Reporte de Actividad 6

Roberto Benard Orci

19/03/2018

1 Síntesis

En esta actividad usamos el artículo de Temple Fay y Sarah Graham sobre las ecuaciones de resortes acoplados para modelar el movimiento de dos resortes acoplados, colgados en serie desde el techo. Usamos un modelo lineal, que utiliza la Ley de Hooke, el movimiento de cada peso se describe por una ecuación diferencial de cuarto orden.

Las soluciones obtenidas producen una amplia variedad de movimientos, el modelo es adecuado para practicar el modelamiento de ecuaciones diferenciales con herramientas de computo.

Introducción

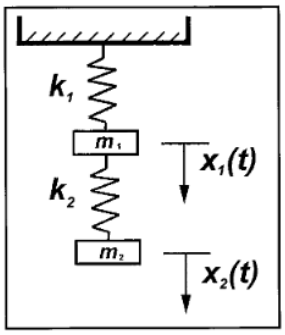
Tenemos dos resortes y dos pesos unidos en serie, colgando del techo. Bajo el supuesto de que las fuerzas de restauración se comportan de acuerdo con Ley de Hooke, este problema de dos grados de libertad está modelado por un par de ecuaciones diferenciales lineales de segundo orden. Al diferenciar y sustituir una ecuación en la otra, se puede mostrar que el movimiento de cada peso esta determinado por una ecuación diferencial lineal de cuarto orden.

Mediante diferentes ejemplos, podemos observar que pequeños cambios en las condiciones iniciales pueden generar cambios significativos en el movimiento de los resortes, movimientos que pueden ser analizados mediante diferentes gráficas.

El modelo de resorte acoplado

El modelo consiste en dos resortes y dos pesos, el primer resorte tiene una constante *k1*, este se adjunta al techo y un peso de masa *m1* está unido al extremo inferior del resorte. A este peso, se le agrega un segundo resorte con constante *k2*, adjunto a este se encuentra un peso de masa *m2*.

Cuando el sistema permanece en equilibrio, medimos el desplazamiento de cada peso *l1* y *l2*. Lo que mediremos serán las posiciones, *x1(t)* y *x2(t)*, que cambiaran con respecto al tiempo.



Se modelan cuatro ejemplos, en los cuales se considera que *m1 = m2 = 1.* Las parámetros de cada ejemplo son los siguientes:

```
Ejm. 2.1
k1=6 ,k2=4,
condiciones iniciales (x1(0),v1(0),x2(0),v2(0))=(1,0,2,0)

Ejm. 2.2
k1=6 ,k2=4,
condiciones iniciales (x1(0),v1(0),x2(0),v2(0))=(-2,0,1,0)

Ejm. 2.3
k1=0.4 ,k2=1.808,
condiciones iniciales (x1(0),v1(0),x2(0),v2(0))=(1/2,0,-1/2,7/10)

Ejm. 2.4
k1=0.4 ,k2=1.808 ,f1=0.1 ,f2=0.2
condiciones iniciales (x1(0),v1(0),x2(0),v2(0))=(1,1/2,2,1/2)
```

Donde *f1* y *f2* en el ejemplo 2.4 son los coeficientes de fricción, y *v1* y *v2* son las velocidades respectivas.

2 Ejemplo Base

Para poder graficar todos los ejemplos usamos como base el código de *Coupled spring-mass system*, en el cual aparecía el siguiente código:

```
def vectorfield(w, t, p):

    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 - L1) + k2 * (x2 - x1 - L2)) / m1,
         y2,
         (-b2 * y2 - k2 * (x2 - x1 - L2)) / m2]
    return f
```

En esta sección de código se definen diferentes vectores, los cuales tendrán diferentes valores de acuerdo a los parámetros que tienen, y la forma de las ecuaciones diferenciales.

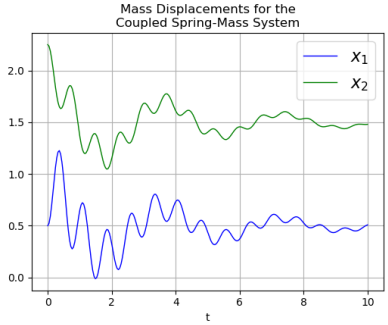
```
from scipy.integrate import odeint

# Parameter values
# Masses:
m1 = 1.0
m2 = 1.5
# Spring constants
k1 = 8.0
k2 = 40.0
# Natural lengths
L1 = 0.5
L2 = 1.0
# Friction coefficients
b1 = 0.8
b2 = 0.5

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = 0.5
y1 = 0.0
x2 = 2.25
y2 = 0.0

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 10.0
numpoints = 250

# 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-springs.dat', 'w') as f:
    # Print & save the solution.
    for t1, w1 in zip(t, wsol):
        print >> f, t1, w1[0], w1[1], w1[2], w1[3]
```

En esta otra sección de código se les da un valor a todos los parámetros, se definen los valores mínimos de los errores de los cálculos, y el número de puntos que se graficaran en un cierto intervalo de tiempo. Luego, con ayuda de *ODEINT* se resuelven las ecuaciones diferenciales que se definieron en la sección de código anterior con los valores que se acaban de establecer, y se guarda una gran lista de datos en el archivo *two-springs.dat*.

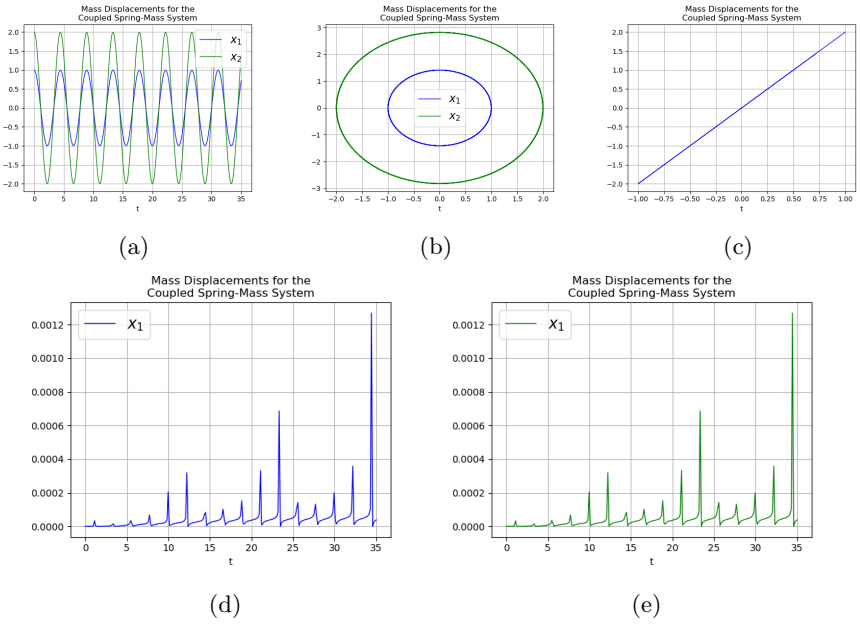
```
from numpy import loadtxt
from pylab import figure, plot, xlabel, grid, hold, legend, title, savefig
from matplotlib.font_manager import FontProperties

t, x1, xy, x2, y2 = loadtxt('two-springs.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
```

Por ultimo, se usa el archivo de datos que se genero en la sección de código anterior para graficar diferentes parámetros. En este ejemplo se graficaron los valores de las posiciones *x1* y *x2* con respecto al tiempo.



3 Resultados

Ejemplo 2.1



(a) (b) (c)

(d) (e)

La figura *(a)* representa a *x1* y *x2* con respecto al tiempo. La figura *(b)* representa a *x1* y *x2* con respecto a *v1* y *v2* respectivamente. La figura *(c)* representa a *x1* con respecto a *x2*. Finalmente, las figuras *(d)* y *(e)* representan los errores relativos de los resultados de las ecuaciones contra los valores verdaderos de la posición con respecto al tiempo.

Estas dos ultimas gráficas fueron hechas agregando esto a la segunda sección

de código:

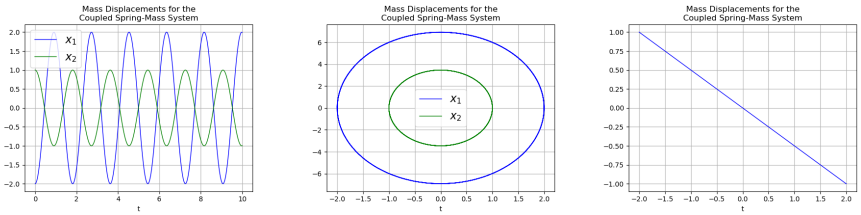
```
print(t1, w1[0], w1[1], w1[2], w1[3], np.abs((w1[0]-(np.cos(np.sqrt(2.0)
*t1)))/(np.cos(np.sqrt(2.0)*t1))), np.abs((w1[2]-(2.0*np.cos(np.sqrt(2.
0)*t1)))/(2.0*np.cos(np.sqrt(2.0)*t1))),file=f)
```

Y esto a la tercera sección de código:

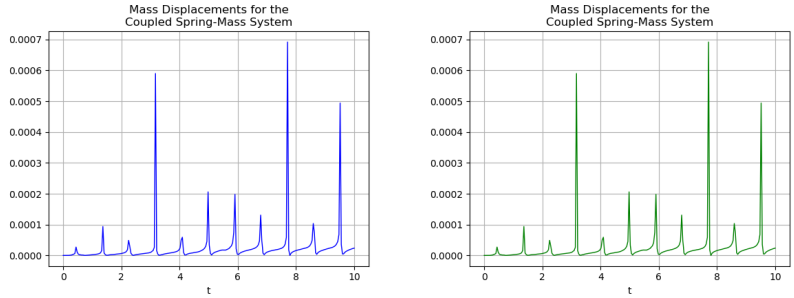
```
t, x1, xy, x2, y2, e1, e2 = loadtxt('two_springs21.dat', unpack=True)
```

Donde *e1* y *e2* son las listas de los valores de los errores relativos con respecto al tiempo.

Ejemplo 2.2



(a) (b) (c)

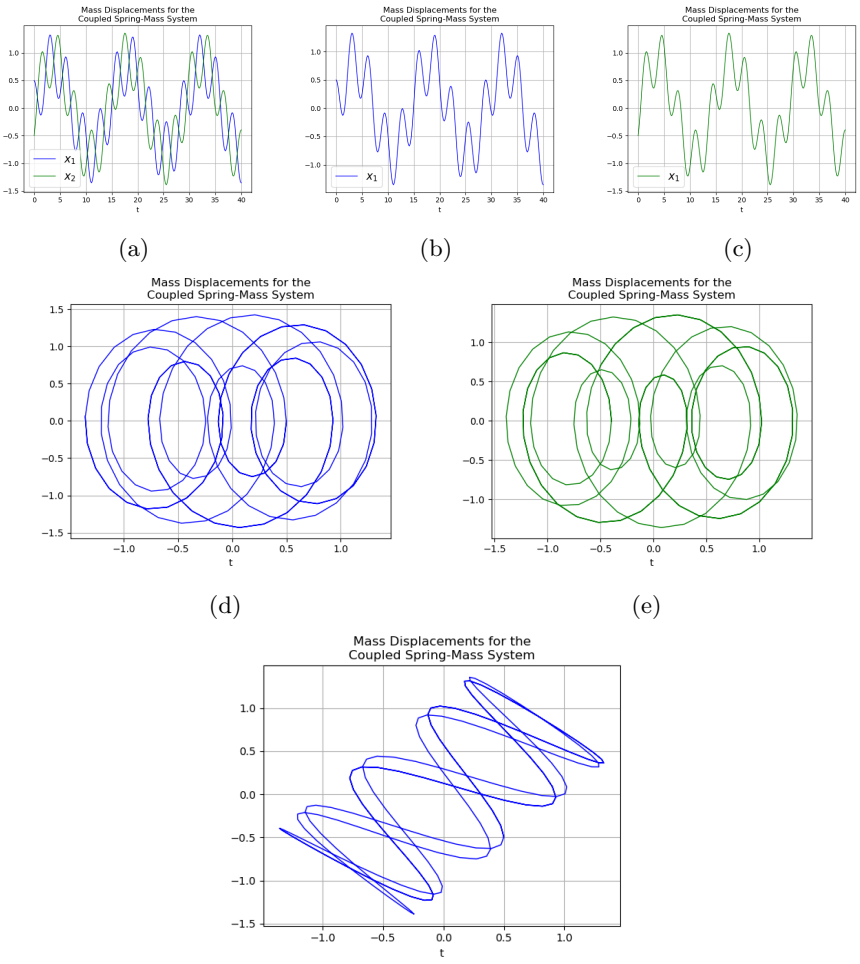


(d) (e)

Cada una de estas gráficas representa lo mismo que en el ejemplo anterior solo que con diferentes parámetros. De igual manera, se agregaron las mismas cosas al código, solo que con las diferencias respectivas de los valores verdaderos de los errores.

Ejemplo 2.3

En este ejemplo y en el siguiente no es posible calcular los errores. La figura *(a)* representa a *x1* y *x2* con respecto al tiempo, pero a diferencia de los anteriores, las figuras *(b)* y *(c)* muestran a *x1* y *x2* respectivamente, con respecto al tiempo,



(a) (b) (c)

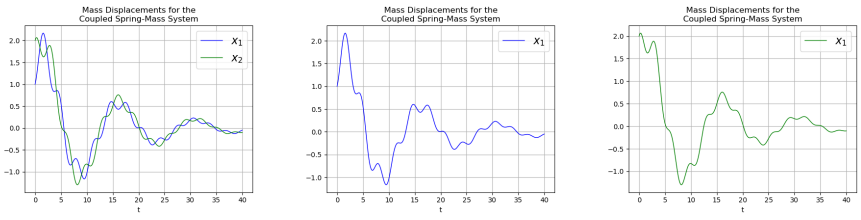
(d) (e)

(f)

para poder visualizar mejor lo que esta pasando. De la misma manera, las figuras *(d)* y *(e)* muestran a *x1* y *x2* con respecto a *v1* y *v2* respectivamente. Finalmente, la figura *(f)* representa a *x1* con respecto a *x2*.

Ejemplo 2.4

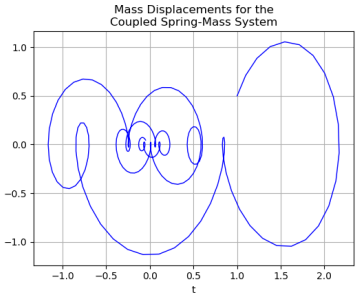
Cada una de estas gráficas representa lo mismo que en el ejemplo anterior solo que con diferentes parámetros. En este ejemplo se implemento la fricción, por lo que se puede ver en las gráficas que el movimiento se va haciendo cada vez mas pequeño.



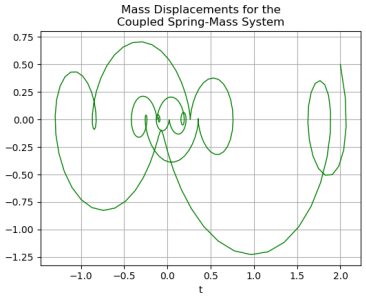
(a)

(b)

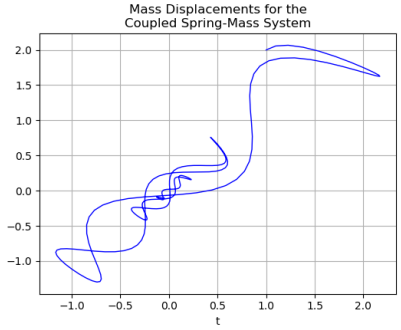
(c)



(d)



(e)



(f)

4 Conclusiones

Al terminar todos los ejemplos, uno puede apreciar lo útiles que son las herramientas de computo para modelar ecuaciones diferenciales. Las soluciones obtenidas producen una amplia variedad de movimientos, cualquier cambio en las condiciones iniciales del modelo, no importa lo pequeños que sean pueden producir grandes cambios en los resultados.

5 Bibliografía

`T. F., & S. G. (2002, September 12). Coupled spring equations [PDF]. Mississippi: University of Southern Mississippi.`

Taylor & Francis

Coupled spring-mass system¶. (n.d.). Retrieved March 18, 2018, from http://scipy-cookbook.readthedocs.io/items/CoupledSpringMassSystem.html

Integration and ODEs (scipy.integrate)¶. (n.d.). Retrieved March 18, 2018, from https://docs.scipy.org/doc/scipy/reference/integrate.html

6 Apéndice

1.- ¿En general te pareció interesante esta actividad de modelación matemática? ¿Qué te gustó mas? ¿Qué no te gustó?

Me pareció muy buena introducción al uso de herramientas de computo para modelar ecuaciones diferenciales. Me gusto mucho el poder visualizar los resultados mediante distintas gráficas.

2.- La cantidad de material te pareció ¿bien?, ¿suficiente?, ¿demasiado?

Me hubiera gustado entender un poco mejor el lo que estábamos modelando, de donde surgían las ecuaciones que se usaron y lo que hacia cada parte del código que usamos. La mayoría de estas cosas aparecían en el archivo de Fay y Graham, pero me hubiera gustado discutirlo en clase.

3.- ¿Cuál es tu primera impresión de Jupyter Lab?

Me parece una muy buena herramienta para resolver distintas ecuaciones.

4.- Respecto al uso de funciones de SciPy, ¿ya habías visto integración numérica en tus cursos anteriores? ¿Cuál es tu experiencia?.

Si, en análisis numérico. Siento que es una muy buena manera de saber a donde quieres llegar con otros métodos.

5.- El tema de sistema de masas acopladas con resortes, ¿ya lo habías resuelto en tu curso de Mecánica 2?

Si, pero solo nos mostraron como se podría resolver con ecuaciones diferenciales.

6.- ¿Qué le quitarías o agregarías a esta actividad para hacerla más interesante y divertida?

Solo explicar en el salón como funcionaba el ejemplo de código que usamos. Como podríamos crear un código así desde cero, que deberíamos de tener en cuenta.