Sistemas de masas con oscilaciones no lineales y forzadas

Ramses Pacheco Ortiz

14 De abril Del 2018

1 Introducción

Esta practica es continuacion de la actividad 6 que incluyen las seciones 1 y 2 del articulo llamado "coupled springs equations" de los autores Fay y Graham,en donde se vio un sistema de resortes acoplado.

En esta actividad abordaremos el resto de las secciones de este articulo, especificamente la seccion 3 y 4, en donde cada seccion tiene su nivel de complejidad por ejemplo,en la seccion 3 toman consideraciones y fuerzas no lineales, esto quiere decir que no es proporcional al desplazamiento y por lo tanto trabajaremos ecuaciones no linenales.

Por otro lado en la seccion 4 es similar a la seccion 3 pero tomando otras consideraciones como un forzamiento externos dando soluciones de $coswt$ llamdas soluciones harmonicas o en tal caso subharmonicas.

2 añadiendo no-linealidad

En esta parte del articulo realizamos varia suposiciones,por ejemplño, Si suponemos que las fuerzas de restauración no son lineales, lo que sin duda es cierto para grandes vibraciones,donde la fuerza de restauración es de la forma $F = -kx$ (ley de Hooke), la cual no describe correctamente el sistema acopaldo,por que lo podemos considerar que la fuerza de restauración tiene la forma $f = kx+ux3$. Entonces de nuestro modelo se deducen las siguientes ecuaciones:

$$m_1\ddot{x}_1 = -\delta_1\dot{x}_1 - k_1x_1 + \mu_1x_1^3 - k_2(x_1 - x_2) + \mu_2(x_1 -$$

$$m_2\ddot{x}_2 = -\delta_2\dot{x}_2 - k_2(x_2 - x_1) + \mu_2(x_2 - x_1)^3$$

2.1 Ejemplo 3.1

Suponga que $m1 = m2 = 1$. Describa el movimiento de las constantes de resorte $k1 = 0.4$ y $k2 = 1.808$, coeficientes de amortiguación b1=0 y b2=0, coeficientes no lineales $u1 = -1/6$ y $u2 = -1/10$, con condiciones iniciales:

$$(x_1(0)), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (1, 0, -1/2, 0).$$

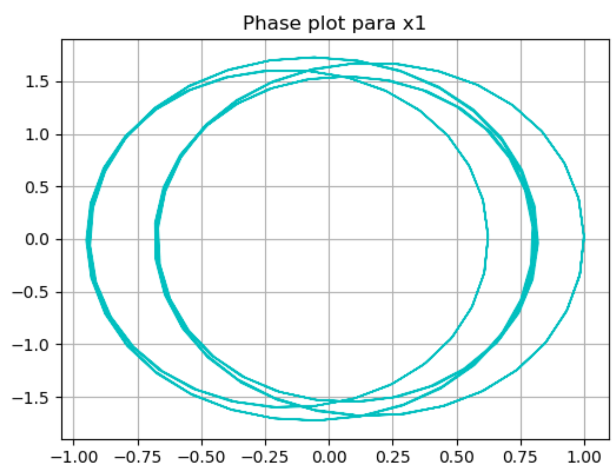
El codigo sigueinte es muy similar al utilizado en la actividad 6 con ecepcion que se agregaron los parametros $u1$ y $u2$

```
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, u1, u2 = p

    # Create f = (x1',y1',x2',y2'):
    f = [y1,
         (-b1 * y1 - k1 * (x1) + k2 * (x2 - x1)+ u1*((x1)**3)+ u2*((x1-x2)**3)) / m1,
         y2,
         (-b2 * y2 - k2 * (x2 - x1)+ u2*((x2-x1)**3)) / m2]
    return f
```

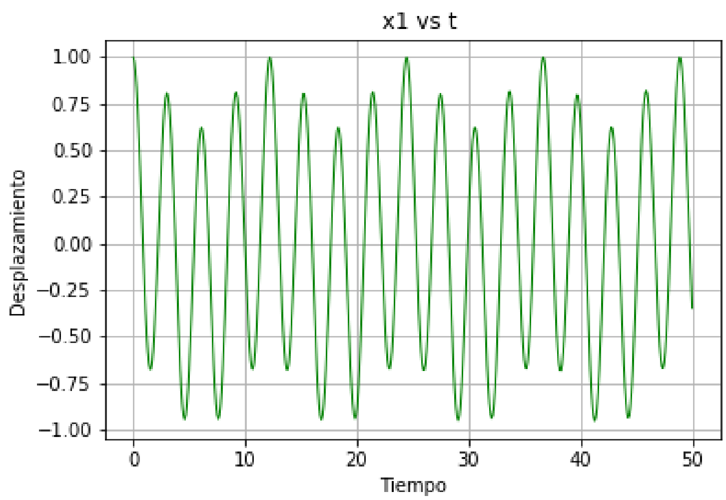
```python
# Use ODEINT to solve the differential equations defined by the vector field
from scipy.integrate import odeint
import numpy as np

# Parameter values
# Masses:
m1 = 1.0
m2 = 1.0
# Spring constants
k1 = 0.4
k2 = 1.808
# Natural lengths
L1 = 0.0
L2 = 0.0
# Friction coefficients
b1 = 0.0
b2 = 0.0

#Non-linear coefficients
u1=-1.0/6.0
u2=-1.0/10.0

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = 1.0
y1 = 0.0
x2 = -1.0/2.0
y2 = 0.0

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 50
numpoints = 500
```

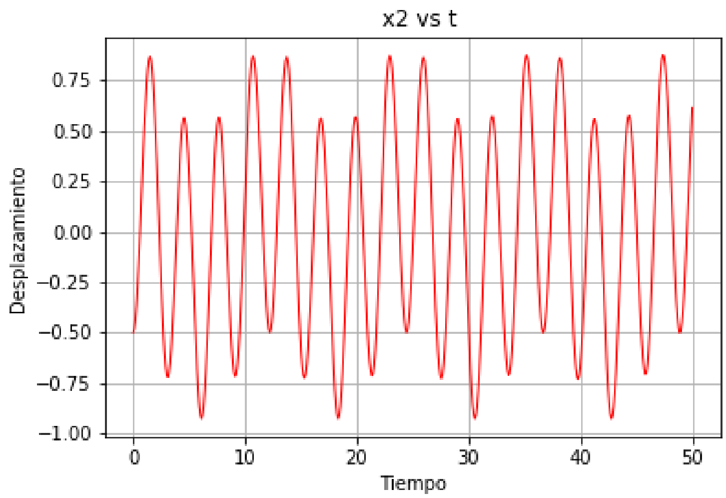
```python
# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

# Pack up the parameters and initial conditions:
p = [m1, m2, k1, k2, L1, L2, b1, b2,u1, u2]
w0 = [x1, y1, x2, y2]

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

with open('nolineal3.1.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)
```

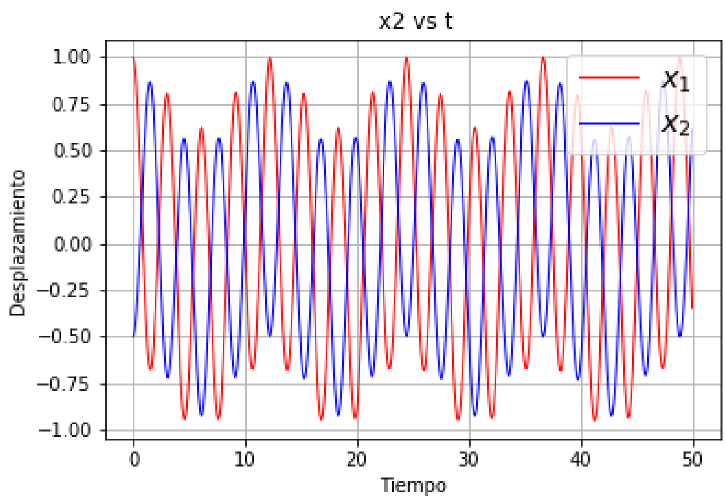
Las graficas que se llevaron a cabo en este problema son las siguientes:

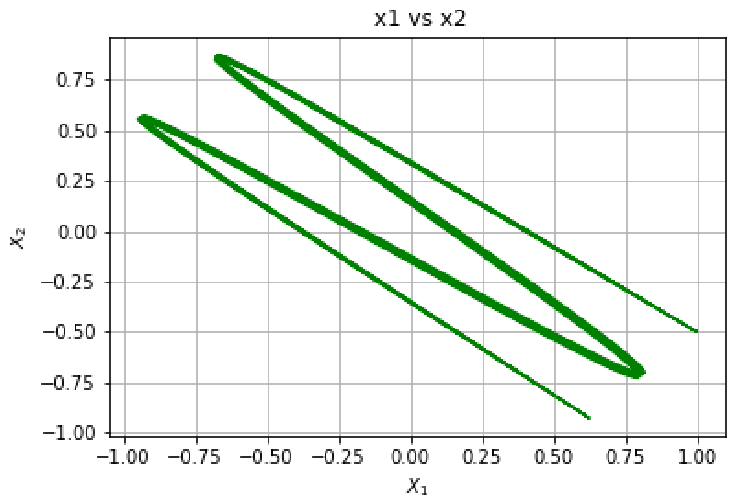



Phase plot para x2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75


x1 vs t
Desplazamiento
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50
Tiempo


x2 vs t
Desplazamiento
Tiempo
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50


x2 vs t
Desplazamiento
Tiempo
x_1
x_2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50



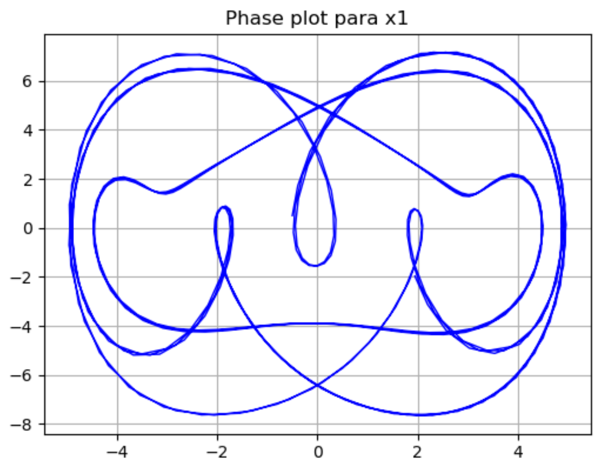
2.2 Ejemplo 3.2

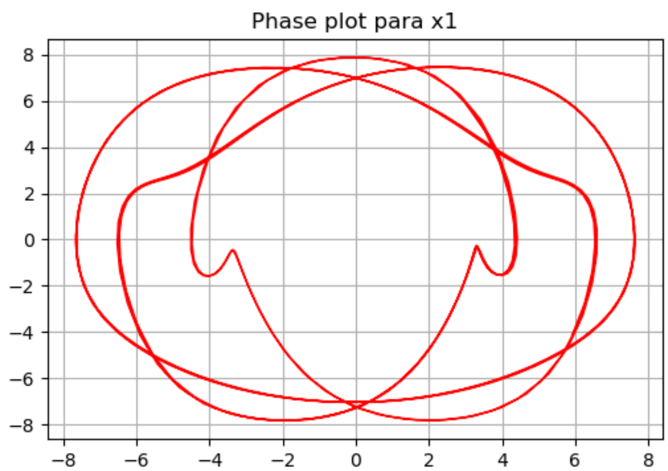
Suponga que $m1 = m2 = 1$. Describa el movimiento de las constantes de resorte $k1 = 0.4$ y $k2 = 1.808$, coeficientes de amortiguación b1=0 y b2=0, coefecientes no lineales $u1 = -1/6$ y $u2 = -1/10$, con condiciones iniciales:

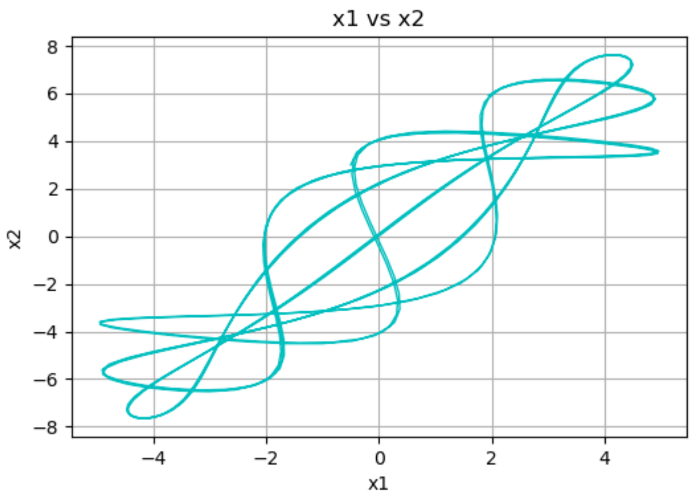
$$(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (-0.6, 1/2, 3.001, 5.9)$$

En este ejemplo el segmento del codigo es el mismo solamente cambiando las condiciones iniciales que ofrece el problema

Las graficas que se llevaron a cabo en este problema son las siguientes:


Phase plot para x1

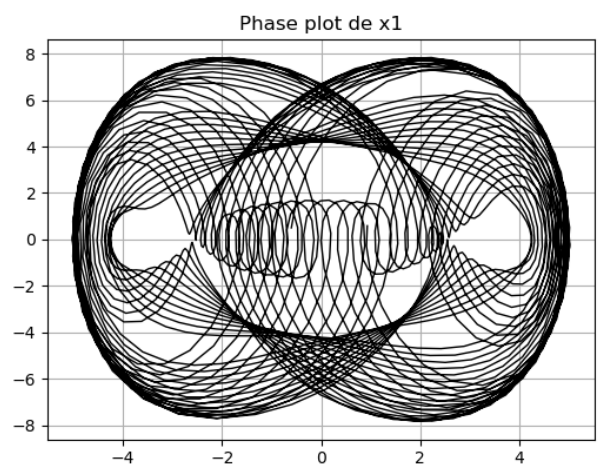

Phase plot para x1

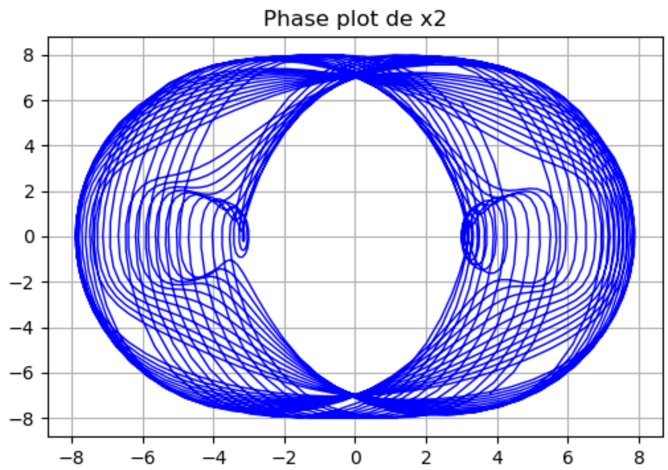


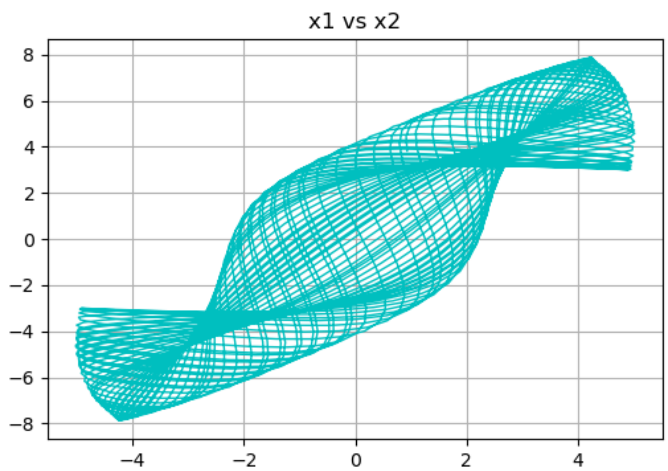
2.3 Ejemplo 3.3

Suponga que $m1 = m2 = 1$. Describa el movimiento de las constantes de resorte $k1 = 0.4$ y $k2 = 1.808$, coeficientes de amortiguación b1=0 y b2=0, coefecientes no lineales $u1 = -1/6$ y $u2 = -1/10$, con condiciones iniciales, para este ejemplo de igual manera no se presenta el segmento del codigo debido aque es el mismo simplemente cambiando las condiciones iniciales.

Las graficas que se realizaron en este ejemplo son las siguientes:







3 Añadiendo forzamiento

En esta parte del texto habla de un sistema o modelo de forzamiento externo y da un ejemplo como el sinusoidal que tiene la forma $Fcoswt$ y se obtendra las siguintes ecuaciones diferenciales:

$$m_1\ddot{x}_1 = -\delta_1\dot{x}_1 - k_1x_1 + \mu_1x_1^3 - k_2(x_1 - x_2) + \mu_2(x_1 - x_2)^3 + F_1\cos\omega_1 t$$

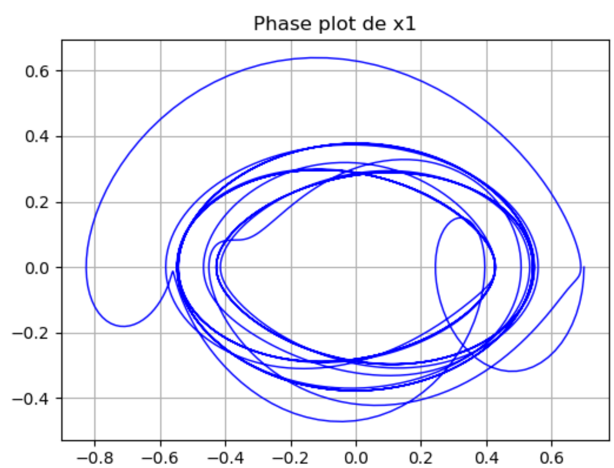
$$m_2\ddot{x}_2 = -\delta_2\dot{x}_2 - k_2(x_2 - x_1) + \mu_2(x_2 - x_1)^3 + F_2\cos\omega_2 t$$

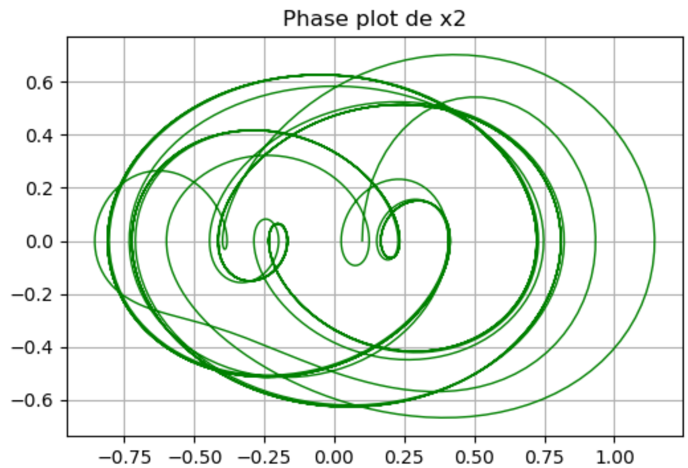
En el codigo se agregaran los parametros f1,f2 y w1,w2 como a continuacion se muestra en la imagen:

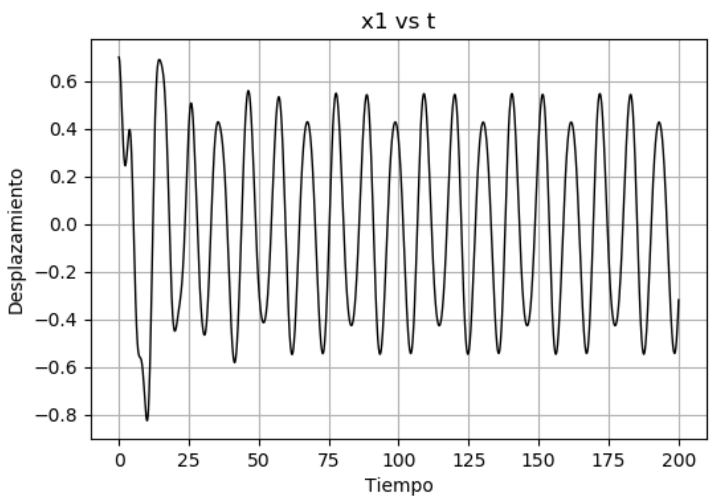
```
import numpy as np
def vectorfield(w, t, p):
    """
    Defines the differential equations for the coupled spring-mass system.

    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, u1, u2, f1, f2, w1, w2 = p

    # Create f = (x1',y1',x2',y2'):
    f = [y1,
         (-b1 * y1 - k1 * (x1) + k2 * (x2 - x1)+ u1*((x1)**3)+ u2*((x1-x2)**3)+ f1*np.cos(w1*t)) /
         y2,
         (-b2 * y2 - k2 * (x2 - x1)+ u2*((x2-x1)**3)+f2*np.cos(w2*t)) / m2]
    return f
```

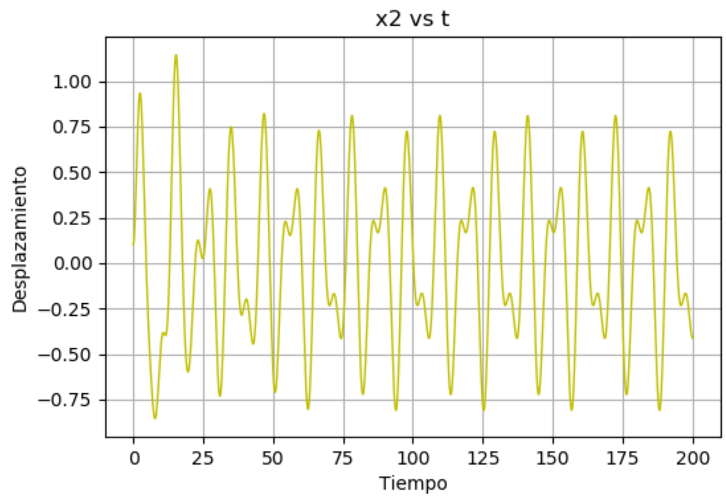
```python
# Use ODEINT to solve the differential equations defined by the vector field
from scipy.integrate import odeint
import numpy as np

# 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 = 1.0/10.0
b2 = 1.0/5.0

#Non-linear coefficients
u1=1.0/6.0
u2=1.0/10.0

#Forcing amplitudes
f1=1.0/3.0
f2=1.0/5.0

#Forcing frecuencies
w1=1.0
w2=3.0/5.0
```

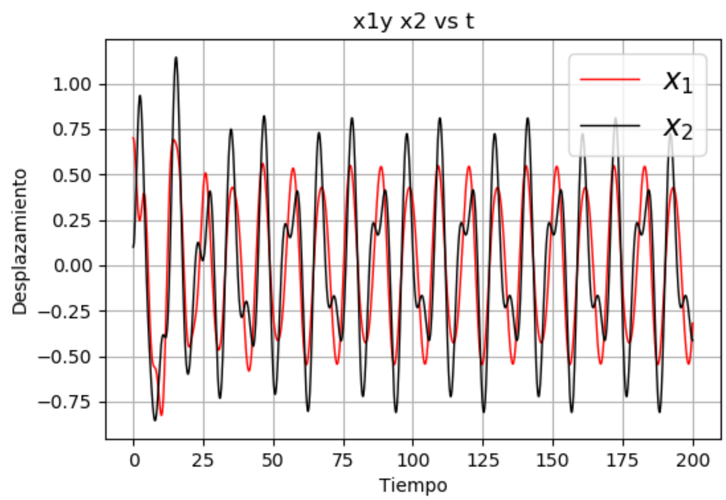
Acontinuacion se mostraran las graficas que se realizaron en este ejemplo:

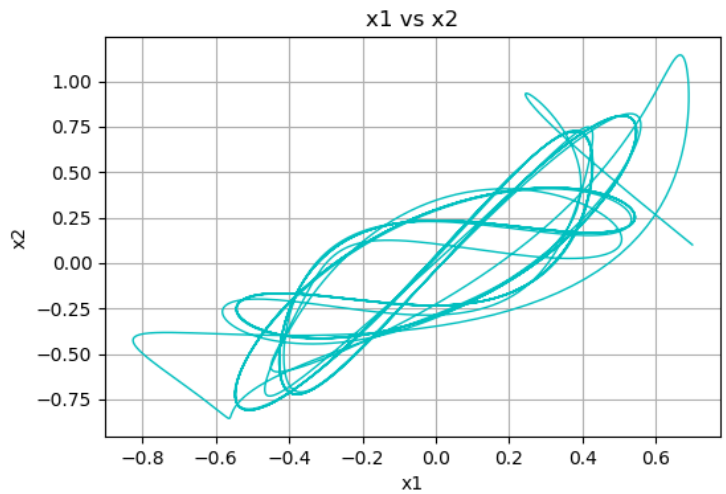



Phase plot de x2
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


x1 vs t
Desplazamiento
Tiempo
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
25
50
75
100
125
150
175
200


x2 vs t
Desplazamiento
Tiempo


x1y x2 vs t
x_1
x_2
Desplazamiento
Tiempo



4 Conclusión

Me paracio muy interesante esta actividad ya que fue un poco diferente a lo de la actividad 6, un poco mas complejo, y la graficas mas "feas"

5 Bibliografia

couple spring equations,TEMPLE H. FAY and SARAH DUNCAN GRAHAM,publicado 12 de septiembre del 2012 $http : //math.oregonstate.edu/ gibsonn/Teaching/MTH323-010S15/Supplements/coupled_spring.pdf$