Reporte de Actividad 7: Sistema de Resortes Acoplados

Jesús Antonio González Espinosa

Física Computacional 1

Sábado, 24 de Marzo del 2018

1 Introducción

Para esta actividad hemos trabajado con modelación de fenómenos físicos en python; específicamente en el de oscilaciones, presentando varios ejemplos de sistemas de resortes acoplados.

En este reporte se continua trabajando con la síntesis del artículo "Coupled spring equations" de TEMPLE H. FAY, pero ahora con la adición de los últimos dos subtemas, el cual nos aporta la teoría necesaria para obtener las bases y comprender los conceptos importantes sobre sistemas de masas con oscilaciones no lineales y forzadas. Similarmente a la actividad pasada, el documento aporta ejemplos, presentando las propiedades y condiciones iniciales de los resortes para poder crear las gráficas presentadas en este reporte. Naturalmente, se presenta el cierre al reporte, incluyendo bilbiografía, conclusiones y el apéndice perteneciente a la actividad.

2 Sintesis

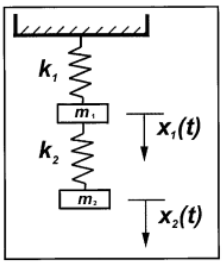
2.1 Introducción

En éste artículo se investiga el problema de los dos resortes y dos pesas adjuntas en serie. Bajo la suposición de que las fuerzas de restauración se comportan de acuerdo a la Ley de Hooke, este problema de dos grados de libertad es modelado por un par de ecuaciones diferenciales lineales de segundo orden que al sustituir una en la otra, el movimiento de cada pesa puede ser determinado por una ecuación diferencial lineal de cuarto orden.

Lo que hace interesante a este problema, es que podemos investigar los movimientos de las masas, para ver si están sincronizadas u opuestas entre ellas. También se puede ver que al agregarle factores no lineares, pueden surgir movimientos interesantes. Al modificar los parámetros podemos observar gráficamente los cambios en la periodicidad, la amplitud, la fase, la sensitividad a loas condiciones iniciales, entre otras propiedades.

2.2 El modelo de resorte acoplado

El modelo consiste en dos resortes y dos pesas. Un resorte tiene la constante elástica k_1 y está adjunto al techo de un extremo y del otro tiene una pesa de masa m_1. La misma pesa tiene adjunto un segundo resorte de constante elástica k_2 y al final de éste, una pesa de masa m_2. Al dejar el sistema llegar al equilibrio, medimos el desplazamiento de cada masa, en función del tiempo, y llamaremos estas medidas como: $x_1(t)$ y $x_2(t)$.



2.2.1 Asumiendo la Ley de Hooke

Asumiendo que hay pequeñas oscilaciones, las fuerzas restauradoras son de la forma $-k_1 l_1$ y $-k_2 l_2$, donde l_1 y l_2 son las elongaciones o compresiones de los dos resortes. Como la masa superior está unida a ambos resortes, ésta siente las dos fuerzas restauradoras actuando sobre ella, mientras que la segunda masa solo siente la fuerza restauradora del segundo resorte. Asumiendo que no hay fuerzas de amortiguamiento, la Ley de Newton implica que las ecuaciones que representan los movimientos de las pesas son:

$$m_1\ddot{x}_1 = -k_1 x_1 - k_2(x_1 - x_2)$$
$$m_1\ddot{x}_2 = -k_2(x_1 - x_2)$$

Entonces para encontrar una ecuación para x_1 que no involucre a x_2, resolvemos la primera ecuación diferencial para x_2, y la sustituimos en la primera, donde después de simplificar, obtenemos:

$$m_1 m_2 x_1^{(4)} + (m_2 k_1 + k_2(m_1 + m_2))\ddot{x}_1 + k_1 k_2 x_1 = 0$$

Ahora, para encontrar una ecuación que involucre de x_2, resolvemos la segunda ecuación diferencial para x_1 y lo sustituimos en la primera ecuación, obteniendo:

$$m_1 m_2 x_2^{(4)} + (m_2 k_1 + k_2(m_1 + m_2))\ddot{x}_1 + k_1 k_2 x_2 = 0$$

Para poder resolver las ecuaciones, solo es necesario las posiciones iniciales y las velocidades iniciales.

2.2.2 Algunos ejemplos con masas idénticas

Consideremos el modelo con dos pesas con la misma masa, $m_1 = m_2 = 1$, ignorando el amortiguamiento y sin fuerzas externas.

Ejemplo 2.1: Describe el movimiento para las constantes de resorte $k_1 = 6$ y $k_2 = 4$ con las condiciones iniciales $(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (1, 0, 2, 0)$. Al resolverlo de manera analítica obtenemos:

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

El movimiento es sincronizado, por lo tanto, las pesas se mueven en fase una con la otra, teniendo el mismo periodo, pero con amplitudes un poco diferentes.

Para poder grafciar esto en Python, utilizamos un código proporcionado. Este código se utilizó durante toda la actividad, solo modificando los parámetros y condiciones iniciales; así como las variables de los ejes coordenados para presentar las diferentes gráficas. En este caso, el código para este ejemplo se ve así:

```
#************************************************ Ejemplo 2.1 ***********
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 - L1) + k2 * (x2 - x1 - L2)) / m1,
         y2,
         (-b2 * y2 - k2 * (x2 - x1 - L2)) / 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
import math

# 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 = 50.0
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('ejemplo_2.1.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], np.abs((w1[0]-(np.cos(np.sqrt(2)*t1)))/(np.cos(np.sqrt(2)*t1))))
```

```python
# Plot the solution that was generated

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

t, x1, y1, x2, y2, e1, e2 = loadtxt('ejemplo_2.1.dat', unpack=True)

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

xlabel('t')
ylabel('x')
grid(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('Ejemplo 2.1: Posición con respecto al Tiempo')
savefig('ejemplo_2.1.1.png', dpi=100)
```

Obteniendo las siguientes gráficas:

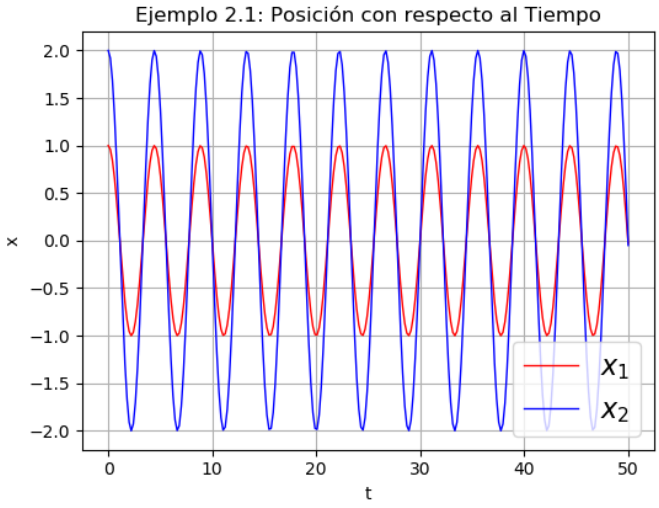
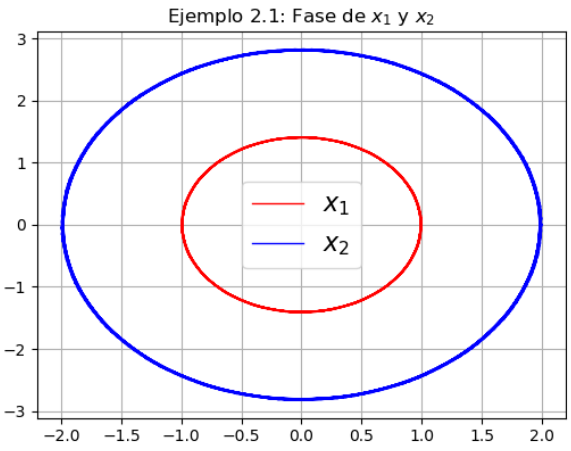
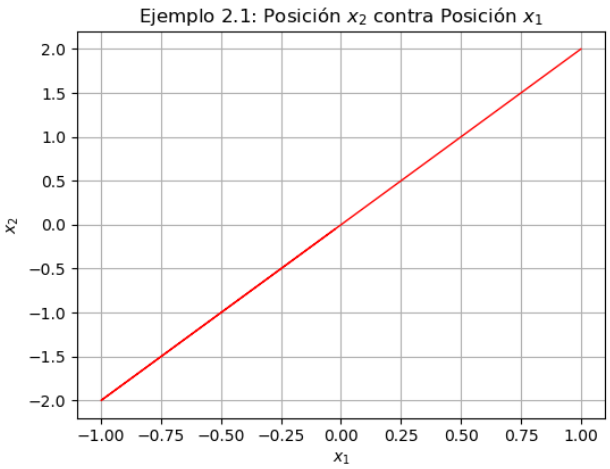


Figure 1: Posición con respecto al Tiempo de x_1 y x_2

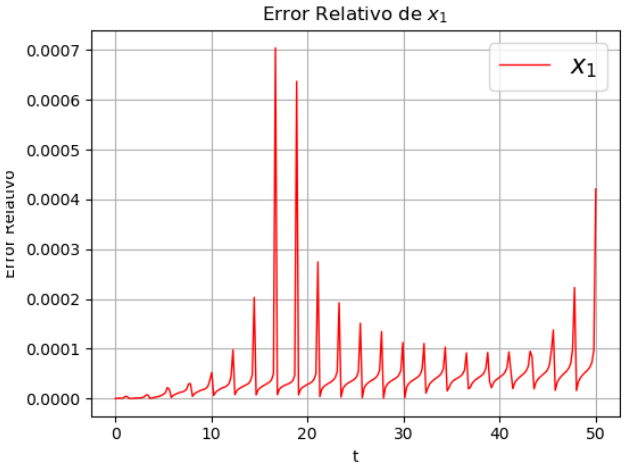


(a) Fase de x_1 y x_2

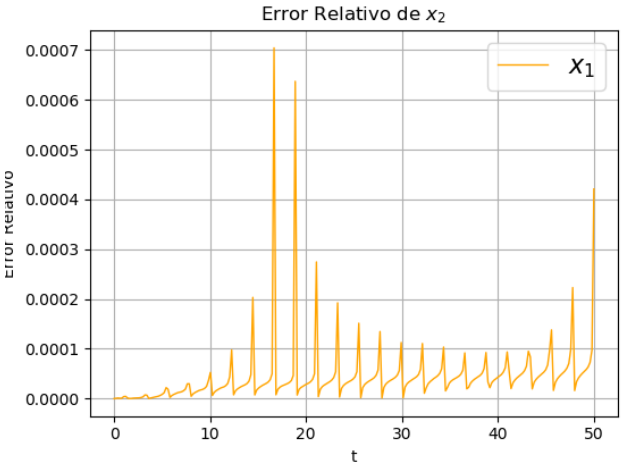


(b) Posición X_2 contra x_1

El error relativo de x_1 y x_2:



(a) Error Relativo de x_1



(b) Error Relativo de x_2

Ejemplo 2.2: Describe el movimiento para las constantes de resorte $k_1 = 6$ y $k_2 = 4$ con las condiciones iniciales $(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (-2, 0, 1, 0)$. Al resolverlo de manera analítica obtenemos:

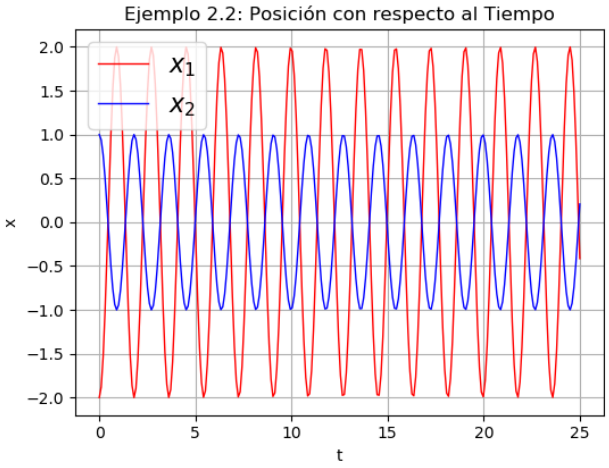
$$x_1(t) = -2cos2\sqrt{3}t$$
$$x_2(t) = cos2\sqrt{3}t$$

En este caso, cuando la primera pesa se mueve hacia arriba, la segunda se mueve hacia abajo; aún tienen el mismo periodo, pero ahora un desfase de 180°

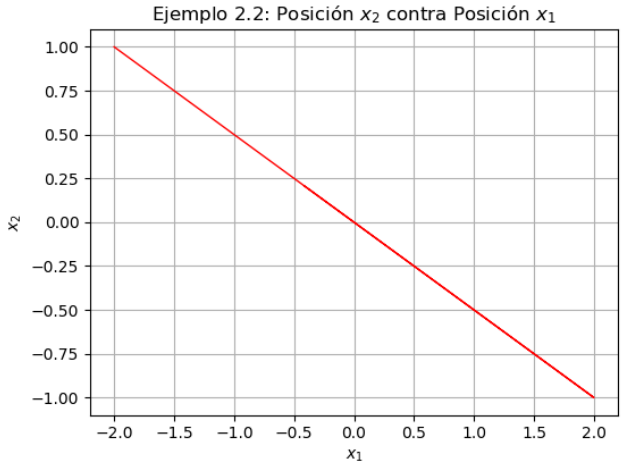
El código modificado para hacer tales gráficas resulta así:

Obteniendo las siguientes gráficas:

```
# 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 = -2.0
y1 = 0.0
x2 = 1.0
y2 = 0.0

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 25.0
numpoints = 250
```

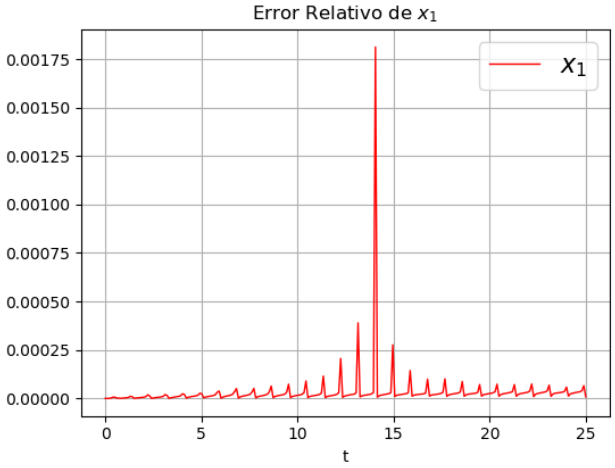


(a) Posición con respecto al Tiempo de x_1 y x_2

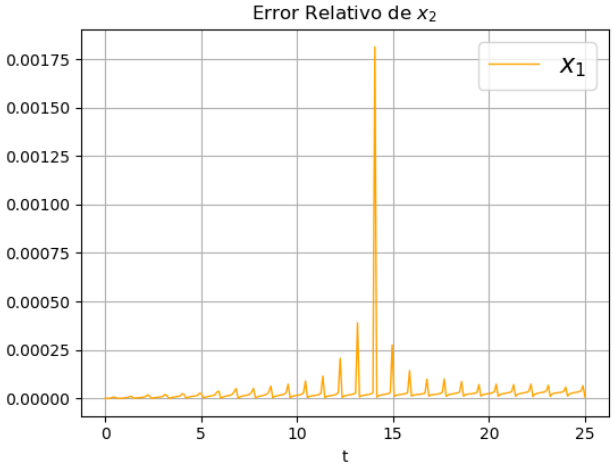


(b) Posición x_2 contra x_1

El error relativo de x_1 y x_2:



(a) Error Relativo de x_1



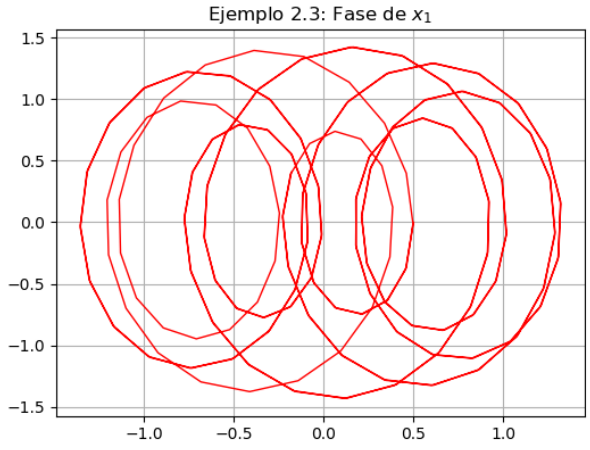
(b) Error Relativo de x_2

Ejemplo 2.3: Describe el movimiento para las constantes de resorte $k_1 = 0.4$ y $k_2 = 1.808$ con las condiciones iniciales $(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (1/2, 0, -1/2, 7/10)$.

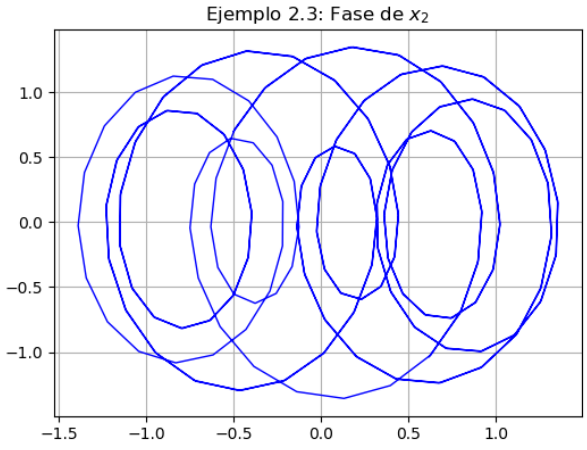
En este problema, los valores de las constantes del resorte determinan el periodo y por lo tanto, la frecuencia; mientras que las condiciones iniciales solo afectan la amplitud y la fase de las soluciones. El código modificado para hacer tales gráficas resulta así:

```
# 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 = 1.0/2.0
y1 = 0.0
x2 = -1.0/2.0
y2 = 7.0/10.0

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 50.0
numpoints = 250
```

Al graficar las soluciones, podemos ver varios patrones interesantes:



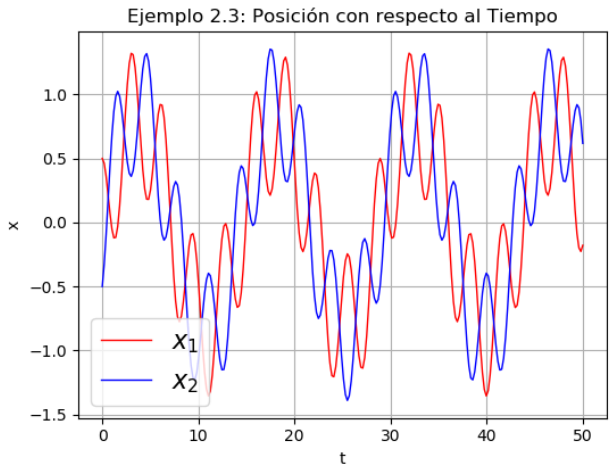
(a) Fase para x_1



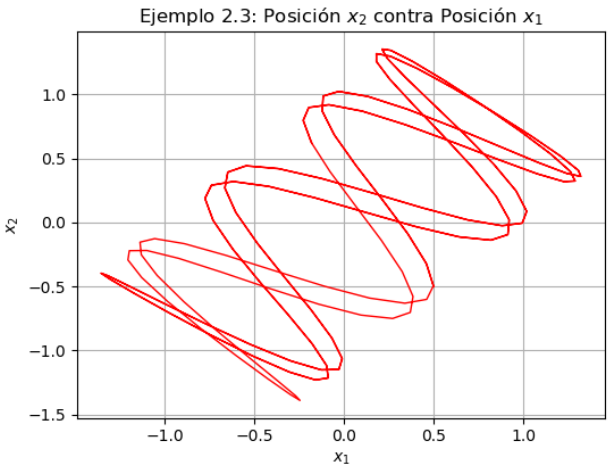
(b) Fase para x_2

(c) Posición con respecto al tiempo de x_1

(d) Posición con respecto al tiempo de x_2



(a) Posición con respecto al tiempo de x_1 y x_2



(b) Posición x_2 contra x_1

2.2.3 Amortiguamiento

El tipo de amortiguamiento más común, son los de viscosidad, donde la fuerza amortiguadora es proporcional a la velocidad. El amortiguamiento de la primera masa depende solamente de su velocidad y no de la velocidad de la segunda masa y vice versa. Para modelar esto, agregamos los términos $-\delta_1\dot{x}_1$ a la primera ecuación y $-\delta_1\dot{x}_2$ a la segunda. Asumiendo que los coeficientes de amortiguamiento δ_1 y δ_2 son pequeños, el modelo se da por:

$$m_1\ddot{x}_1 = -\delta_1\dot{x}_1 - k_1x_1 - k_2(x_1 - x_2)$$
$$m_2\ddot{x}_2 = -\delta\dot{x}_2 - k_2(x_2 - x_1)$$

Ahora, para hacer una ecuación que solo dependa de x_2, podemos resolver la segunda ecuación para x_1 y sustituirlo en la primera, obteniendo una ecuación de cuarto orden:

$$m_1m_2x_2^{(4)} + (m_1\delta_1 + m_2\delta_2)\dddot{x}_2 + (m_2k_1 + k_2(m_1 + m_2) + \delta_1\delta_2)\ddot{x}_2 + (k_1\delta_2 + k_2(\delta_1 + \delta_2)\dot{x}_2 + k_1k_2x_2 = 0$$

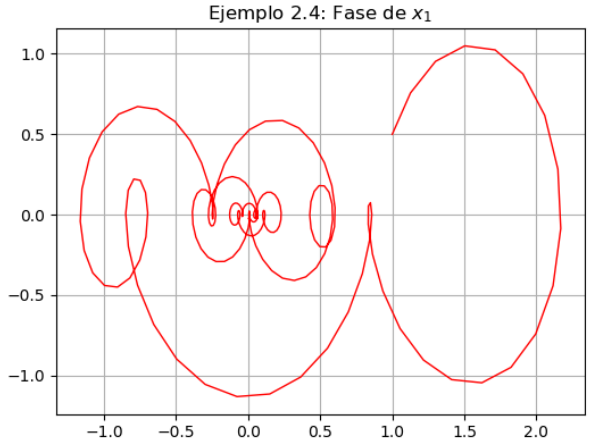
Podemos hacer el mismo procedimiento para que dependa de x_1. Con esto, obtenemos una ecuación diferencial lineal que representa el movimiento de ambas pesas.

Ejemplo 2.4: Asuma que $m_1 = m_2 = 1$. Describe el movimiento para las constantes de resorte $k_1 = 0.4$ y $k_2 = 1.808$, los coeficientes de amortiguamiento $\delta_1 = 0.1$ y $\delta_2 = 0.2$ con las condiciones iniciales $(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (1, 1/2, 2, 1/2)$.

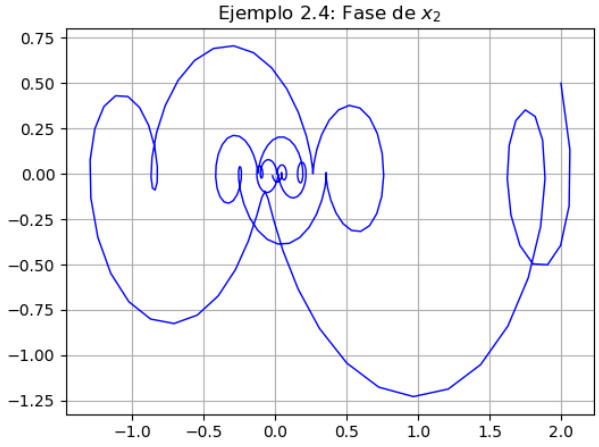
Se puede observar un patrón regular en donde el movimiento avanza con una disminución en su amplitud. El código para graficar las soluciones resulta así:

```
# 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 = 1.0
y1 = 1.0/2.0
x2 = 2.0
y2 = 1.00/2.00

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 50.0
numpoints = 250
```

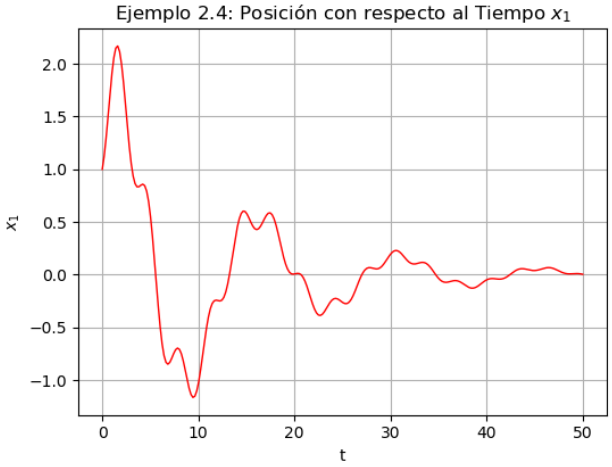
Resultando en las siguientes graficas



(a) Fase para x_1

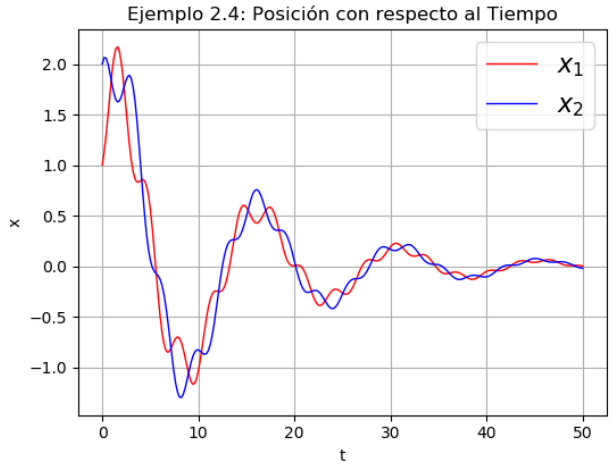


(b) Fase para x_2

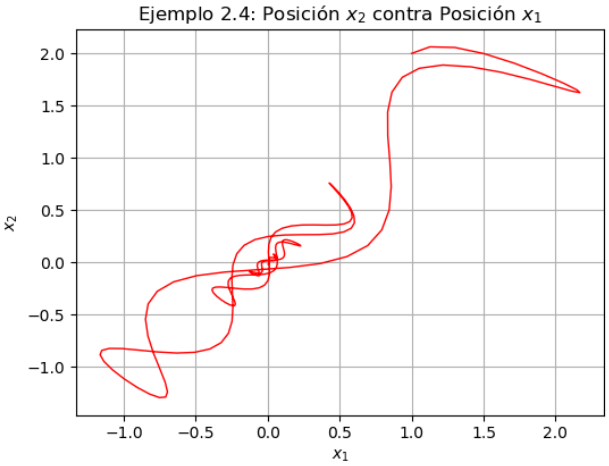


(c) Posición con respecto al tiempo de x_1

(d) Posición con respecto al tiempo de x_2



(e) Posición con respecto al tiempo de x_1 y x_2



(f) Posición x_2 contra x_1

2.3 Agregando No Linealidad

Si asumimos que las fuerzas restauradoras son no lineales, podemos acomodar el modelo para que encaje a esto. Asumiendo que la fuerza restauradora tienen la forma de $-kx + \mu x^3$. Entonces el modelo se convierte en:

$$m_1\ddot{x}_1 = -\delta_1\dot{x}_1 - k_1x_1 + \mu_1x_1^3 - k_2(x_1 - x_2) + \mu_2(x_1 - x_2)^3$$
$$m_2\ddot{x}_2 = -\delta\dot{x}_2 - k_2(x_2 - x_1) + \mu_2(x_2 - x_1)^3$$

Los rangos de movimiento para los modelos no lineales son mucho más complicados que los lineales; la exactitud cada vez se vuelve más cuestionable al resolver las ecuaciones, no hay solución numérica que se mantenga precisa.

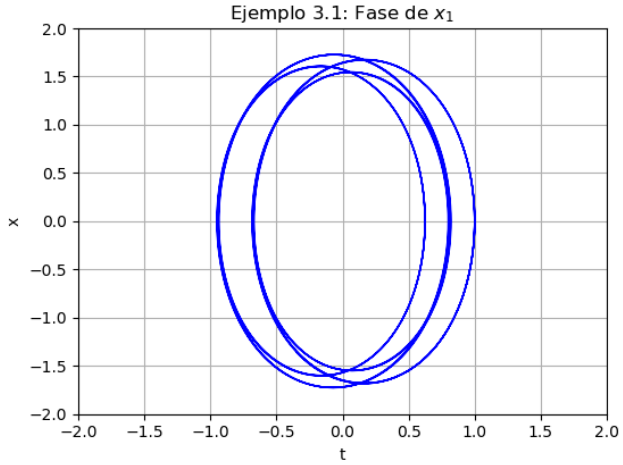
Ejemplo 3.1: Asuma que $m_1 = m_2 = 1$. Describe el movimiento para las constantes de resorte $k_1 = 0.4$ y $k_2 = 1.808$, los coeficientes de amortiguamiento $\delta_1 = 0$ y $\delta_2 = 0$, coeficientes no lineales $\mu_1 = -1/6$ y $\mu_2 = -1/10$ con las condiciones iniciales $(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (1, 0, -1/2, 0)$. En este ejemplo no hay amortiguamiento, entonces los movimientos parecen ser periódicos. Por la no linealidad, el modelo puede mostrar sensitividad a las condiciones iniciales.

Para poder graficar esto, tenemos que retomar el código que hemos estado utilizando, pero con unas modificaciones para poder agregar la no linealidad. Después de los cambios, el código resulta así:

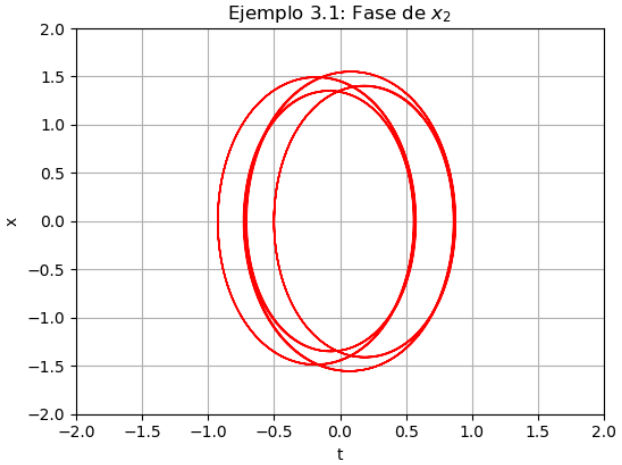
```
#************************************************** Ejemplo 3.1 **************************************************
def vectorfield(w, t, p):
    """
    Defines the differential equations for the coupled spring-mass system.

    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,mu1,mu2]
    """
    x1, y1, x2, y2 = w
    m1, m2, k1, k2, L1, L2, b1, b2, mu1, mu2 = p

    # Create f = (x1',y1',x2',y2'):
    f = [y1,
         (-b1 * y1 - k1 * (x1 - L1) + mu1*(x1 - L1)**3 - k2 * (x1 - x2 - L2) + mu2 * (x1 - x2 - L2)**3) / m1,
         y2,
         (-b2 * y2 - k2 * (x2 - x1 - L2) + mu2 * (x2 - x1 - L2)**3) / m2]
    return f
```

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


# 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
mu1 = -1/6
mu2 = -1/10

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = 1.0
y1 = 0.0
x2 = -1/2
y2 = 0.0
# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 50.0
numpoints = 2500

# 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, mu1, mu2]
w0 = [x1, y1, x2, y2]

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

with open('ejemplo_3.1.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)
```

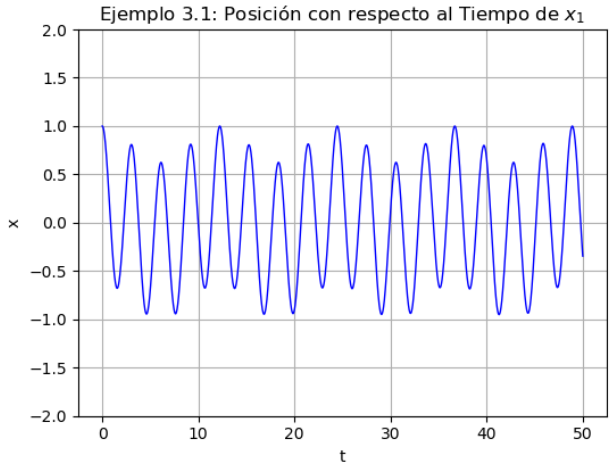
Y similarmente a los ejemplos anteriores, graficamos, obteniendo los siguientes resultados:



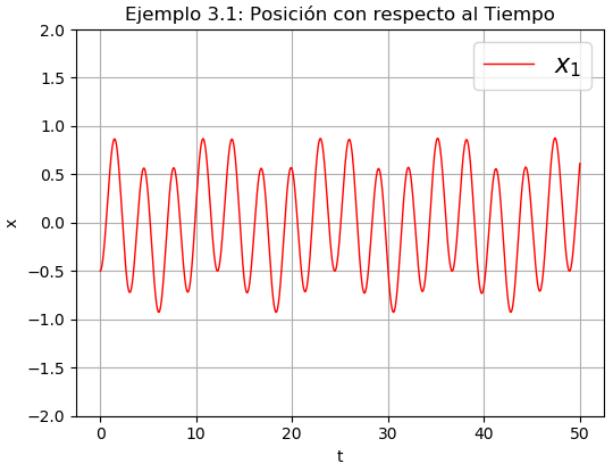
(a) Fase para x_1



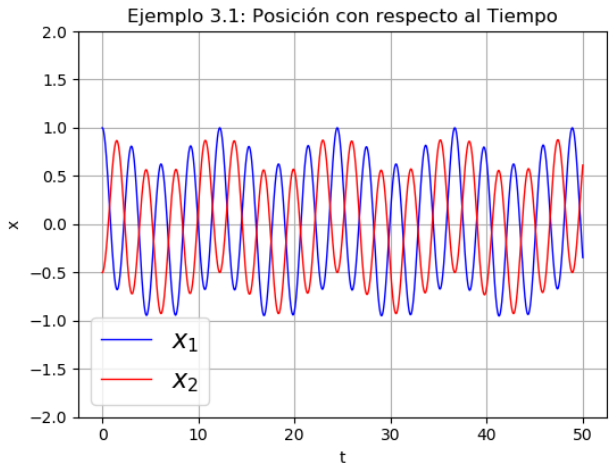
(b) Fase para x_2



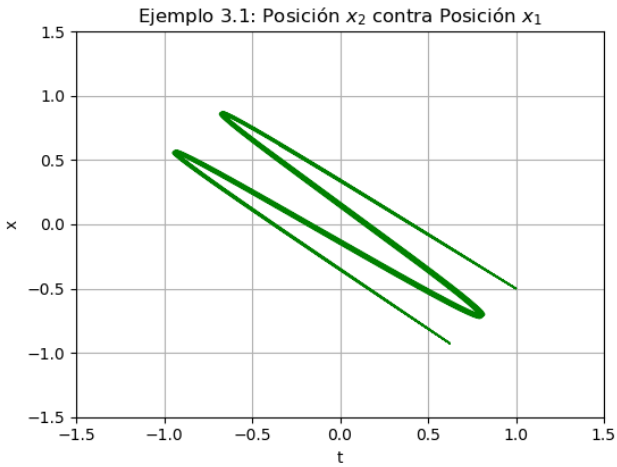
(c) Posición con respecto al tiempo de x_1



(d) Posición con respecto al tiempo de x_2



(a) Posición con respecto al tiempo de x_1 y x_2

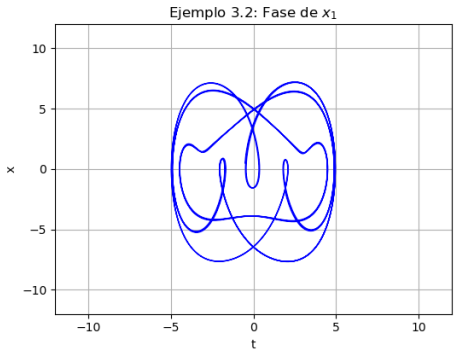


(b) Posición x_2 contra x_1

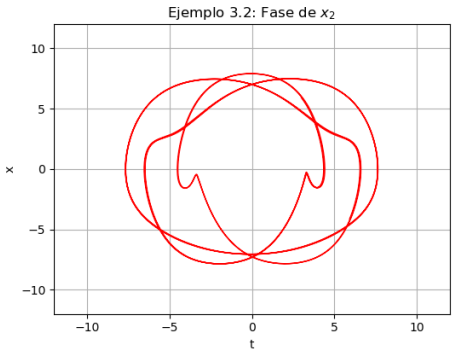
Ejemplo 3.2: Asuma que $m_1 = m_2 = 1$. Describe el movimiento para las constantes de resorte $k_1 = 0.4$ y $k_2 = 1.808$, los coeficientes de amortiguamiento $\delta_1 = 0$ y $\delta_2 = 0$, coeficientes no lineales $\mu_1 = -1/6$ y $\mu_2 = -1/10$ con las condiciones iniciales
$(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (-0.5, 1/2, 3.001, 5.9)$.
Agregando las condiciones iniciales y propiedades del sistema:

```
# 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
mu1 = -1/6
mu2 = -1/10

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = -0.5
y1 = 1/2
x2 = 3.001
y2 = 5.9

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 50.0
numpoints = 2500
```

Con lo que obtenemos las siguientes gráficas:



(a) Fase de x_1



(b) Fase de x_2

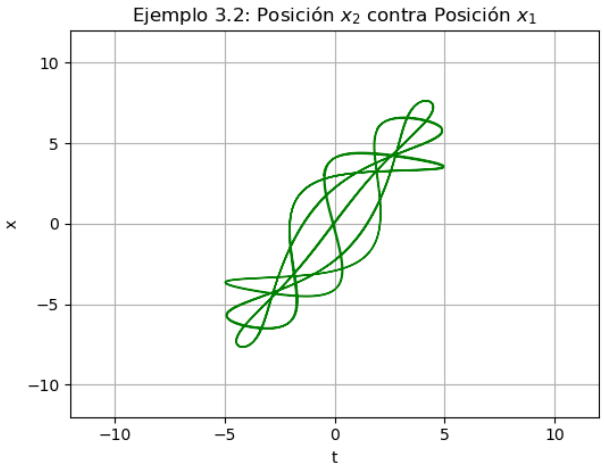
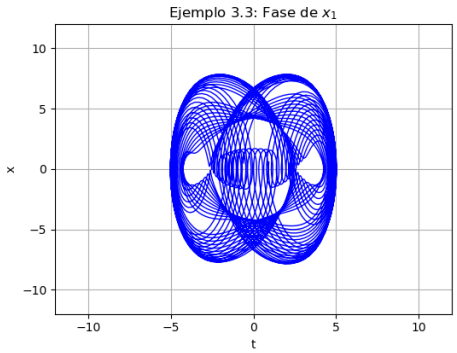


Figure 12: Posición X_2 contra x_1

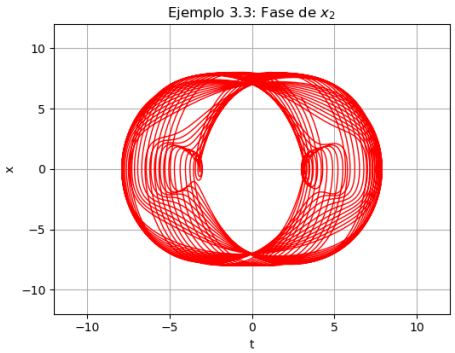
Ejemplo 3.3: Asuma que $m_1 = m_2 = 1$. Describe el movimiento para las constantes de resorte $k_1 = 0.4$ y $k_2 = 1.808$, los coeficientes de amortiguamiento $\delta_1 = 0$ y $\delta_2 = 0$, coeficientes no lineales $\mu_1 = -1/6$ y $\mu_2 = -1/10$ con las condiciones iniciales
$(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (-0.6, 1/2, 3.001, 5.9)$.
Agregando las condiciones iniciales y propiedades del sistema:

```
# 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
mu1 = -1/6
mu2 = -1/10

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = -0.6
y1 = 1/2
x2 = 3.001
y2 = 5.9

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 200.0
numpoints = 4000
```

Con lo que obtenemos las siguientes gráficas:



(a) Fase de x_1



(b) Fase de x_2

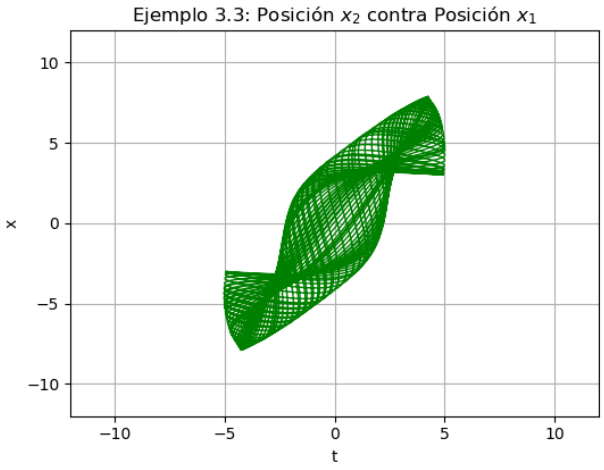


Figure 14: Posición x_2 contra x_1

2.4 Agregando Forzamiento

Agregarle fuerza externa al modelo es muy simple, suponga que asumimos un forzamiento sinusoidal simple, de la forma $Fcos(\omega t)$. Entonces el modelo queda:

$$m_1\ddot{x}_1 = -\delta_1\dot{x}_1 - k_1x_1 + \mu_1x_1^3 - k_2(x_1 - x_2) + \mu_2(x_1 - x_2)^3 + F_1cos\omega_1 t$$
$$m_2\ddot{x}_2 = -\delta\dot{x}_2 - k_2(x_2 - x_1) + \mu_2(x_2 - x_1)^3 + F_2cos\omega_2 t$$

Los rangos de movimiento para modelos no lineales con forzamiento son muy vastos, podemos encontrar resonancia no lineal, soluciones armónicas, soluciones armónicas y soluciones periódicas en estado estable. Las condiciones en las que esto ocurre no son fáciles de declarar.

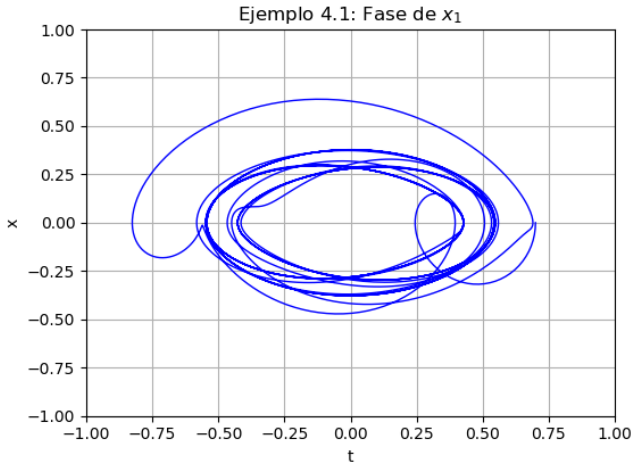
Ejemplo 4.1: Asuma que $m_1 = m_2 = 1$. Describe el movimiento para las constantes de resorte $k_1 = 2/5$ y $k_2 = 1$, los coeficientes de amortiguamiento $\delta_1 = 1/10$ y $\delta_2 = 1/5$, coeficientes no lineales $\mu_1 = 1/6$ y $\mu_2 = 1/10$, amplitudes de forzamiento $F_1 = 1/3$ y $F_2 = 1/5$, y frecuencias de forzamiento $\omega_1 = 1$ y $\omega_2 = 3/5$, con las condiciones iniciales
$(x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0)) = (0.7, 0, 0.1, 0)$.
Por el amortiguamiento, podemos esperar comportamientos diferentes con valores pequeños de tiempo y comportamientos de estado estable para valores grandes de tiempo.

De nuevo, el código necesita modificarse para poder agregar el forzamiento. Después de los cambios, el código resulta así:

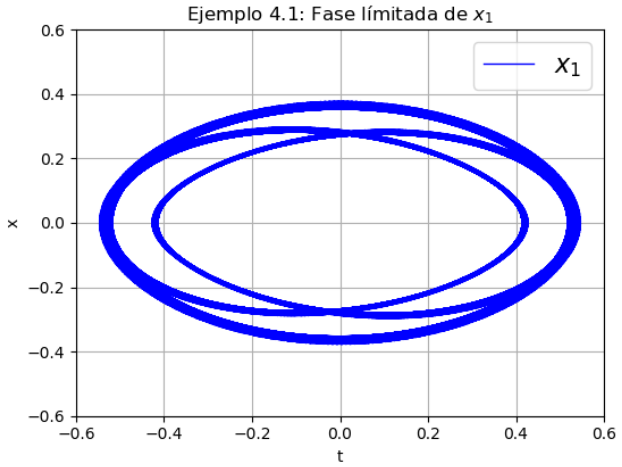
```
################################################## Ejemplo 4.1 ##################################################
import numpy as np
def vectorfield(w, t, p):
    """
    Defines the differential equations for the coupled spring-mass system.

    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,m1,m2,f1,f2,w1,w2]
    """
    x1, y1, x2, y2 = w
    m1, m2, k1, k2, L1, L2, b1, b2, mu1, mu2, f1, f2, om1, om2 = p

    # Create f = (x1',y1',x2',y2'):
    f = [y1,
         (-b1 * y1 - k1 * (x1 - L1) + mu1 * (x1 - L1) ** 3 - k2 * (x1 - x2 - L2) + mu2 * (x1 - x2 - L2) ** 3 + f1 * (np.cos(om1 * t))) / m1,
         y2,
         (-b2 * y2 - k2 * (x2 - x1 - L2) + mu2 * (x2 - x1 - L2) ** 3 + f2 * (np.cos(om2 * t))) / m2]
    return f
```

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


# Parameter values
# Masses:
m1 = 1.0
m2 = 1.0
# Spring constants
k1 = 2.0/5.0
k2 = 1.0
# Natural lengths
L1 = 0.0
L2 = 0.0
# Friction coefficients
b1 = 1.0/10.0
b2 = 1.0/5.0
mu1 = 1.0/6.0
mu2 = 1.0/10.0
f1 = 1.0/3.0
f2 = 1.0/5.0
om1 = 1.0
om2 = 3.0/5.0

# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
x1 = 0.7
y1 = 0
x2 = 0.1
y2 = 0

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 200
numpoints = 3000

# 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, mu1, mu2, f1, f2, om1, om2]
w0 = [x1, y1, x2, y2]

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

with open('ejemplo_4.1.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)# Plot the solution that was generated
```

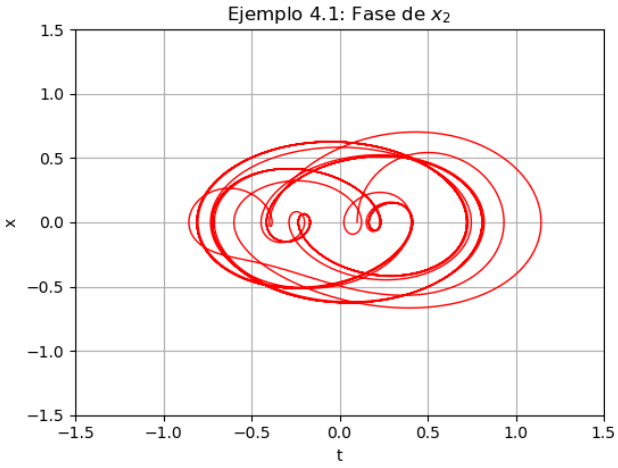
Obteniendo las siguientes gráficas:



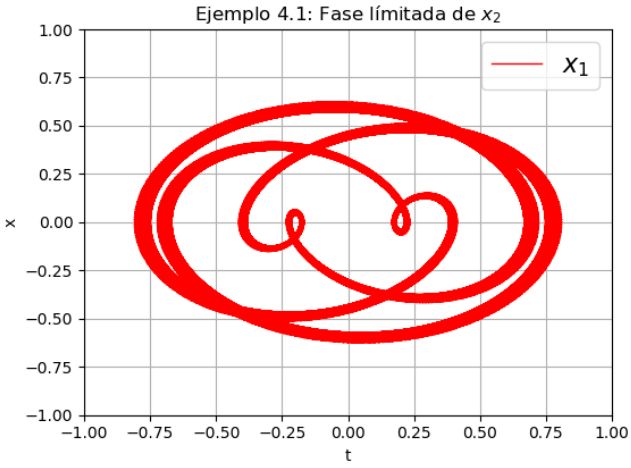
(a) Fase de x_1



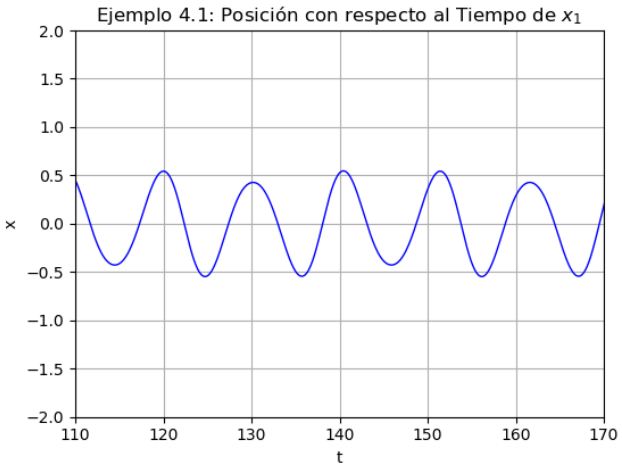
(b) Ciclo limitado para x_1



(c) Fase de x_2

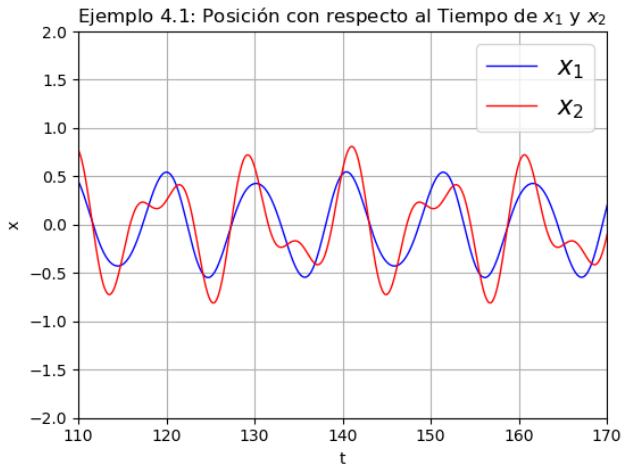


(d) Ciclo limitado para x_2

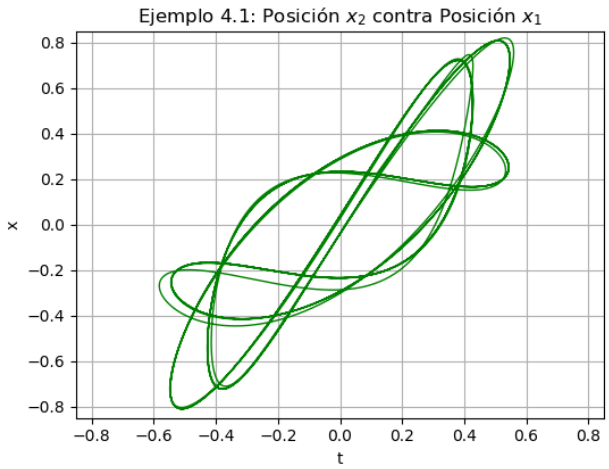


(a) Posición con respecto al tiempo de x_1

(b) Posición con respecto al tiempo de x_2



(a) Posición con respecto al tiempo de x_1 y x_2



(b) Posición x_2 contra x_1

3 Conclusión

Ha sido interesante continuar con el tema, ya que a diferencia de la actividad pasada, esta vez el tener que modificar el código para agregar las propiedades de oscilaciones no lineales y forzadas ha presentado un mayor reto. A pesar de esto, todo fue lograda con éxito, la parte de modificar el código fue lo único complicado, fuera de eso, el resto fue como seguir trabajando con la actividad pasada, lo que estuvo bien, por la poca disponibilidad de tiempo.
Por otra parte, con esta actividad se puede reiterar en la opinión de que Jupyter Lab aporta un ambiente más agradable y cómodo que Jupyter Notebook, por sus múltiples y mejores herramientas.

4 Bibliografía

1. Fay, T., Graham, S. (2003) Coupled Srping Equations. Recuperado el 17 de Marzo del 2018 desde http://math.oregonstate.edu/ gibsonn/Teaching/MTH323-010S15/Supplements/coupled_spring.pdf

5 Apéndice

1. **¿Qué más te llama la atención de la actividad completa? ¿Qué se te hizo menos interesante?**

   Me gustó que hubo más interacción con el código, para adaptarlo para que pudiera graficar correctamente los ejemplos del artículo. Ésta y la actividad pasada han sido muy entretenidas, ya que es interesante ver temas vistos en cursos pasados ser aplicados a la programación.

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?**

   Alguna vez fue mencionado por el profesor en el curso de Mecánica II, pero nunca supe cómo eran o funcionaban, hasta ahora; pero fuera de eso, yo no conocía su existencia.

3. **¿Qué propondrías para mejorar esta actividad? ¿Te ha parecido interesante este reto?**

   La actividad así como tal, estuvo muy bien, ya que no tomó tanto tiempo y fue como continuar con la actividad pasada, pero un nivel más arriba.

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

   Sí, ha sido muy interesante ver estos temas, además de que pienso que son herramientas muy útiles, no solo para el curso, si no para cualquier área de la física.