plate
y
t
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
0
20
40
60
80
100



4 9. Novembris

4.1 A project about linear systems [5.2.14]

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \; a, b, c, d \in [-1, 1]$$

Uniform distribution

In [23]:
```
from numpy.random import rand

N = 10000000

a = 2*rand(N) - 1
b = 2*rand(N) - 1
c = 2* rand(N) -1
d = 2*rand(N) - 1

Det = a*d - b*c
Det_prob = len(Det[Det>0])/N
print('Probability for Det>0: {}'.format(Det_prob))

tau = a + d
tau_prob = len(tau[tau>0])/N
```

```
print('Probability for tau>0: {}'.format(tau_prob))

D = tau**2 - 4*Det
D_prob = len(D[D>0])/N
print('Probability for D>0: {}'.format(D_prob))
```

```
Probability for Det>0: 0.5000334
Probability for tau>0: 0.5001687
Probability for D>0: 0.6806914
```

Normal distribution

In [24]:

```
from numpy.random import randn

N = 10000000

a = randn(N)
b = randn(N)
c = randn(N)
d = randn(N)

Det = a*d - b*c
Det_prob = len(Det[Det>0])/N
print('Probability for Det>0: {}'.format(Det_prob))

tau = a + d
tau_prob = len(tau[tau>0])/N
print('Probability for tau>0: {}'.format(tau_prob))

D = tau**2 - 4*Det
D_prob = len(D[D>0])/N
print('Probability for D>0: {}'.format(D_prob))
```

```
Probability for Det>0: 0.4998732
Probability for tau>0: 0.4999786
Probability for D>0: 0.707114
```

Conclusion

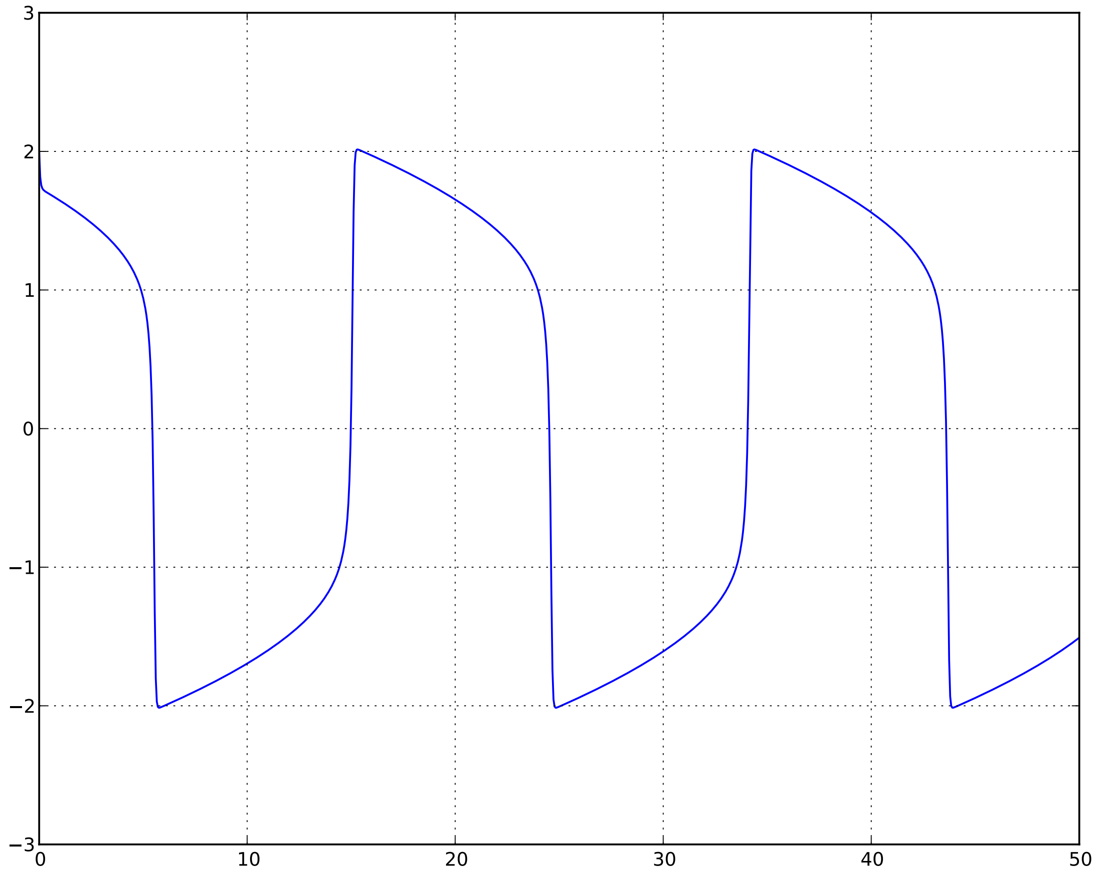
- There are 50 % chance that it will be saddle point
- 15 % for stable and unstable nodes
- 35 % for stable and unstable spirals
- Probability does not changes significantly if numbers are taken from normal distribution instead.

5 23. Novembris

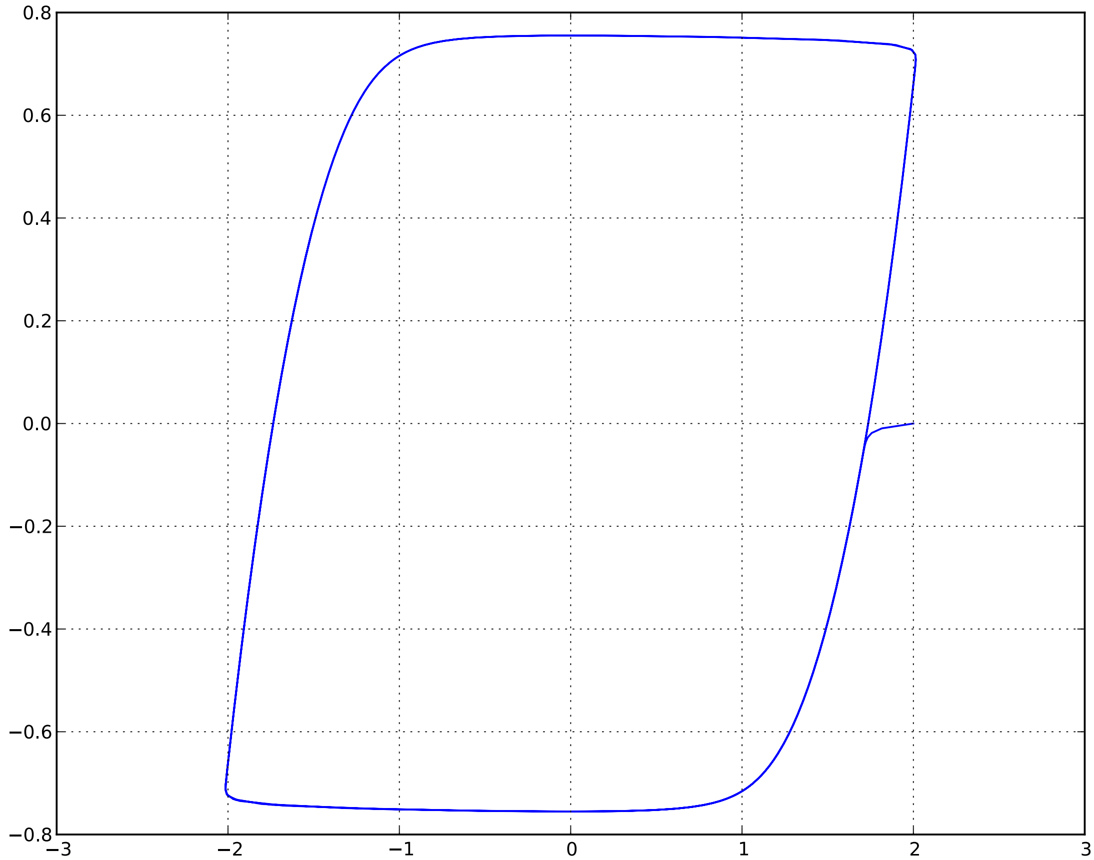
5.1 Eksperiments ar Vander-Pola osciliatoru, lai novērtētu skaitlisko risinātāju

In [25]:

```
run -i van_der_Pol.py
```



3
2
1
0
−1
−2
−3
0
10
20
30
40
50



6 19. Decembris

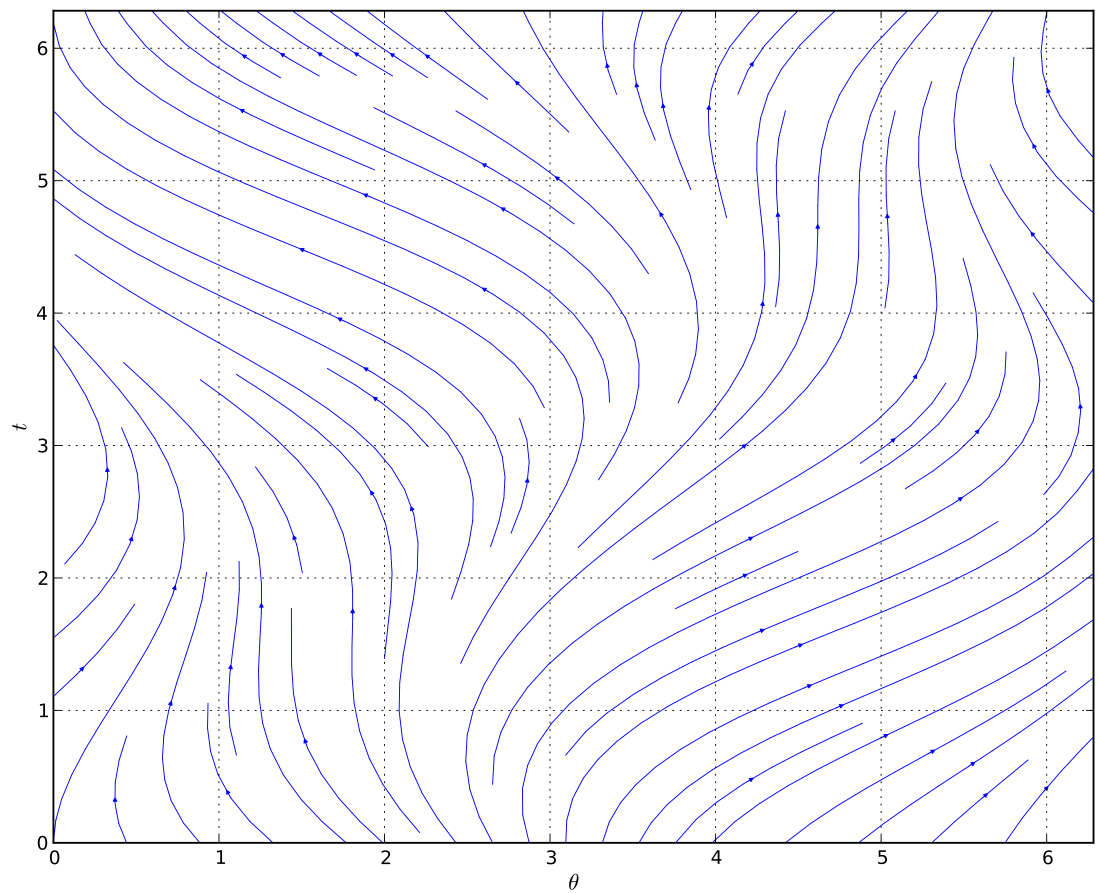
6.1 Another driven overdamped system [8.7.7]

In [26]:

```
def f(theta,tau):

    Dtheta = sin(tau) - sin(theta)
    Dtau = ones_like(Dtheta)

    return [Dtheta,Dtau]

t = linspace(0,2*pi)
theta = linspace(0,2*pi)

#Dtheta = sin(t[:,newaxis]) - sin(theta)
#Dt = ones_like(Dtheta)
Dtheta,Dt = f(theta,t[:,newaxis])

fig = figure()
streamplot(theta,t,Dtheta,Dt,linewidth=0.5,arrowsize=0.5)
#quiver(theta,t,Dtheta,Dt)
xlabel(r'$\theta$')
ylabel('$t$')
```

Out [26]:

```
<matplotlib.text.Text at 0xb00b602c>
```



7 20. Decembris

7.1 10.1.[1,8]

In [27]:

```
def explore(f_str):
    def f(x):
        try:
            xp = eval(f_str)

            if abs(xp)>100: raise ValueError('large')
            return xp
        except:
            return nan

    x = [-2,-1.1,-0.5,0,0.5,1.1,2]

    print(f_str)

    for n in range(10):
        print('\t' + '\t'.join(['{:.3f}'.format(xi) for xi in x]))
        x = [f(xi) for xi in x]

    print()
```

In [28]:
```
functions = ['sqrt(x)',
             'x**3',
             'exp(x)',
             'log(x)',
             'sinh(x)',
             'tanh(x)',
             '1/tan(x)',
             'tan(x)'
             ]
for fi in functions: explore(fi)
```

```
sqrt(x)
        -2.000  -1.100  -0.500  0.000   0.500   1.100   2.000
        nan     nan     nan     0.000   0.707   1.049   1.414
        nan     nan     nan     0.000   0.841   1.024   1.189
        nan     nan     nan     0.000   0.917   1.012   1.091
        nan     nan     nan     0.000   0.958   1.006   1.044
        nan     nan     nan     0.000   0.979   1.003   1.022
        nan     nan     nan     0.000   0.989   1.001   1.011
        nan     nan     nan     0.000   0.995   1.001   1.005
        nan     nan     nan     0.000   0.997   1.000   1.003
        nan     nan     nan     0.000   0.999   1.000   1.001

x**3
        -2.000  -1.100  -0.500  0.000   0.500   1.100   2.000
        -8.000  -1.331  -0.125  0.000   0.125   1.331   8.000
        nan     -2.358  -0.002  0.000   0.002   2.358   nan
        nan     -13.110 -0.000  0.000   0.000   13.110  nan
        nan     nan     -0.000  0.000   0.000   nan     nan
        nan     nan     -0.000  0.000   0.000   nan     nan
        nan     nan     -0.000  0.000   0.000   nan     nan
        nan     nan     -0.000  0.000   0.000   nan     nan
        nan     nan     -0.000  0.000   0.000   nan     nan
        nan     nan     -0.000  0.000   0.000   nan     nan

exp(x)
        -2.000  -1.100  -0.500  0.000   0.500   1.100   2.000
        0.135   0.333   0.607   1.000   1.649   3.004   7.389
        1.145   1.395   1.834   2.718   5.200   20.169  nan
        3.142   4.035   6.259   15.154  nan     nan     nan
        23.155  56.534  nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan

log(x)
        -2.000  -1.100  -0.500  0.000   0.500   1.100   2.000
        nan     nan     nan     nan     -0.693  0.095   0.693
        nan     nan     nan     nan     nan     -2.351  -0.367
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
        nan     nan     nan     nan     nan     nan     nan
```

```
        nan     nan     nan     nan     nan     nan     nan

sinh(x)
        -2.000  -1.100  -0.500  0.000   0.500   1.100   2.000
        -3.627  -1.336  -0.521  0.000   0.521   1.336   3.627
        -18.784 -1.770  -0.545  0.000   0.545   1.770   18.784
        nan     -2.849  -0.572  0.000   0.572   2.849   nan
        nan     -8.610  -0.604  0.000   0.604   8.610   nan
        nan     nan     -0.642  0.000   0.642   nan     nan
        nan     nan     -0.687  0.000   0.687   nan     nan
        nan     nan     -0.742  0.000   0.742   nan     nan
        nan     nan     -0.812  0.000   0.812   nan     nan
        nan     nan     -0.904  0.000   0.904   nan     nan

tanh(x)
        -2.000  -1.100  -0.500  0.000   0.500   1.100   2.000
        -0.964  -0.800  -0.462  0.000   0.462   0.800   0.964
        -0.746  -0.664  -0.432  0.000   0.432   0.664   0.746
        -0.633  -0.581  -0.407  0.000   0.407   0.581   0.633
        -0.560  -0.524  -0.386  0.000   0.386   0.524   0.560
        -0.508  -0.480  -0.368  0.000   0.368   0.480   0.508
        -0.468  -0.447  -0.352  0.000   0.352   0.447   0.468
        -0.437  -0.419  -0.338  0.000   0.338   0.419   0.437
        -0.411  -0.396  -0.326  0.000   0.326   0.396   0.411
        -0.389  -0.377  -0.315  0.000   0.315   0.377   0.389

1/tan(x)
        -2.000  -1.100  -0.500  0.000   0.500   1.100   2.000
        0.458   -0.509  -1.830  nan     1.830   0.509   -0.458
        2.030   -1.792  0.266   nan     -0.266  1.792   -2.030
        -0.495  0.225   3.675   nan     -3.675  -0.225  0.495
        -1.853  4.369   1.694   nan     -1.694  -4.369  1.853
        0.290   0.357   -0.124  nan     0.124   -0.357  -0.290
        3.350   2.681   -8.014  nan     8.014   -2.681  -3.350
        4.723   -2.013  0.162   nan     -0.162  2.013   -4.723
        -0.011  0.473   6.131   nan     -6.131  -0.473  0.011
        -92.669 1.952   -6.518  nan     6.518   -1.952  92.669

tan(x)
        -2.000  -1.100  -0.500  0.000   0.500   1.100   2.000
        2.185   -1.965  -0.546  0.000   0.546   1.965   -2.185
        -1.418  2.406   -0.608  0.000   0.608   -2.406  1.418
        -6.491  -0.906  -0.696  0.000   0.696   0.906   6.491
        -0.210  -1.275  -0.835  0.000   0.835   1.275   0.210
        -0.214  -3.283  -1.105  0.000   1.105   3.283   0.214
        -0.217  -0.143  -1.992  0.000   1.992   0.143   0.217
        -0.220  -0.144  2.234   0.000   -2.234  0.144   0.220
        -0.224  -0.145  -1.280  0.000   1.280   0.145   0.224
        -0.228  -0.146  -3.346  0.000   3.346   0.146   0.228

van_der_Pol.py:1: RuntimeWarning: invalid value encountered in sqrt

van_der_Pol.py:1: RuntimeWarning: invalid value encountered in log
```

```
van_der_Pol.py:1: RuntimeWarning: divide by zero encountered in log

van_der_Pol.py:1: RuntimeWarning: divide by zero encountered in
double_scalars
```

7.2 Orbit diagram

In [29]:
```
%config InlineBackend.figure_format = 'png'
```

In [30]:
```
def orbit_map(f,rs,x0):
    ys = []
    #rs = linspace(0, 4, 400)

    for r in rs:
        x = x0#0.1 # Initial condition
        for i in range(300):
            x = f(x, r)

        for i in range(300):
            x = f(x, r)
            ys.append([r, x])

    ys = array(ys)

    plot(ys[:,0], ys[:,1], '.',markersize=0.2)
    #plot(ys[:,1], '.')
    #return ys
```

10.2.3

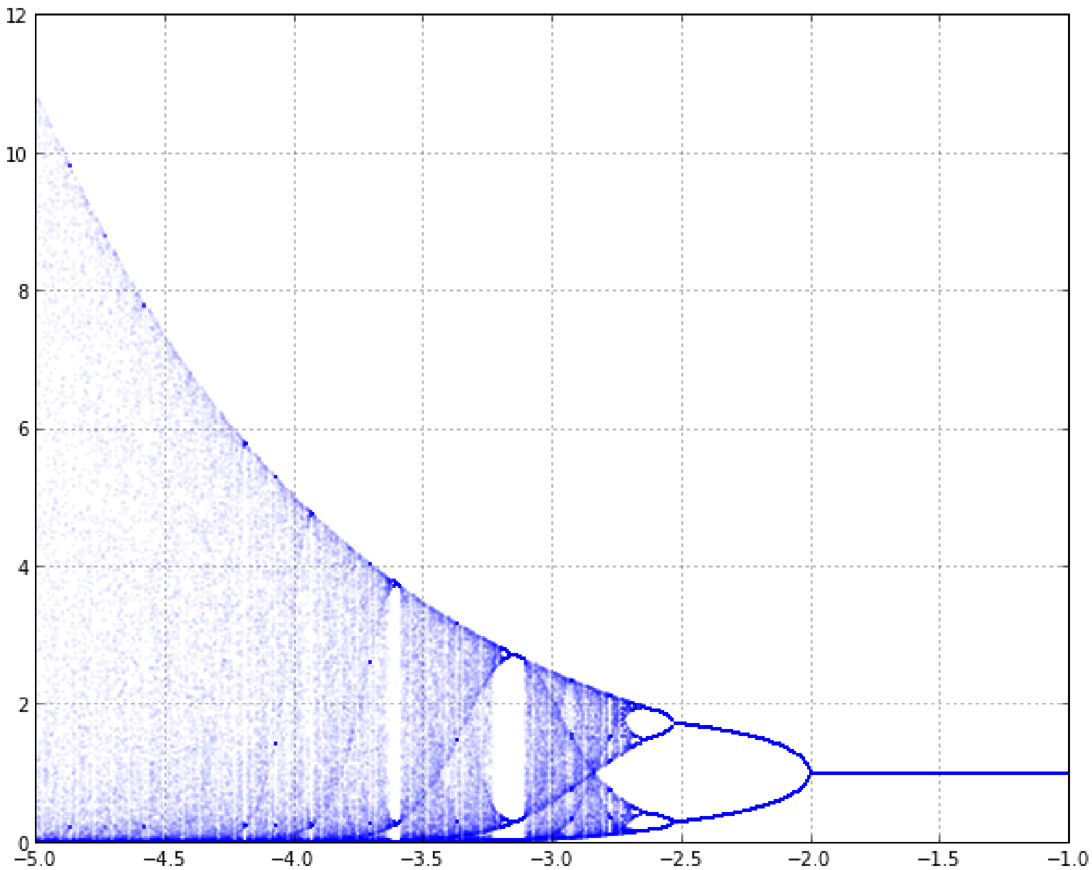
$x_{n+1} = x_n e^{-r(1-x_n)}$

In [31]:
```
x = 0.5#0.1 # Initial condition
r = -5

def f(x,r): return x*exp(-r*(1-x)) #r*x*(1-x)
#def f(x,r): return r*x*(1-x)


values = ''
for i in range(300):
    x = f(x, r)

for i in range(10):
    x = f(x, r)
    values += '{:.5g} \n'.format(x)

print(values)
```

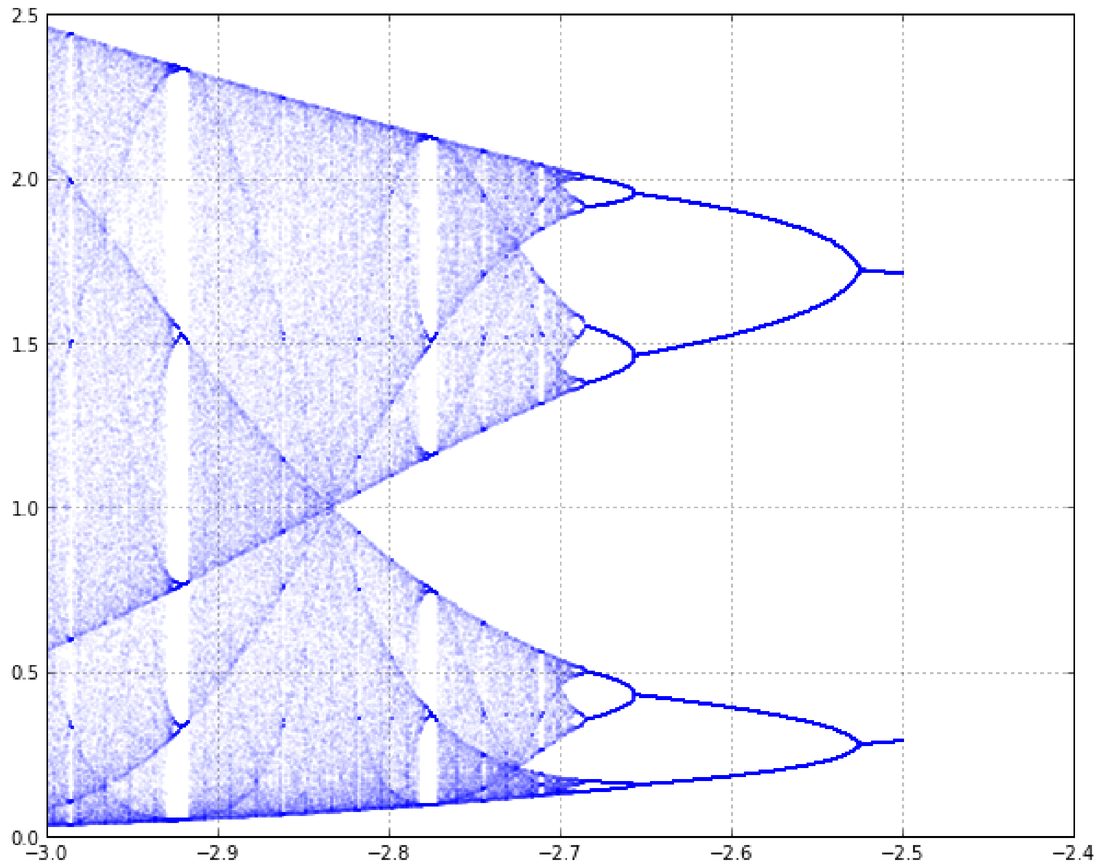
```
9.7115e-16
1.4413e-13
2.1391e-11
3.1747e-09
4.7117e-07
6.9927e-05
0.010375
1.4619
0.14519
```

10.426

In [32]:
```
rs = linspace(-5,-1,500)
orbit_map(f,rs,0.5)
```



In [33]:
```
rs = linspace(-3,-2.5,500)
orbit_map(f,rs,0.5)
```



7.3 10.2.6

$x_{n+1} = r\cos x_n$

In [34]:
```
x = 0.5#0.1 # Initial condition
r = 1.5

def f(x,r): return r*cos(x)#r*x*(1-x)
#def f(x,r): return r*x*(1-x)


values = ''
for i in range(300):
    x = f(x, r)

for i in range(10):
    x = f(x, r)
    values += '{:.5g} \n'.format(x)

print(values)
```

```
1.4887
0.12308
1.4887
0.12308
1.4887
0.12308
1.4887
0.12308
```

```
1.4887
0.12308
```

In [35]:
```
rs = linspace(1.1,10,500)
orbit_map(f,rs,0.5)
```