Reporte: Actividad 7

Luisa Julieta Casas Cervantes

23 de Marzo del 2018

1 Introducción

Esta es una continuación de la actividad previa (Actividad 6) en la cual trabajamos con sistemas de dos masas acopladas, excepto que en esta ocasión, se añade un factor que cambia el sistema y lo vuelve no lineal.

2 Python y Gráficas resultantes

Definición de variables

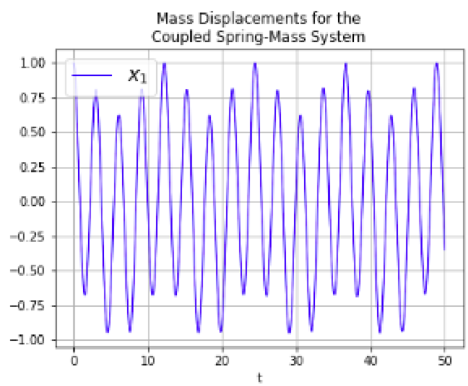
```
In [1]: ####
#y1=dx1,y2=dx2
#dy1=d²x1, dy2= d²x2
##dy1= (-b1*y1-k1*x1+n1*(x1*x1*x1)-k2*(x1-x2)+n2*(x1-x2)*(x1-x2)*(x1-x2)+F1*cos(w1*t))/m1
##dy2= (-b2*y2-k2*(x2-x1)+n2*(x2-x1)*(x2-x1)*(x2-x1)+F2*cos(w2*t))/m2
####
import numpy as np
import matplotlib.pyplot as plt
import math
def vectorfield(w, t, p):
    """
    Defines the differential equations for the coupled spring-mass system.

    Arguments:
        w :  vector of the state variables:
                  w = [x1,y1,x2,y2]
        t :  time
        p :  vector of the parameters:
                  p = [m1,m2,k1,k2,L1,L2,b1,b2]
    """
    x1, y1, x2, y2 = w
    m1, m2, k1, k2, L1, L2, b1, b2, F1, F2, w1, w2, n1, n2, s = p

    f =[y1,
      (-b1*y1-k1*(x1-L1)+n1*(x1*x1*x1)-k2*(x1-x2)+n2*(x1-x2)*(x1-x2)*(x1-x2)+F1*np.cos(w1*s))/m1,
       y2,
       (-b2*y2-k2*(x2-x1-L2)+n2*(x2-x1)*(x2-x1)*(x2-x1)+F2*np.cos(w2*s))/m2]
    return f
```

In [2]:
```python
# Use ODEINT to solve the differential equations defined by the vector field
from scipy.integrate import odeint

# Parameter values
# Masses:
m1 = 1
m2 = 1
# Spring constants
k1 = 0.4
k2 = 1.808
# Natural lengths
L1 = 0
L2 = 0
# Friction coefficients
b1 = 0
b2 = 0
# Nonlinear coefficients
n1 = -1/6
n2 =-1/10
# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = 1
y1 = 0
x2 = -1/2
y2 = 0
#Time
s = 0
#Forces
F1= 0
F2= 0
#Phase
w1= 0
w2= 0
# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 20.0
numpoints = 1000

# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

# Pack up the parameters and initial conditions:
p = [m1, m2, k1, k2, L1, L2, b1, b2, F1, F2, w1, w2, n1, n2, s]
w0 = [x1, y1, x2, y2]

# Call the ODE solver.
wsol = odeint(vectorfield, w0, t, args=(p,),
              atol=abserr, rtol=relerr)

with open('two_springs.dat', 'w') as f:
    # Print & save the solution.
    for t1, w1 in zip(t, wsol):
        print (t1, w1[0], w1[1], w1[2], w1[3], file=f)
```

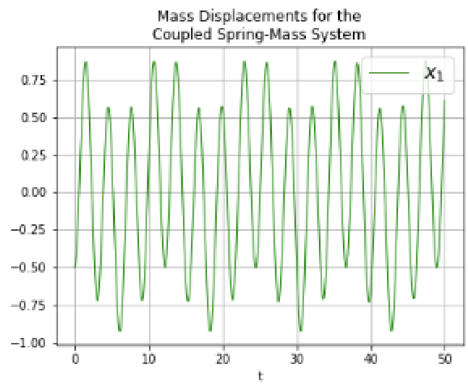
Ejemplo 3.1

In [8]:

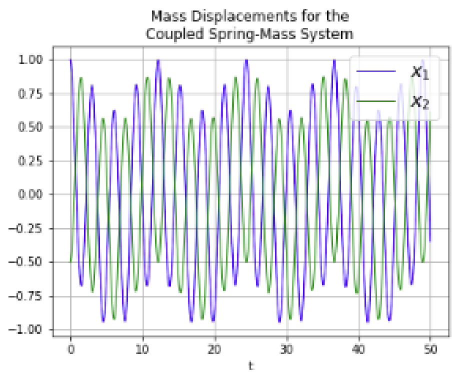
```
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
%matplotlib inline
t, x1, xy, x2, y2 = loadtxt('two_springs3.1.dat', unpack=True)

figure(1, figsize=(6, 4.5))

xlabel('t')
grid(True)
# hold(True)
lw = 1

plot(t, x1, 'b', linewidth=lw)

legend((r'$x_1$', r'$x_2$'), prop=FontProperties(size=16))
title('Mass Displacements for the\nCoupled Spring-Mass System')
savefig('two_springs.png', dpi=100)
```



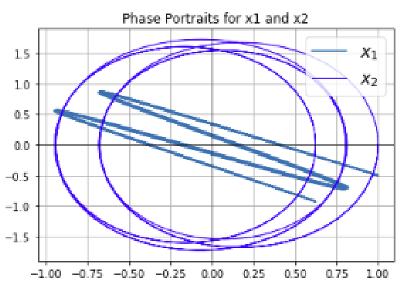
In [9]:
```
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
%matplotlib inline
t, x1, xy, x2, y2 = loadtxt('two_springs3.1.dat', unpack=True)

figure(1, figsize=(6, 4.5))

xlabel('t')
grid(True)
# hold(True)
lw = 1

plot(t, x2, 'g', linewidth=lw)

legend((r'$x_1$', r'$x_2$'), prop=FontProperties(size=16))
title('Mass Displacements for the\nCoupled Spring-Mass System')
savefig('two_springs.png', dpi=100)
```



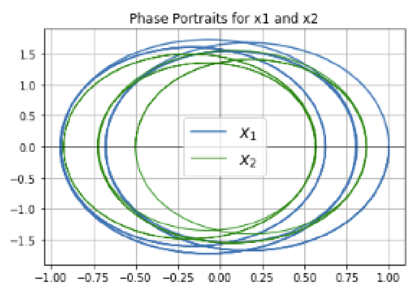
In [11]:
```
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
%matplotlib inline
t, x1, xy, x2, y2 = loadtxt('two_springs3.1.dat', unpack=True)

figure(1, figsize=(6, 4.5))

xlabel('t')
grid(True)
# hold(True)
lw = 1

plot(t, x1, 'b', linewidth=lw)
plot(t, x2, 'g', linewidth=lw)

legend((r'$x_1$', r'$x_2$'), prop=FontProperties(size=16))
title('Mass Displacements for the\nCoupled Spring-Mass System')
savefig('two_springs.png', dpi=100)
```



In [10]:
```
#x1 vs x2
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib
plot(x1,x2)
grid(True)
matplotlib.pyplot.axis("on")
t, x1, xy, x2, y2 = loadtxt('two_springs.dat', unpack=True)

figure(1, figsize=(6, 4.5))

grid(True)

lw = 1

plot(x1, xy, 'b', linewidth=lw)

plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)

legend((r'$x_1$', r'$x_2$'), prop=FontProperties(size=16))
title('Phase Portraits for x1 and x2')
savefig('G2.1b.png', dpi=100)
```



In [7]:
```
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib
plot(x1,xy)
grid(True)
matplotlib.pyplot.axis('on')
t, x1, xy, x2, y2 = loadtxt('two_springs.dat', unpack=True)

figure(1, figsize=(6, 4.5))

grid(True)

lw = 1

plot(x2, y2, 'g', linewidth=lw)

plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)

legend((r'$x_1$', r'$x_2$'), prop=FontProperties(size=16))
title('Phase Portraits for x1 and x2')
savefig('G2.1b.png', dpi=100)
```

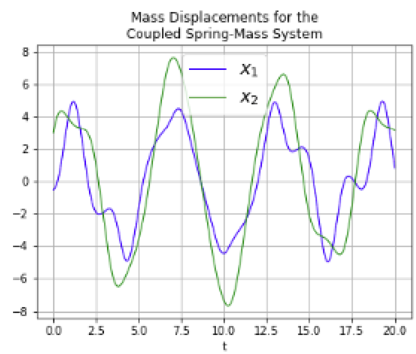


Ejemplo 3.2

In [9]:
```python
from scipy.integrate import odeint

# Parameter values
# Masses:
m1 = 1
m2 = 1
# Spring constants
k1 = 0.4
k2 = 1.808
# Natural lengths
L1 = 0
L2 = 0
# Friction coefficients
b1 = 0
b2 = 0
# Nonlinear coefficients
n1 = -1/6
n2 =-1/10
# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = -0.5
y1 = 0.5
x2 = 3.001
y2 = 5.9
#Time
s = 0
#Forces
F1= 0
F2= 0
#Phase
w1= 0
w2= 0
# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 20.0
numpoints = 1000

# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

# Pack up the parameters and initial conditions:
p = [m1, m2, k1, k2, L1, L2, b1, b2, F1, F2, w1, w2, n1, n2, s]
w0 = [x1, y1, x2, y2]

# Call the ODE solver.
wsol = odeint(vectorfield, w0, t, args=(p,),
              atol=abserr, rtol=relerr)

with open('two_springs3.2.dat', 'w') as f:
    # Print & save the solution.
    for t1, w1 in zip(t, wsol):
        print (t1, w1[0], w1[1], w1[2], w1[3], file=f)
```

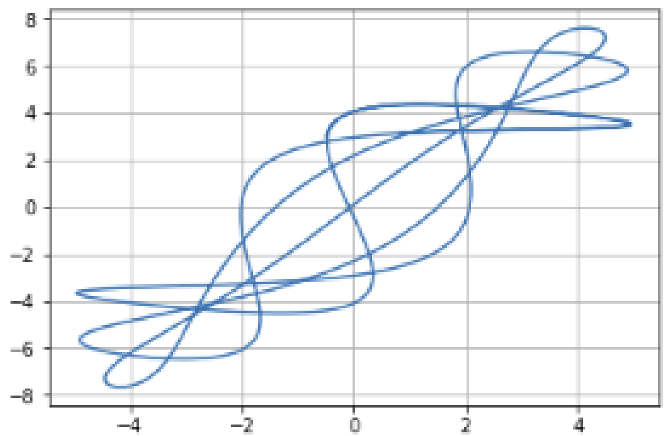
In [11]:

```
# Plot the solution that was generated
#Ejemplo 3.2 del archivo
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
%matplotlib inline
t, x1, xy, x2, y2 = loadtxt('two_springs3.2.dat', unpack=True)

figure(1, figsize=(6, 4.5))

xlabel('t')
grid(True)
# hold(True)
lw = 1

plot(t, x1, 'b', linewidth=lw)
plot(t, x2, 'g', linewidth=lw)

legend((r'$x_1$', r'$x_2$'), prop=FontProperties(size=16))
title('Mass Displacements for the\nCoupled Spring-Mass System')
savefig('two_springs3.2.png', dpi=100)
```



In [12]:
```python
import matplotlib
plot(x1,x2)
grid(True)
matplotlib.pyplot.axis('on')
```

Out[12]:
```
(-5.4558851475110002,
 5.4481485733110002,
 -8.4288548270979997,
 8.3874542207180003)
```

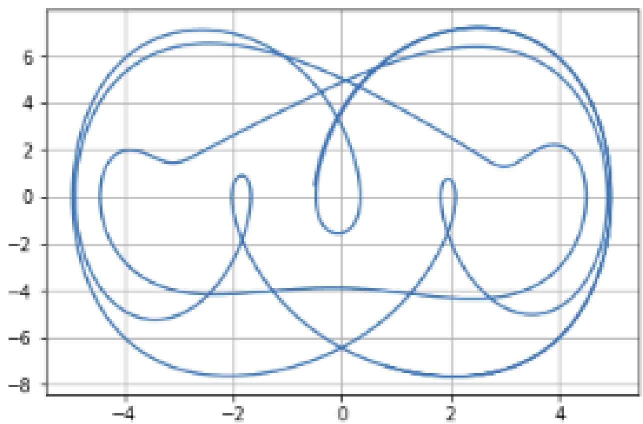


In [13]:

```python
import matplotlib
# y1 = xy
plot(x1,xy)
grid(True)
matplotlib.pyplot.axis('on')
```

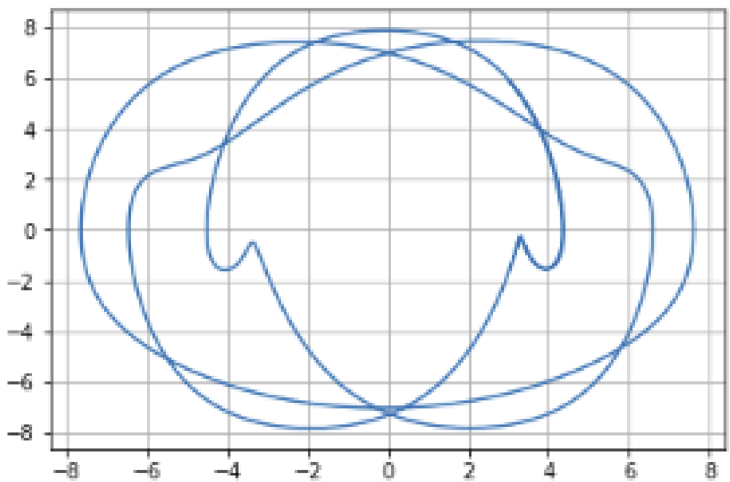
Out[13]:

```
(-5.4558851475110002,
 5.4481485733110002,
 -8.4093577131364992,
 7.9476264041065008)
```



In [14]:
```
import matplotlib
plot(x2,y2)
grid(True)
matplotlib.pyplot.axis('on')
```

Out[14]: (-8.4288540270979997, 8.3874542207180003, -8.6334360774615, 8.685820

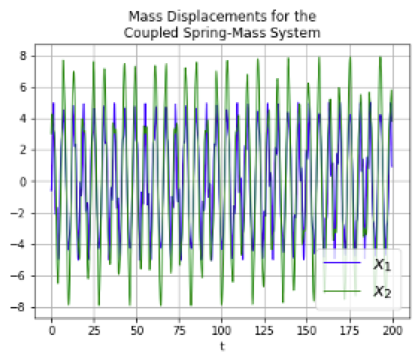


Ejemplo 3.3

In [7]:

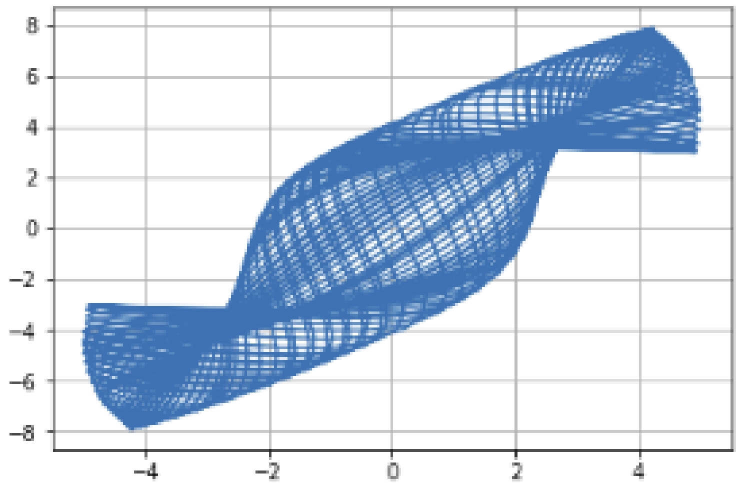
```python
from scipy.integrate import odeint

# Parameter values
# Masses:
m1 = 1
m2 = 1
# Spring constants
k1 = 0.4
k2 = 1.808
# Natural lengths
L1 = 0
L2 = 0
# Friction coefficients
b1 = 0
b2 = 0
# Nonlinear coefficients
n1 = -1/6
n2 =-1/10
# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = -0.6
y1 = 0.5
x2 = 3.001
y2 = 5.9
#Time
s = 0
#Forces
F1= 0
F2= 0
#Phase
w1= 0
w2= 0
# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 200.0
numpoints = 1500

# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

# Pack up the parameters and initial conditions:
p = [m1, m2, k1, k2, L1, L2, b1, b2, F1, F2, w1, w2, n1, n2, s]
w0 = [x1, y1, x2, y2]

# Call the ODE solver.
wsol = odeint(vectorfield, w0, t, args=(p,),
              atol=abserr, rtol=relerr)

with open('two_springs3.3.dat', 'w') as f:
    # Print & save the solution.
    for t1, w1 in zip(t, wsol):
        print (t1, w1[0], w1[1], w1[2], w1[3], file=f)
```

In [8]:
```
# Plot the solution that was generated
#Ejemplo 3.2 del archivo
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
%matplotlib inline
t, x1, xy, x2, y2 = loadtxt('two_springs3.3.dat', unpack=True)

figure(1, figsize=(6, 4.5))

xlabel('t')
grid(True)
# hold(True)
lw = 1

plot(t, x1, 'b', linewidth=lw)
plot(t, x2, 'g', linewidth=lw)

legend((r'$x_1$', r'$x_2$'), prop=FontProperties(size=16))
title('Mass Displacements for the\nCoupled Spring-Mass System')
savefig('two_springs3.3.png', dpi=100)
```



In [9]:
```
import matplotlib
plot(x1,x2)
grid(True)
matplotlib.pyplot.axis('on')
```

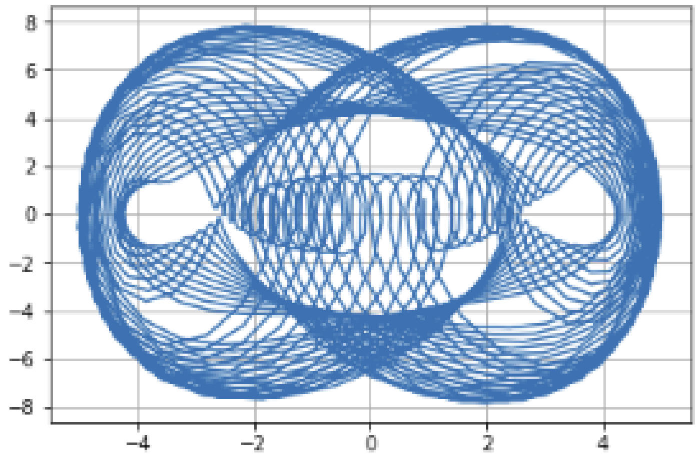
Out[9]: (-5.5166427039629999, 5.520455961443, -8.6737054749630005, 8.6728357



In [10]:

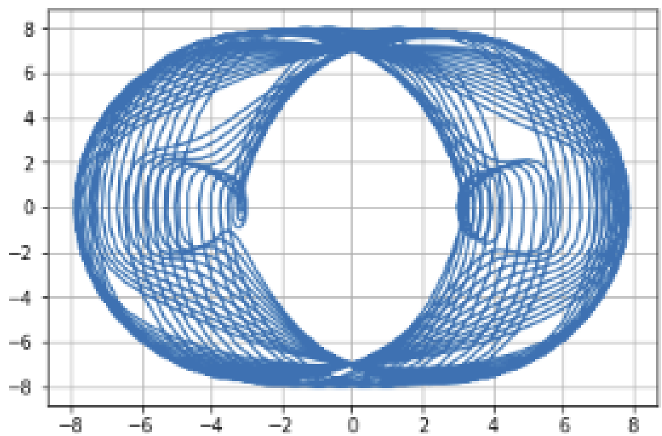
```
import matplotlib
# y1 = xy
plot(x1,xy)
grid(True)
matplotlib.pyplot.axis('on')
```

Out[10]: (-5.5166427039629999, 5.520455961443, -8.6240619587659992, 8.624787711726



```
In [11]: import matplotlib
         plot(x2,y2)
         grid(True)
         matplotlib.pyplot.axis('on')
```

```
Out[11]: (-8.6737054749630005,
          8.6728357936429994,
          -8.8095977776550001,
          8.8019072051750005)
```



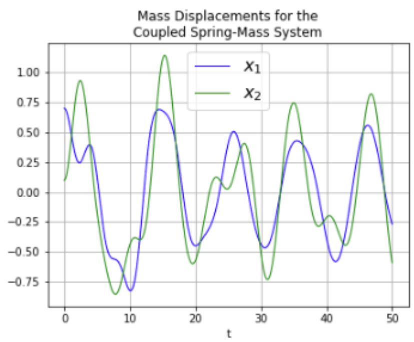
Ejemplo 4.1

In [5]:

```python
# Use ODEINT to solve the differential equations defined by the vector field
from scipy.integrate import odeint

#Parametros para el ejemplo 4.1
# Parameter values
# Masses:
m1 = 1.0
m2 = 1.0
# Spring constants
k1 = (2/5)
k2 = 1.0
# Natural lengths
L1 = 0.0
L2 = 0.0
# Friction coefficients
b1 = (1/10)
b2 = (1/5)
#Fuerzas aplicadasw
F1 = (1/3)
F2 = (1/5)
#Valores no lineales
n1 = (1/6)
n2 = (1/10)
#frecuencias w
w1 = 1.0
w2 = (3/5)

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = 0.7
y1 = 0.0
x2 = 0.1
y2 = 0.0


# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 50.0
numpoints = 1500

# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

# Pack up the parameters and initial conditions:
p = [m1, m2, k1, k2, L1, L2, b1, b2, F1, F2, n1, n2, w1, w2]
w0 = [x1, y1, x2, y2]

# Call the ODE solver.
wsol = odeint(vectorfield, w0, t, args=(p,),
              atol=abserr, rtol=relerr)

with open('two_springs41.dat', 'w') as f:
    # Print & save the solution.
    for t1, w1 in zip(t, wsol):
        print  ( t1, w1[0], w1[1], w1[2], w1[3], file=f )
```

In [6]:
```python
# Plot the solution that was generated
#Ejemplo 4.1 del archivo
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties
%matplotlib inline
t, x1, xy, x2, y2 = loadtxt('two_springs41.dat', unpack=True)

figure(1, figsize=(6, 4.5))

xlabel('t')
grid(True)
# hold(True)
lw = 1

plot(t, x1, 'b', linewidth=lw)
plot(t, x2, 'g', linewidth=lw)

legend((r'$x_1$', r'$x_2$'), prop=FontProperties(size=16))
title('Mass Displacements for the\nCoupled Spring-Mass System')
```

Out[6]:
```
Text(0.5,1,'Mass Displacements for the\nCoupled Spring-Mass System')
```

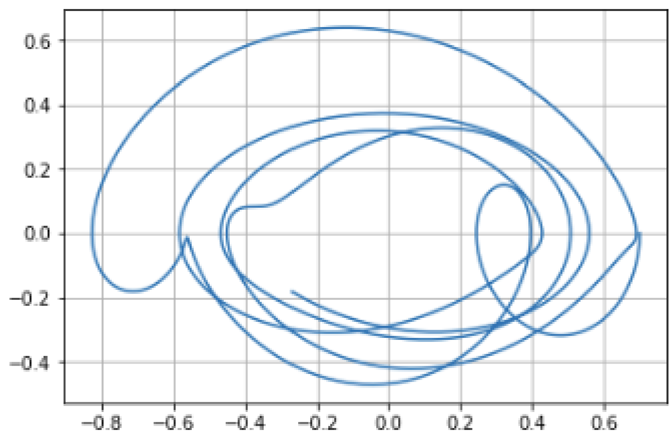


In [7]:

```
import matplotlib
plot(x1,xy)
grid(True)
matplotlib.pyplot.axis('on')
```

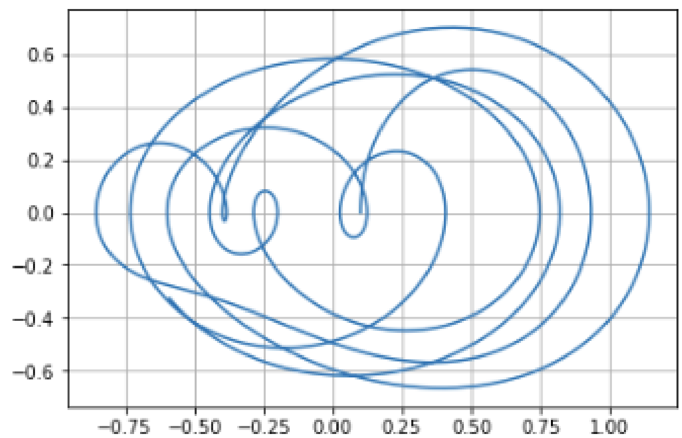
Out[7]:

```
(-0.90225599797864997,
 0.77629790466564996,
 -0.52707077144134995,
 0.69455044091634999)
```



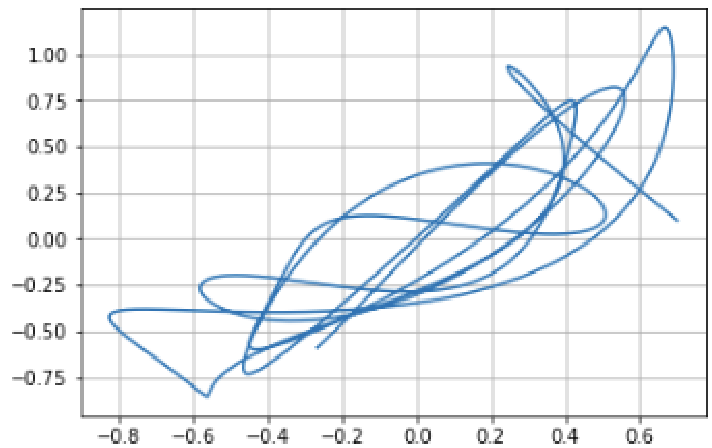
In [8]:
```
import matplotlib
plot(x2,y2)
grid(True)
matplotlib.pyplot.axis('on')
```

Out[8]:
```
(-0.95524030764955004,
 1.2442741760785501,
 -0.73512919680184996,
 0.77001096275085001)
```



In [9]:
```
import matplotlib
plot(x1,x2)
grid(True)
matplotlib.pyplot.axis('on')
```

Out[9]:
```
(-0.90225599797864997,
 0.77629790466564996,
 -0.95524030764955004,
 1.2442741760785501)
```



3 Apéndice

1. ¿Qué más te llama la atención de la actividad completa? ¿Que se te hizo menos interesante?
   Las figuras que salen de movimientos oscilatorios. Lo que no me gustó es que no cambian mucho con diferentes datos.

2. ¿De un sistema de masas acopladas como se trabaja en esta actividad, hubieras pensado que abre toda una nueva área de fenómenos no lineales?
   Me lo sospechaba.

3. ¿Qué propondrías para mejorar esta actividad? ¿Te ha parecido interesante este reto?
   Datos más variados para ver las figuras.

4. ¿Quisieras estudiar mas este tipo de fenómenos no lineales?
   Teóricamente sí.