

Driven, damped, pendulum

Note: The notation and graphs in this notebook parallel those in *Chaotic Dynamics* by Baker and Gollub. (There's a copy in the department office.)

For the driven, damped, pendulum, Newton's second law gives

$$\frac{d^2\theta}{dt^2} = -\omega_0^2 \sin \theta - \gamma \frac{d\theta}{dt} + A \cos(\omega_d t),$$

where ω_0 is the natural frequency of the low-amplitude, undamped, and undriven pendulum, and ω_d is the frequency of the drive. Using dimensionless variables in which time is measured in units of ω_0^{-1} , i.e., $t' = \omega_0 t$, this equation reduces to

$$\frac{d^2\theta}{dt'^2} = -\sin \theta - b \frac{d\theta}{dt'} + g \cos(\omega'_d t'),$$

where $b \equiv \gamma/\omega_0$, $\omega'_d \equiv \omega_d/\omega_0$, and $g \equiv A/\omega_0^2$. In these units the period is 2π . (This is essentially the parametrization of Baker and Gollub.) From this point on I will drop the primes, but retain the understanding that these are nondimensional quantities.

Before doing the numerical integration I will break this up into two coupled first order equations:

$$\begin{aligned} \frac{d\theta}{dt} &= \omega \\ \frac{d\omega}{dt} &= -b\omega - \sin \theta + g \cos(\omega_d t) \end{aligned}$$

These equations will be entered in the function `eqs()` below.

```
In [1]: import scipy as sp
from scipy import integrate # not included in basic scipy

import matplotlib.pyplot as plt
# ML finds plots too big, thus the customizations:
plt.rcParams['figure.figsize'] = (6,4.5) # Change default size of plots
plt.rcParams['font.size'] = 14 # Change default fontsize for figures
plt.rcParams['figure.autolayout'] = True # Adjusts for changes

# Following is an Ipython magic command that puts figures in the notebook.
# For figures in separate windows, remove this command.
%matplotlib notebook
```

Differential Equations

```
In [2]: # Function returning derivatives of the dependent quantities u[0]
# and u[1], or more physically in this case, theta and omega.
def eqs(u,t,b,c,om_d):
    th = u[0]
    om = u[1]
    return (om, -sp.sin(th) - b*om + c*sp.cos(om_d*t))
```

Example of period doubling

See Golub and Baker Fig. 3.4b

Parameters

These parameters result in period doubling; see graphs below.

```
In [3]: th0 = 0.1      # Initial theta
om0 = 0              # Initial angular velocity
u0 = sp.array([th0,om0])# Combine initial conditions in array
b = 0.5              # Damping parameter
g = 1.07             # Driving amplitude
om_d = 2./3          # Driving frequency
```

Pick times for output, and solve DEs:

Note: odeint returns an array:

```
[[th_0  om_0],
 [th_1  om_1],
 [th_2  om_2], ... ]
```

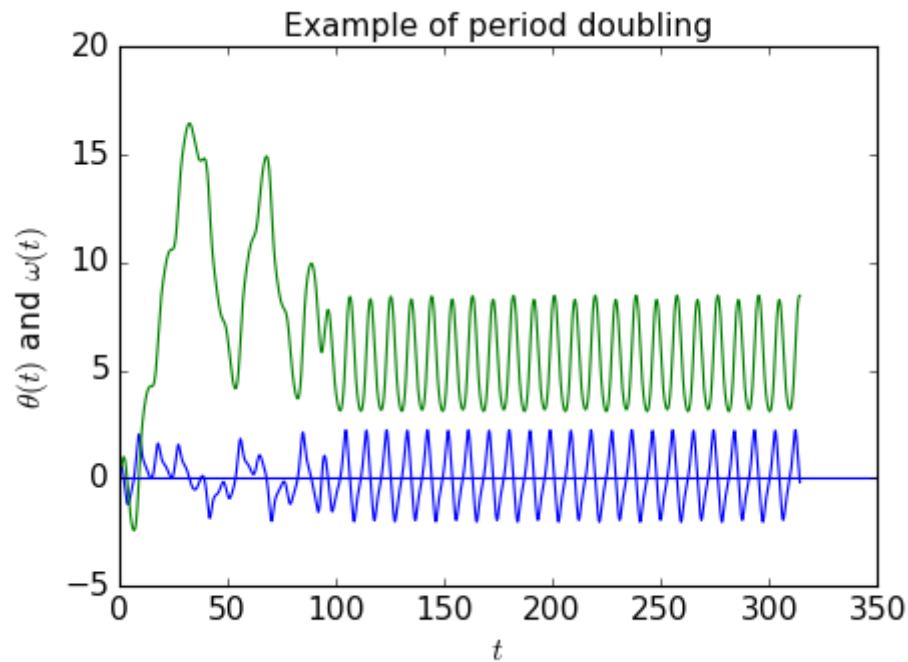
To get single list for th and single list for om we need the transpose of the returned array.
(Could also keep return as a single array if that's more useful down the road.)

```
In [4]: # NOTE: The time points t selected for the output are not the
# points used for the numerical evaluation.
t = sp.linspace(0,50*2.*sp.pi,50*201)

th, om = sp.integrate.odeint(eqs,u0,t,args=(b,g,om_d)).T
```

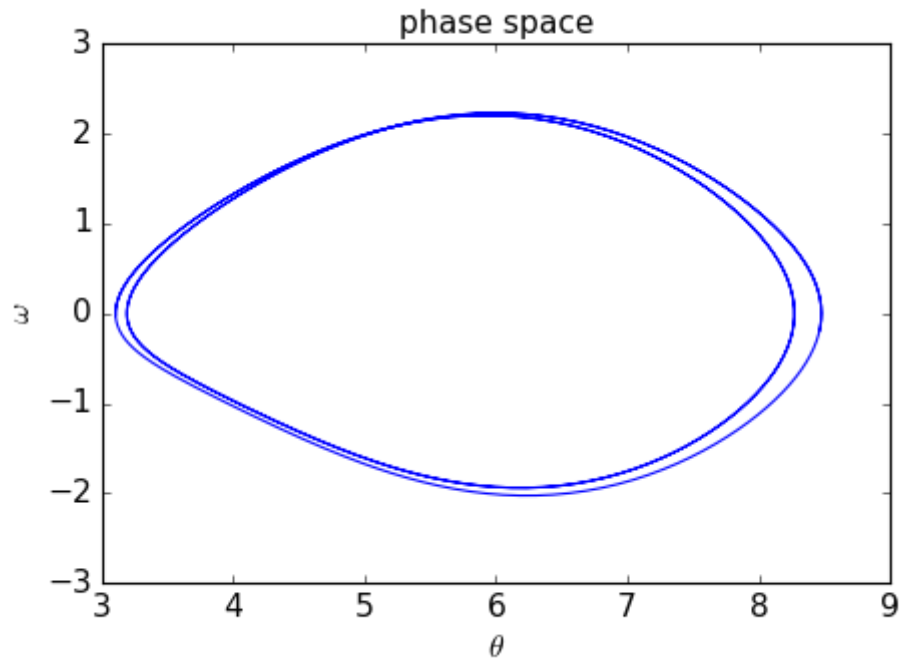
```
In [5]: plt.figure(1)
plt.plot(t,om)
plt.plot(t,th)
plt.axhline(0)
plt.title("Example of period doubling",fontsize=14)
plt.xlabel("$t$")
plt.ylabel("$\theta(t)$ and $\omega(t)$"); # No idea why I need \theta
```

Figure 1



```
In [6]: plt.figure(2)
plt.xlabel("$\\theta$")
plt.ylabel("$\\omega$")
plt.title("phase space", fontsize=14)
tplot = 9000
plt.plot(th[tplot:], om[tplot:]);
```

Figure 2



Example of chaos

Same parameters as above, except for drive amplitude g .

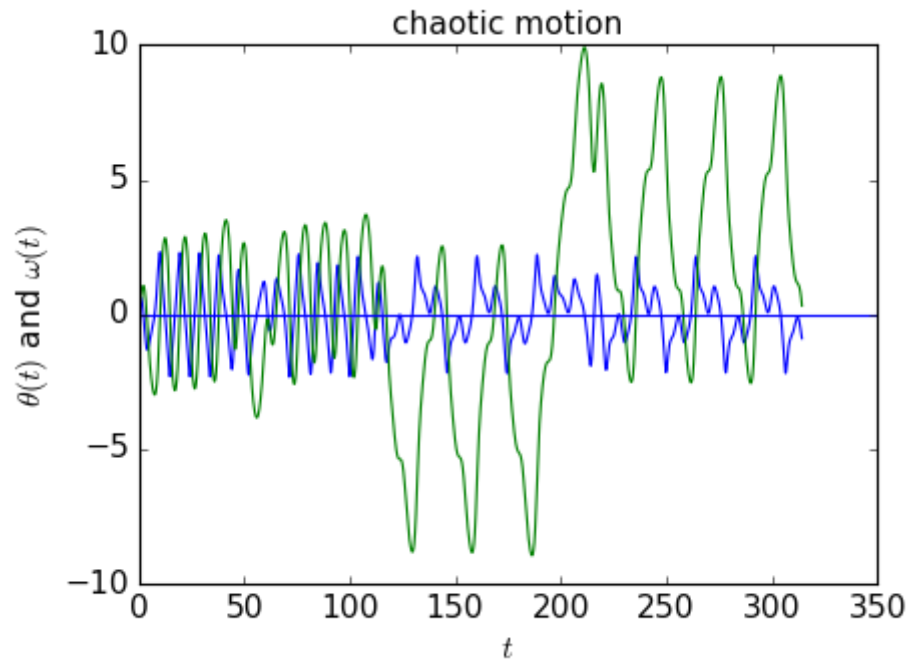
See Golub and Baker Fig. 3.4c

```
In [7]: th0 = 0.1      # Initial theta
om0 = 0              # Initial angular velocity
u0 = sp.array([th0, om0]) # Combine initial conditions in array
b = 0.5              # Damping parameter
g = 1.15             # Driving amplitude
om_d = 2./3          # Driving frequency

t = sp.linspace(0, 50*2.*sp.pi, 50*201) # NOTE: The points selected
                                           # for plotting are not the
                                           # points used for the numerical evaluation
th, om = sp.integrate.odeint(eqs, u0, t, args=(b, g, om_d)).T
```

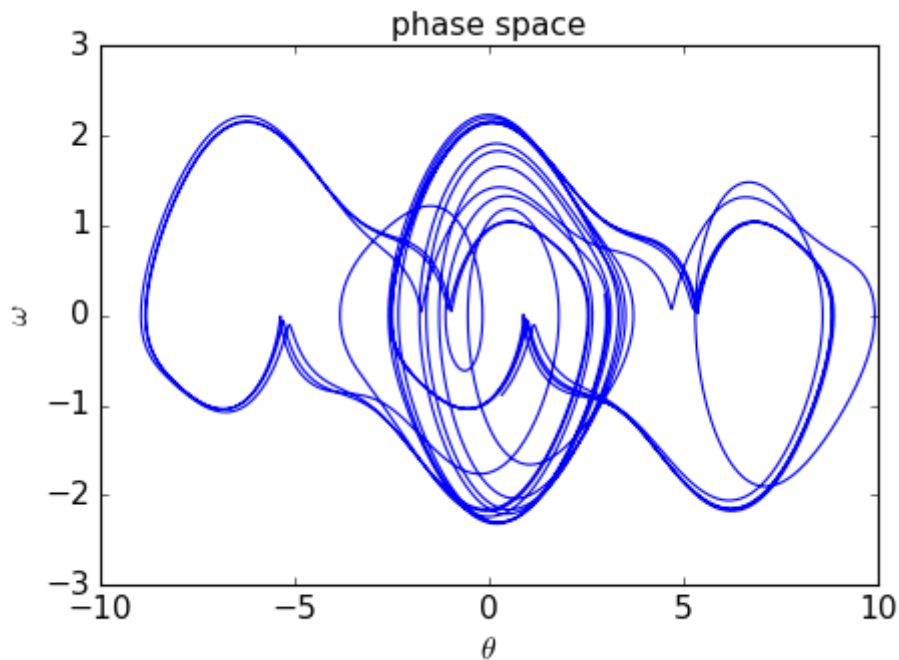
```
In [8]: plt.figure(3)
plt.plot(t,om)
plt.plot(t,th)
plt.axhline(0)
plt.title("chaotic motion",fontsize=14)
plt.xlabel("$t$")
plt.ylabel("$\\theta(t)$ and $\\omega(t)$"); # No idea why I need \\theta
```

Figure 3



```
In [9]: plt.figure(4)
plt.xlabel("$\\theta$")
plt.ylabel("$\\omega$")
plt.title("phase space", fontsize=14)
tplot = 1000
plt.plot(th[tplot:], om[tplot:]);
```

Figure 4


[Back to previous view](#)

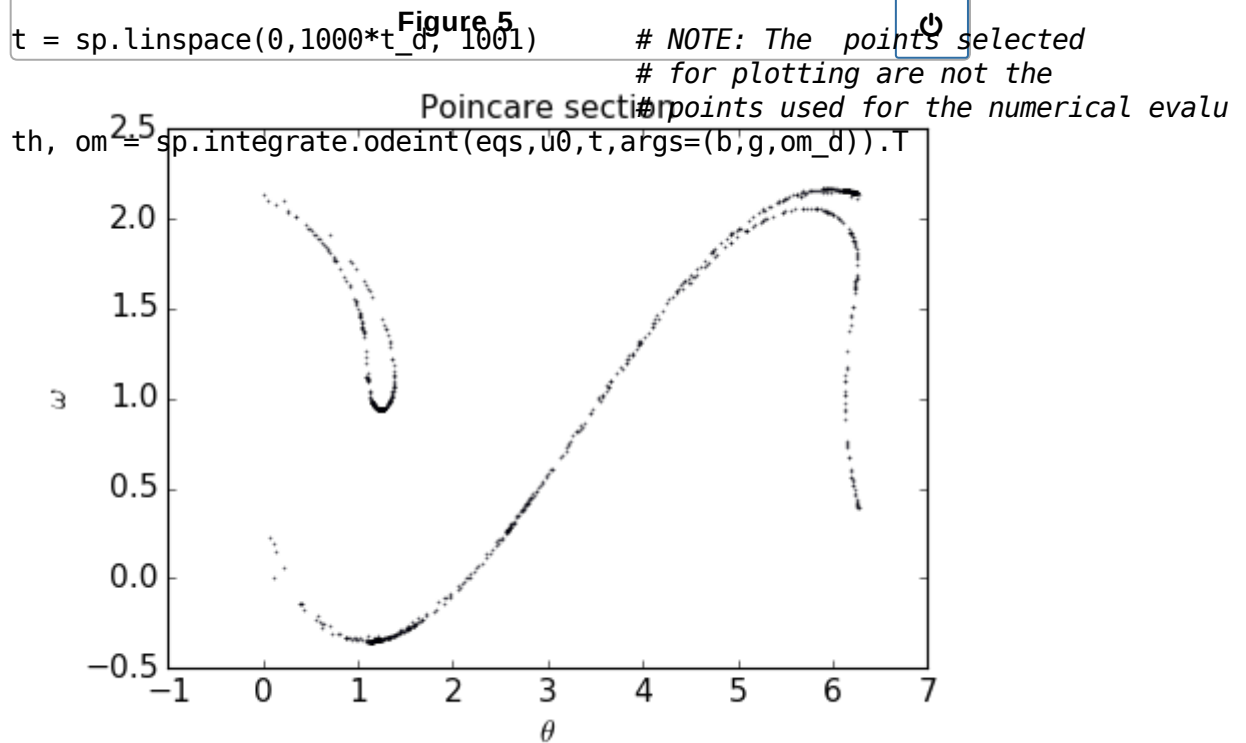
Poincare Section

Simply choose points for output of numerical integration (in the `linspace()` function) that are multiples of the drive period:

$$t_n = n \frac{2\pi}{\omega_d}.$$

In [19]:

```
In [22]: th0 = 0.1 # Initial theta
         om0 = 0 # Initial angular velocity
         u0 = sp.array([th0, om0]) # Combine initial conditions in array
         plt.xlabel("$\\theta$")
         plt.ylabel("$\\omega$")
         # Damping parameter
         plt.title("Poincare section", fontsize=14)
         # Driving amplitude
         plt.scatter(th%(2*sp.pi), om, s=0.1);
         om_d = 2.73 # Driving frequency
         t_d = 2*sp.pi/om_d
```



In []: