Sistemas de resortes acoplados

Ramses Pacheco Ortiz

De Febrero Del 2018

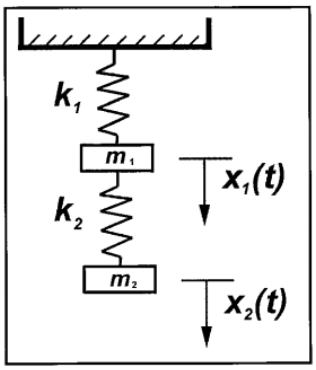
1 Introducción

En esta actividad,tiene como objetivo la modelación de fenómenos físicos, apoyados con Python(jupyter lab).Iniciaremos estudiando la ley de hooke u oscilaciones,un tema antes visto en la materia de mecanica 1 y 2 ,en la cual aplicaremos nuestro conocimiento,para esto nos basaremos en un archivo llamado "coupled springs equations" de los autores Fay y Graham. analizaremos cuatro casos en donde cada uno de ellos cuenta con ciertos parametros conocidos como las constantes k de resortes, masas,desplazameintos,longitud natural,etc.

presentaremos el codigo que se utilizo para resolver dicho problema mostrando su error relativo en lo calculado utilizando la biblioteca scipy.integrate.odeint para despues compararlo con la solucion que nos presenta dicho articulo,incluiremos una serie de graficas generadas por los datos, y para finalizar mostreremos un conclusion general,basados en unas preguntas reflexivas sobre lo aprendido.

2 Modelo de resortes acoplados

la sigueinte imagen muestras el sistema de resortes acoplados:



Como se puede ver en la imagen el modelo consisite en dos resortes y dos masas,un resorte con una constante$k1$ esta sujetado del techo colgando un masa(

$m1$), y en este mismo resorte esta sujetado el segundo resorte con una constante $k2$ tambien colgando un masa($m2$). el sistema permite que descanse en equilibrio, medimos el desplazamiento del centro de masa de cada peso del equilibrio, como una función del tiempo medidas por $x1(t)$ y $x2(t)$, respectivamente.

Asumiendo que el sistema se mueve en oscilaciones pequeñas, y debido a la ley de hooke, la fuerza restauradora que presentaron los resortes seran de la forma $k1l1$ y $k2l2$ respectivamente donde $l1$ y $l2$ son las elongaciones o compresiones de los resortes.

realizando un diagrama de fuerzas en el sistema y aplicando la ley de newton podemos destacar estas 2 ecuaciuon siguientes al momento de hacer la sumatoria de fuerzas:

$$m_1\ddot{x}_1 = -k_1x_1 - k_2(x_1 - x_2)$$

$$m_2\ddot{x}_2 = -k_2(x_2 - x_1)$$

De esta manera obtendremos un par ecuaciones diferenciales de segundo orden, encontrando una ecuación para $x1$ pero que no involucre la variable $x2$ despejando a $x2$ nos queda:

$$x_2 = \frac{m_1\ddot{x}_1}{k_2} + \frac{k_1 + k_2}{k_2}x_1$$

sustituyendo $x2$ en la segunda ecuacion diferencial y simplificando obtenemos:

$$m_1m_2x_1^{(4)} + (m_2k_1 + k_2(m_1 + m_2))\ddot{x}_1 + k_1k_2x_1 = 0$$

podemos ver que el movimiento de la primera masa ya esta determinada por la ecuacion lineal de cuarto orden, ahora para botener un cuacion que solo involucre la variable $x2$ despejaremos la $x1$ de la segunda ecuacion diferencial:

$$x_1 = \frac{m_2}{k_2}\ddot{x}_2 + x_2$$

sustituyendo la ecuacion anterior en la primera ecuacion y simplificando nos que da:

$$m_1m_2x_2^{(4)} + (m_2k_1 + k_2(m_1 + m_2))\ddot{x}_2 + k_1k_2x_2 = 0$$

podemos observar que esta ultima ecuacion tiene la misma estructura que la ecuacion para el primer resorte y si consideramos que las masas de los dos resortes vale $1kg$ podemos simplificarla dando nos la siguiente ecuacion:

$$m^4 + (k_1 + 2k_2)m^2 + k_1k_2 = 0$$

la cual sus racies son:

$$\pm\sqrt{-\frac{1}{2}k_1 - k_2 \pm \frac{1}{2}\sqrt{k_1^2 + 4k_2^2}}$$

3 Problemas

3.1 Ejemplo 2.1

Describir el movimiento para las constantes del resorte k1=6 y k2=4 con las condiciones iniciales :

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

para comenzar primeramente tenemos que establecer el vectorfield que define las ecuaciones del sistema de resorte acoplado, a continuacion se mostrara como se definio el vertorfield:

```
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 = p

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

las variables de entrada son w,t,p:

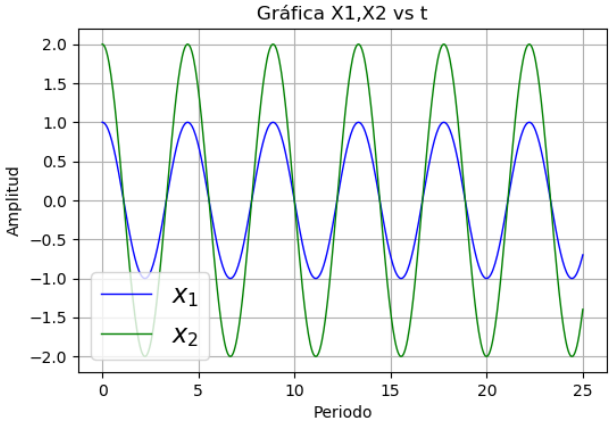
- w es el vector que contienen los parametros como la posicion(xi) y la velocidad(wi)
- t es el tiempo
- p es el vector que contiene varias constantes como la k,logitud(l),masas(m),etc.

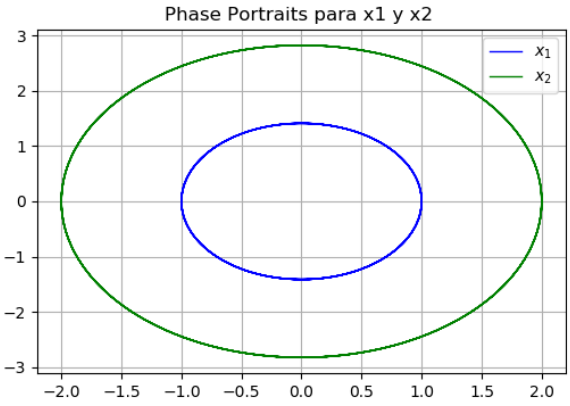
para proseguir se importan las bibliotecas numpy y spicy con la funcion odeint para poder resolver las ecuaciones diferenciales:

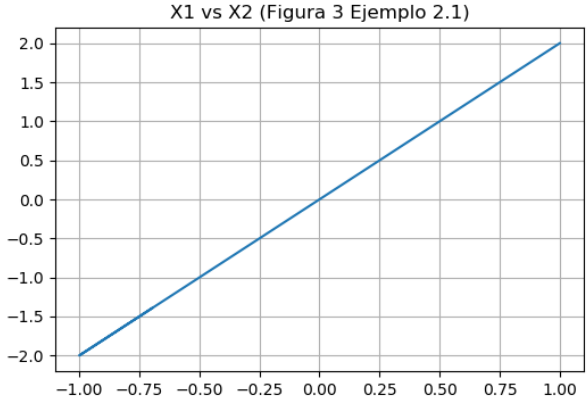
```
# Use ODEINT to solve the differential equations defined by the vector field
from scipy.integrate import odeint
import numpy as np

# Parameter values
# Masses:
m1 = 1.0
m2 = 1.0
# Spring constants
k1 = 6.0
k2 = 4.0
# Natural lengths
L1 = 0.0
L2 = 0.0
# Friction coefficients
b1 = 0.0
b2 = 0.0

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

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 25
numpoints = 500

# 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]
w0 = [x1, y1, x2, y2]

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

#La solución analítica es:
#X1(t)=cos(sqrt(2)*t)
#X2(t)=2*cos(sqrt(2)*t)
#El arreglo W1 tiene como columna 0 a x1 y como columna 2 a x2

with open('two_springs1.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],np.abs((w1[0]-(np.cos(np.sqrt(2)*t1)))/
(np.cos(np.sqrt(2)*t1))), np.abs((w1[2]-(2*np.cos(np.sqrt(2)*t1)))/(2*np.cos(np.sqrt(2)*t1))), file=f)
```

Ya que resolvimos la ecuacion se procedio a graficar los desplazameintos x1 y x2 contra el tiempo, la phase portraits y x1 vs x2:



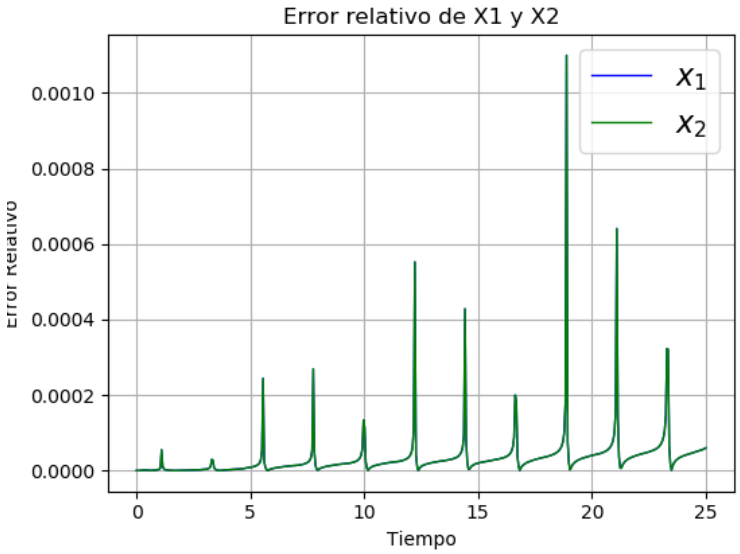




la solucion analitica que se obtubo en el articulo esta dada por:

$$x_1(t) = \cos\sqrt{2}t$$
$$x_2(t) = 2\cos\sqrt{2}t$$

La grafica del error relativo entre la solucion calculada y la que ofrece el articulo es la siguiente:



3.2 Ejemplo 2.2

Describir el movimiento para las constantes del resorte k1=6 y k2=4 con las condiciones iniciales :

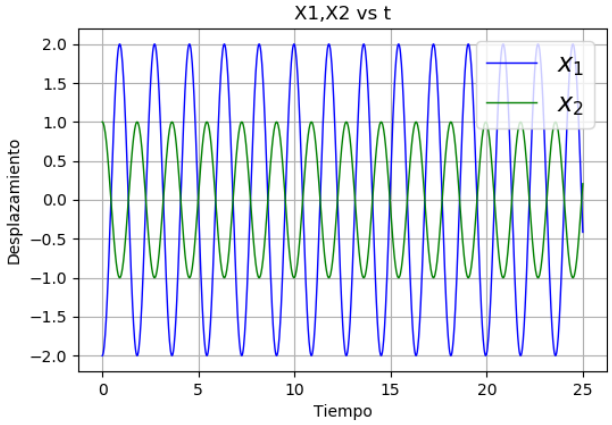
$$(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (-2, 0, 1, 0).$$

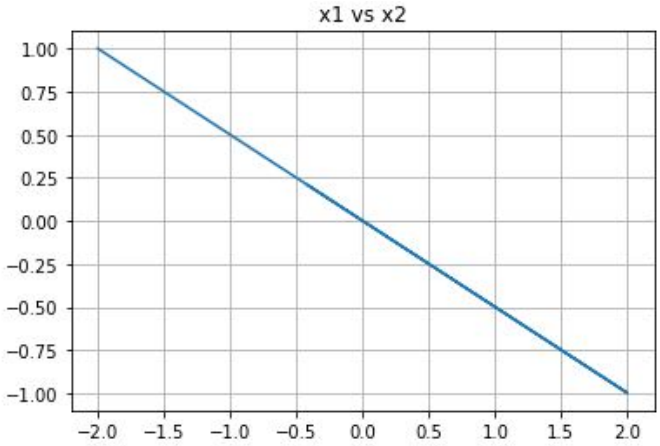
Acontinuacion se muestra el cambio de codigo que se tubo que hacer,recuerde que el vectorfield simepre se realiza en todos los ejemplos:

```
from scipy.integrate import odeint
import numpy as np

# Parameter values
# Masses:
m1 = 1.0
m2 = 1.0
# Spring constants
k1 = 6
k2 = 4.0
# Natural lengths
L1 = 0.0
L2 = 0.0
# Friction coefficients
b1 = 0.0
b2 = 0.0

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = -2
y1 = 0.0
x2 = 1.0
y2 = 0.0

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 25
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]
w0 = [x1, y1, x2, y2]

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

#La solución analitica es:
#X1(t)=-2cos(2*sqrt(3)*t)
#X2(t)=cos(2*sqrt(3)*t)
#El arreglo W1 tiene como columna 0 a x1 y como columna 2 a x2

with open('two_springs2.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],
    np.abs((w1[0]-(-2*np.cos(2*np.sqrt(3)*t1)))/
    (-2*np.cos(2*np.sqrt(3)*t1))), np.abs((w1[2]-(np.cos(2*np.sqrt(3)*t1)))
    /(np.cos(2*np.sqrt(3)*t1))), file=f)
```

Las graficas resultantes son las siguientes:

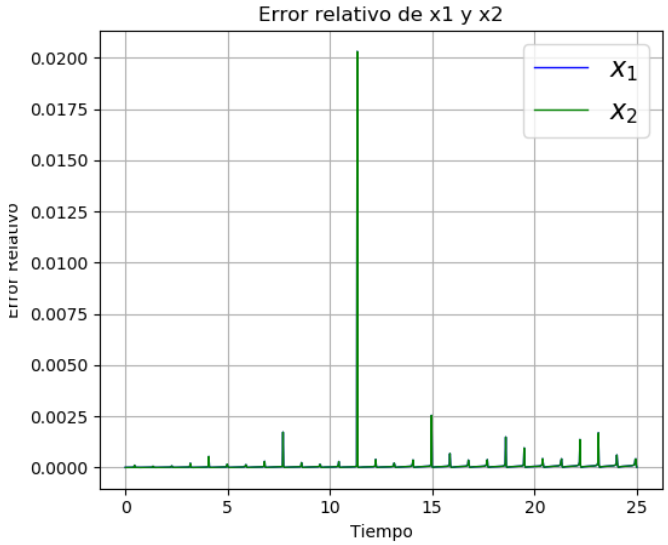




La solucion analitica esta dada por:

$$x_1(t) = -2\cos 2\sqrt{3}t$$
$$x_2(t) = \cos 2\sqrt{3}t$$

La grafica del error relativo de la solicion calculcalada y la que ofrece el articulo es la siguiente:



3.3 Ejemplo 2.3

Describir el movimiento para las constantes del resorte k1=0.4 y k2=1.808 con las condiciones iniciales :

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

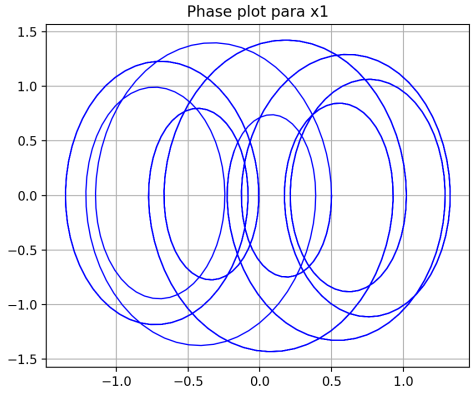
En este ejemplo y en el siguiente no se mostrara el error relativo debido al articulo ya que no muestra la solucion analitica.

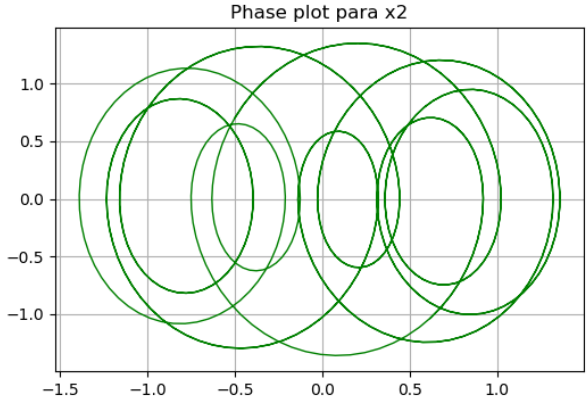
Acontinuacion se muestra el codigo que se modifico:

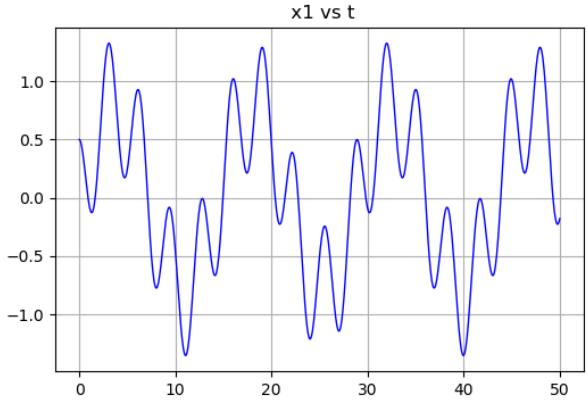
```
# Use ODEINT to solve the differential equations defined by the vector field
from scipy.integrate import odeint
import numpy as np

# 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

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = 1./2.
y1 = 0.0
x2 = -1./2.
y2 = 7./10.

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 50
numpoints = 800
```

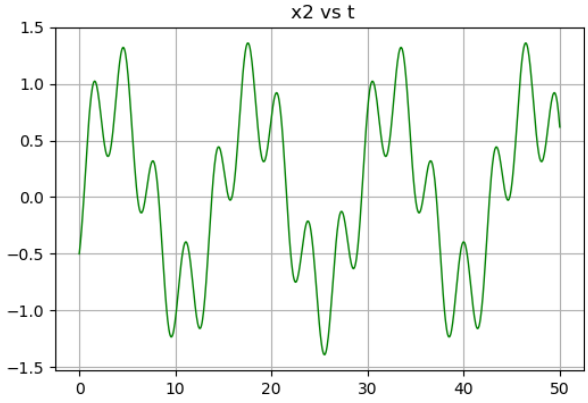
```
# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

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

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

with open('two_springs3.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)
```

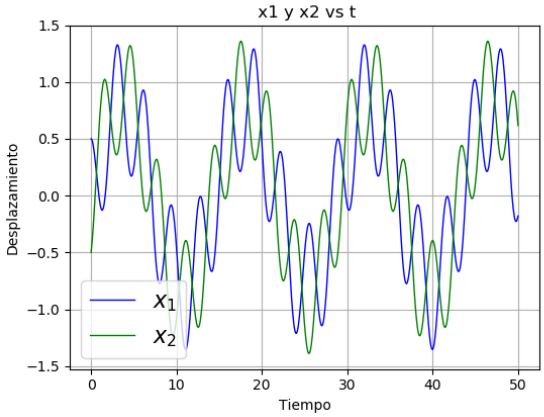
Las graficas que se llevaron acabo en este problema son las siguientes:

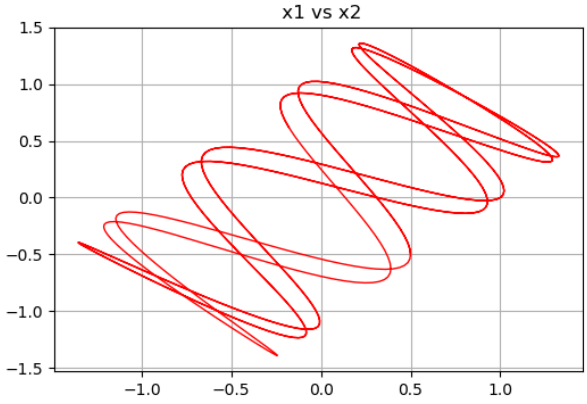



Phase plot para x2
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0


x1 vs t
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40
50


x2 vs t
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50





3.4 Ejemplo 2.4

Describir el movimiento para las constantes del resorte $k1 = 0.4$ y $k2 = 1.808$ asumiendo que $m1 = m2 = 1$, coeficientes de amortiguameinto $b1 = 0.1$ y $b2 = 0.2$ con las condiciones iniciales :

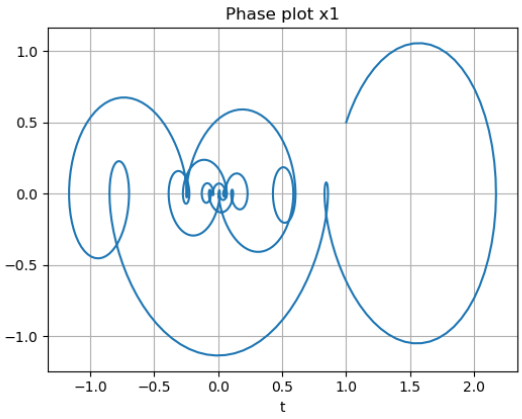
$$(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (1, 1/2, 2, 1/2).$$

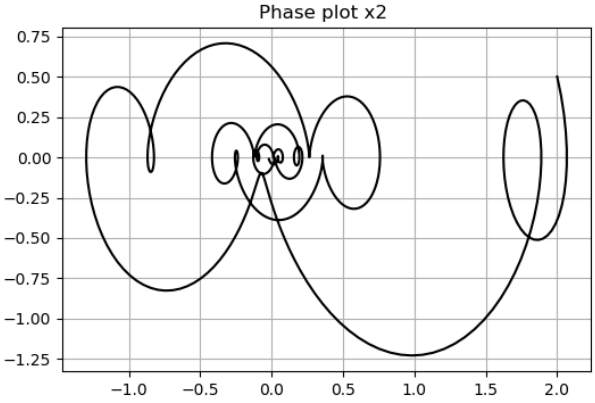
La diferencia que observaran en el sigueinte codigo es que ahora se tomaron encuenta los coeficientes de amortiguamiento

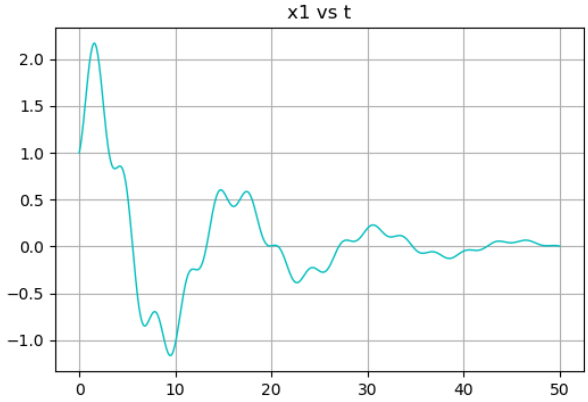
```
# Use ODEINT to solve the differential equations defined by the vector field
from scipy.integrate import odeint

# 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 = 0.1
b2 = 0.2

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = 1
y1 = 1./2.
x2 = 2.
y2 = 1./2.

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 50
numpoints = 800
```

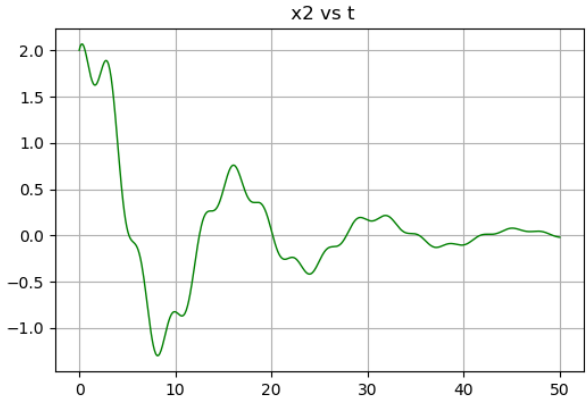
```
# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

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

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

with open('two_springs4.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)
```

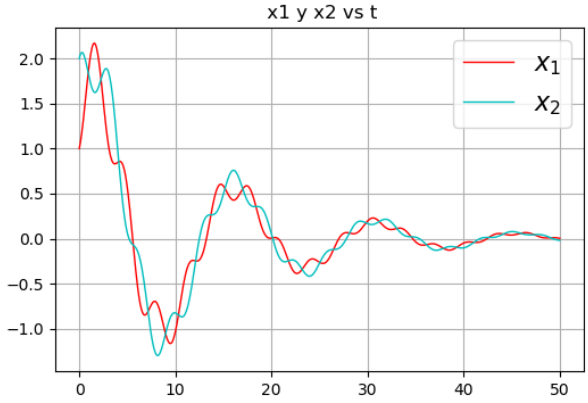
Acontinuacion se motraran las graficas que se llevaron a cabo en este ejemplo:

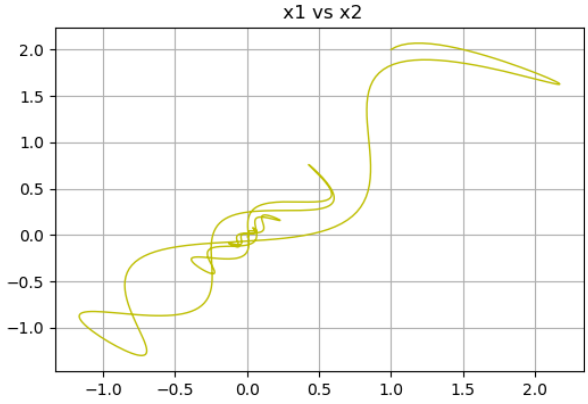



Phase plot x2
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


x1 vs t
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40
50


x2 vs t
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40
50





4 Conclusion

Esta actividad se trabajo con modelados fisicos ,fue un acercamiento a lo que considero un trabajo de un fisico o mas bien aprender a realizar modelados fisicos con ayuda de herramientas computacionales para resolver ecuacion diferenciales que representen dicho modelo ya que es algo que el fisico tiene que hacer dia a dia.

Unas de las observaciones que se puede resalatar al realizar esta practica, son las graficas, como observamos mientras teniamos mas parametros las grafica se fueron haciendo un poco mas ”feas” o no tan lineales como las del ejemplo 1 y 2.

5 Apéndice

¿En general te pareció interesante esta actividad de modelación matemática?
Me gusto mucho ya que me sirvo repasar esos tipos de probemlas sobre la ley de hooke y la materia de ecuaciones diferenciales

¿Qué te gustó mas? ¿Qué no te gustó?
Me gusto mucho el tema escogido ya que es un tema que me gusta mucho ya que es algo que vemos en la vida diaria o que podemos aplicarla.

La cantidad de material te pareció ¿bien?, ¿suficiente?, ¿demasiado?
La cantidad de material me paracio algo bien ya que se nos dio la oportunidad de observar diferentes tipos de graficas

¿Cuál es tu primera impresión de Jupyter Lab?
La verdad me senti muy como con la plataforma muy similar al jupyter notebook.

Respecto al uso de funciones de SciPy, ¿ya habías visto integración numérica en tus cursos anteriores? ¿Cuál es tu experiencia?.
Si, ya habiamos visto integracion numerica en la materia de analisis numerico,lvdd esta muy padre pero no recurdo muy bien mis programas.

El tema de sistema de masas acopladas con resortes, ¿ya lo habías resuelto en tu curso de Mecánica 2?
La verdad no lo recuerdo, pero si recuerdo aver visto la ley de hooke y oscilaciones.

¿Qué le quitarías o agregarías a esta actividad para hacerla más interesante y divertida?
Esta muy bien organizada.