Actividad 6

Fernando Leyva Cárdenas

April 15, 2018

El programa clásico para el comienzo de las ecuaciones diferenciales está cambiando rápidamente, desde enfatizar las técnicas de solución para una variedad de tipos de ecuaciones diferenciales a los sistemas de énfasis y más aspectos cualitativos de la teoría de las ecuaciones diferenciales ordinarias. En particular, hay un énfasis en las ecuaciones no lineales debido en gran parte a la amplia disponibilidad de algoritmos numéricos de alta potencia y capacidades gráficas casi sin esfuerzo que vienen con el álgebra de la computadora; sistemas como Mathematica y Maple.

En el mundo hay muchos problemas que no se pueden resolver analiticamente con las herramientas matematicas rudimentarias,para eso necesitaremos los metodos numericos, para poder aproximar las soluciones usando la herramienta de computo.

Eso nos lleva alo que consiste esta actividad la cual sera resolver ecuaciones diferenciales, en este caso un sistema de dos resortes; para esto usaremos la plataforma de jupyter lab (usando phyton). Nuestro primer problema son dos resortes con dos masas, con su cierta longitud, coeficientes de friccion, longitud, contante elastica, masas, etc; para nuestro ejemplo base tendremos m1=m2=1, con constantes elasticas de 8 y 40 respectivamente, con longitudes naturales de 0.5 y 1 respectivamente, coeficientes de friccion de 0.8 y 0.5, y las condiciones iniciales seran x1 = 0.5, y1 = 0.0, x2 = 2.25, y2 = 0.0, en donde x1 y x2 son los desplazamientos iniciales y y1, y2 las velocidades. Ahora utilizando el lenguaje de programacion phyton pondremos nuestras ecuaciones diferenciales utilizando **def vectorfield(w, t, p)**, es decir:

```
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
```

Una vez ejecutado este segmento de codigo, se escribira el codigo donde se introduciran los datos del problema, es decir:

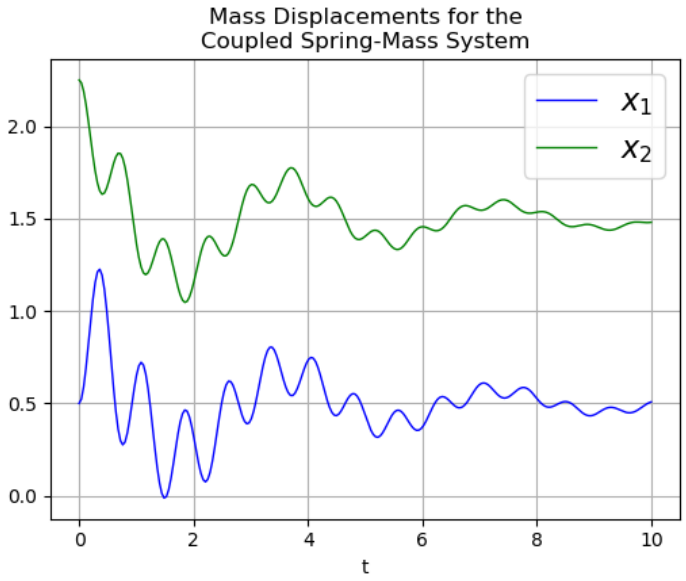
```
# Use ODEINT to solve the differential equations defined by the vector field
from scipy.integrate import odeint

# Parameter values
# Masses:
m1 = 1.0
m2 = 1.5
# 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  (t1, w1[0], w1[1], w1[2], w1[3], file=f)
```

Una vez hecho, entonces graficaremos el resultado con el siguiente codigo:

```
# Plot the solution that was generated

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_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)
```

Y obtenemos una grafica asi:



Este fue nuestro ejemplo base de la seccion del articulo; ahora haremos otros ejemplos diferentes para ver lo que podemos apreciar, iniciaremos con el ejemplo 2.1, en donde tiene ecuaciones:

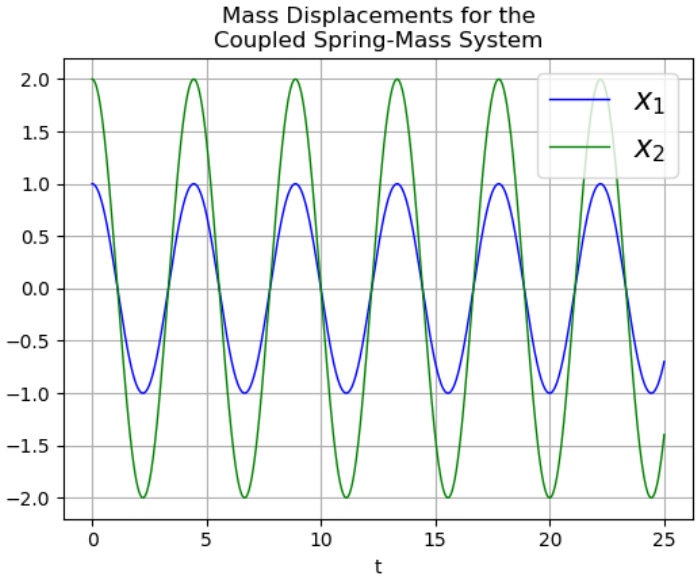
Basicamente se ejecutara el mismo codigo anterior, solamente que con otro nombre, y con condiciones:

```
# Parameter values
# Masses:
m1 = 1.0
m2 = 1.0
# Spring constants
k1 = 6.0
k2 = 4.0
# Natural lengths
```

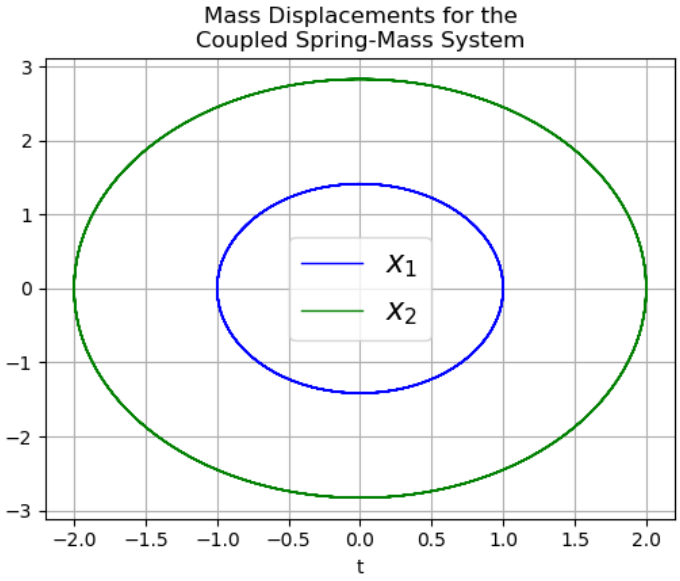
```
L1 = 0.0
L2 = 0.0
# Friction coefficients
b1 = 0.0
b2 = 0.0

# 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
```

En donde obtuvimos las graficas correspondientes:

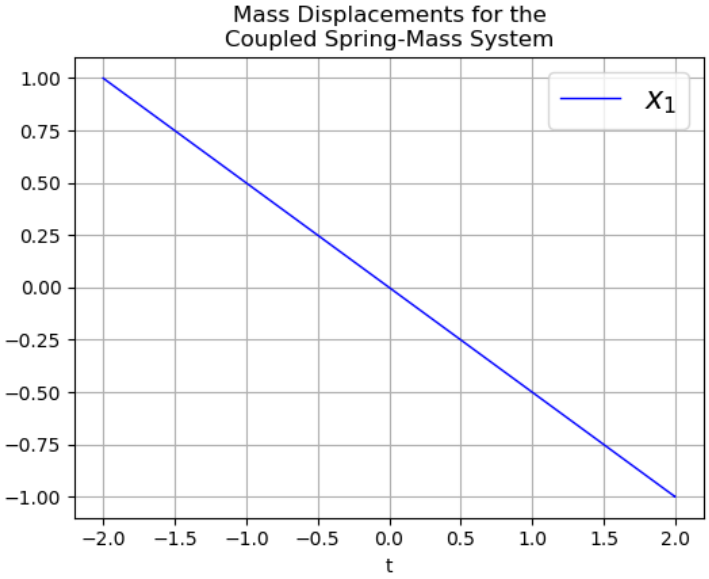
la gracica de x1 y x2 con respecto del tiempo



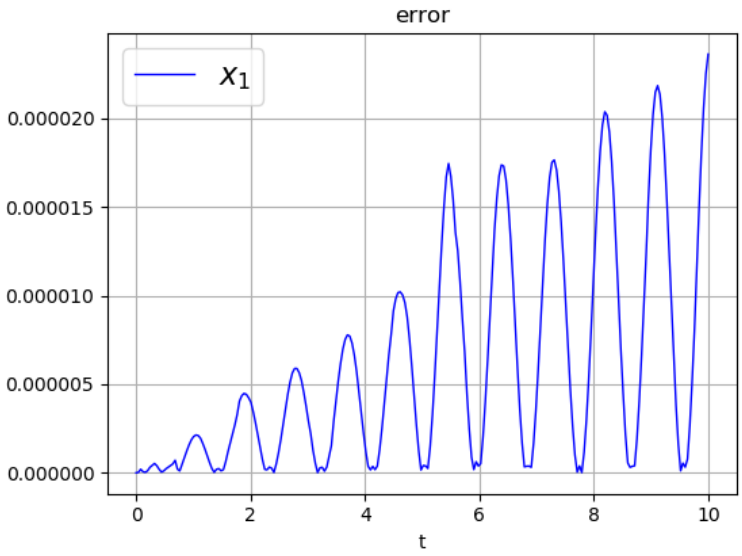
grafica de x1 y x2 con respcto a v1 y v2:



x1 con respecto a x2:



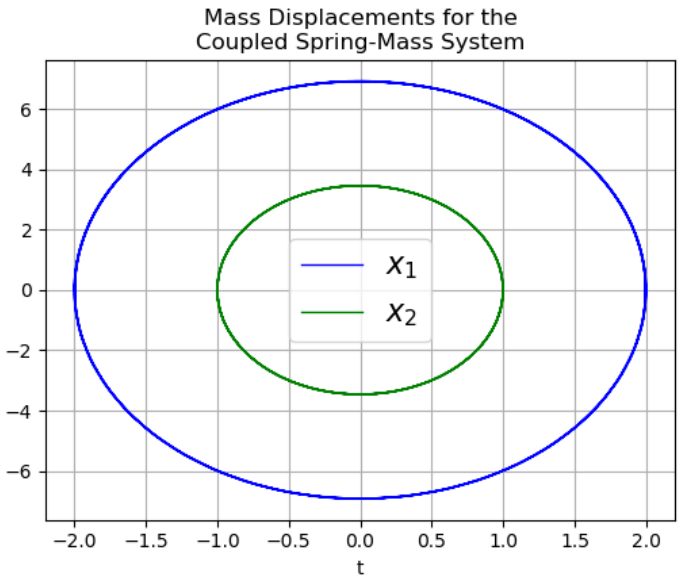
Y la grafica del error relativo:

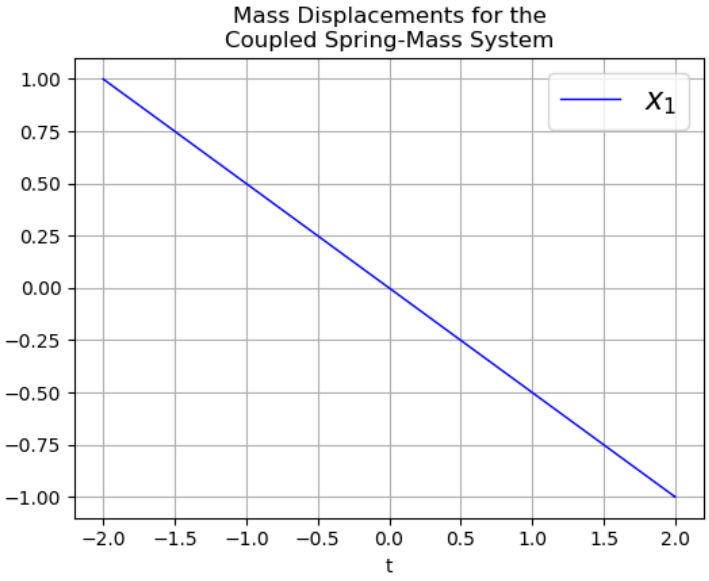


Para el ejemplo 2.2 usaremos la misma ecuacion diferencial, pero ahora con condiciones:

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

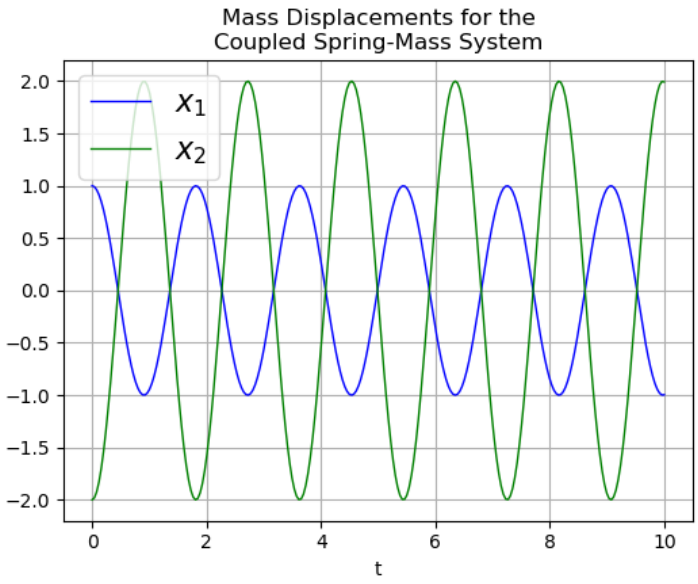
Cambiando el nombre y ejecutando los mismos pasos del ejemplo anterior, tenemos las siguientes graficas:

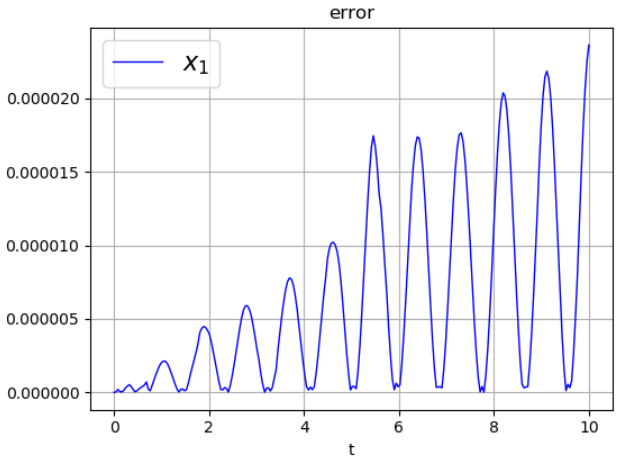




Ahora para el ejemplo 2.3, tenemos las siguientes condiciones iniciales:

```
# 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
```

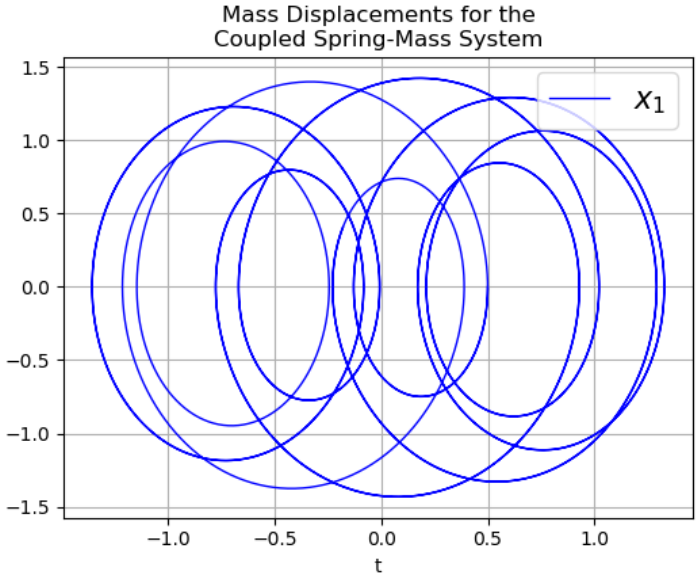


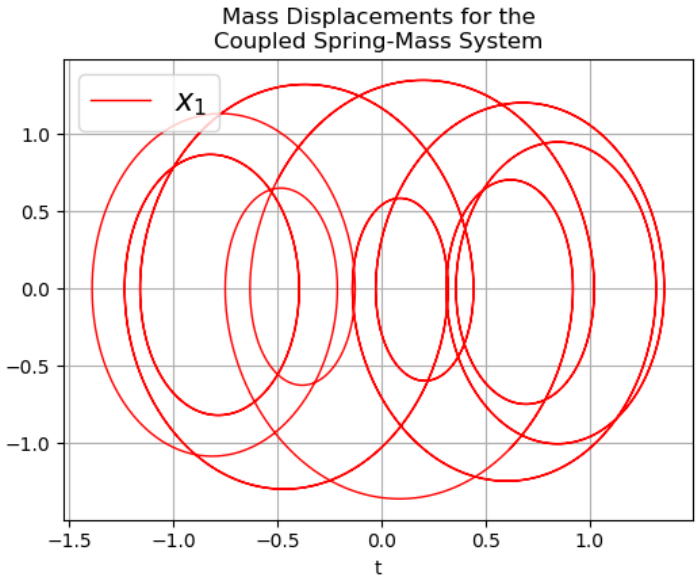


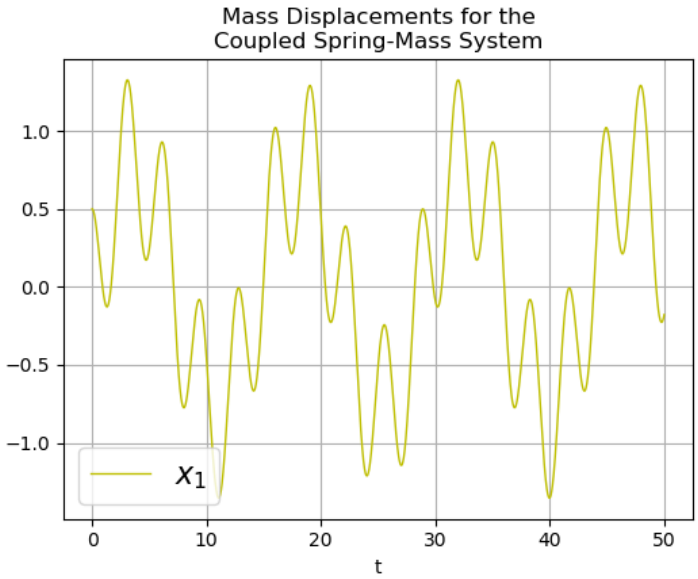

```
# Spring constants
k1 = 0.4
k2 = 1.808
# Natural lengths
L1 = 0.0
L2 = 0.0
# Friction coefficients
b1 = 0.0
b2 = 0.0

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

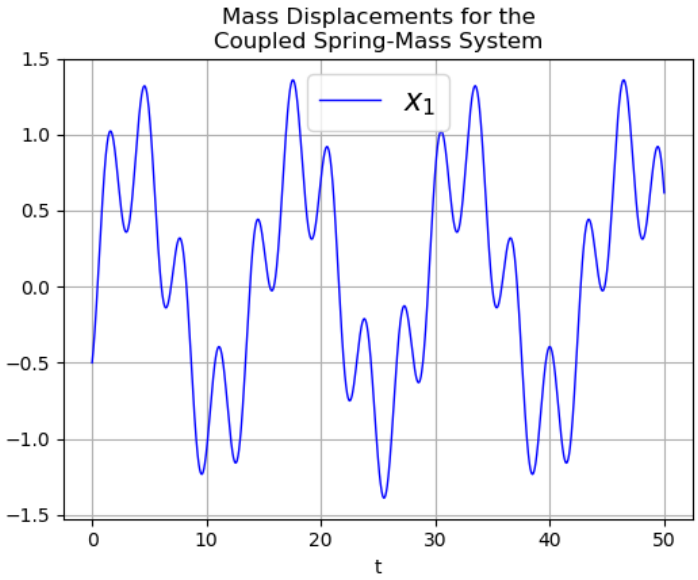
```
y2 = 0.7
```

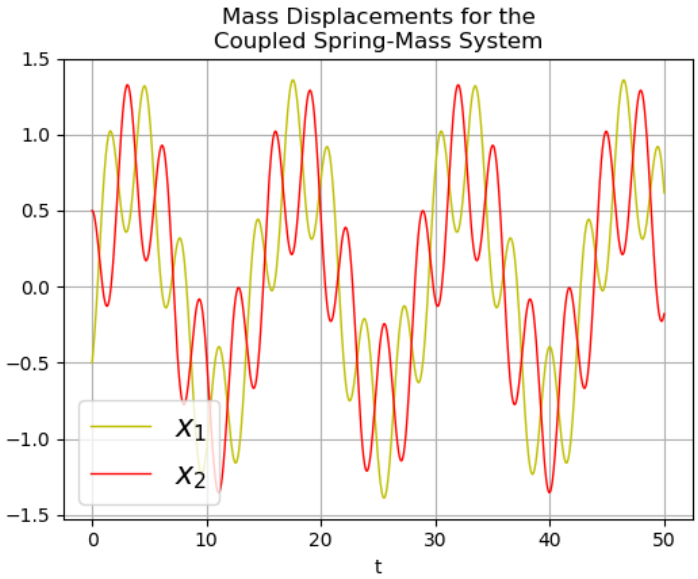
En la cual obtenemos las siguientes graficas: Y a partir de la pagina 12, estan las graficas del ejemplo 2.4, con sus condiciones

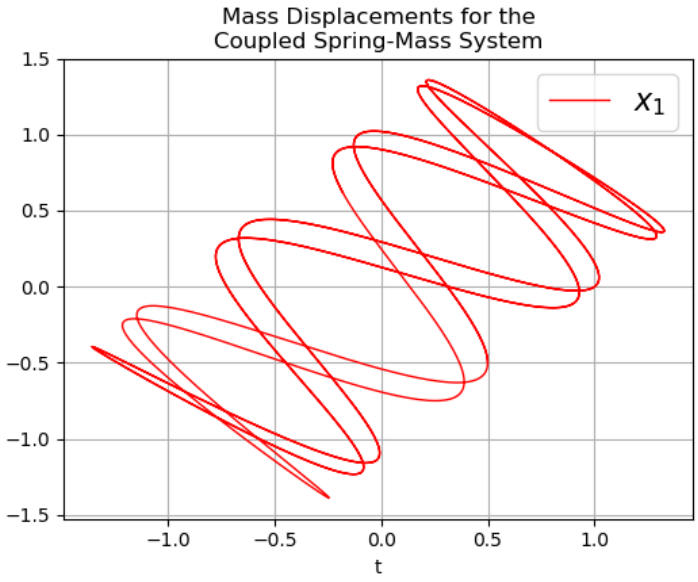


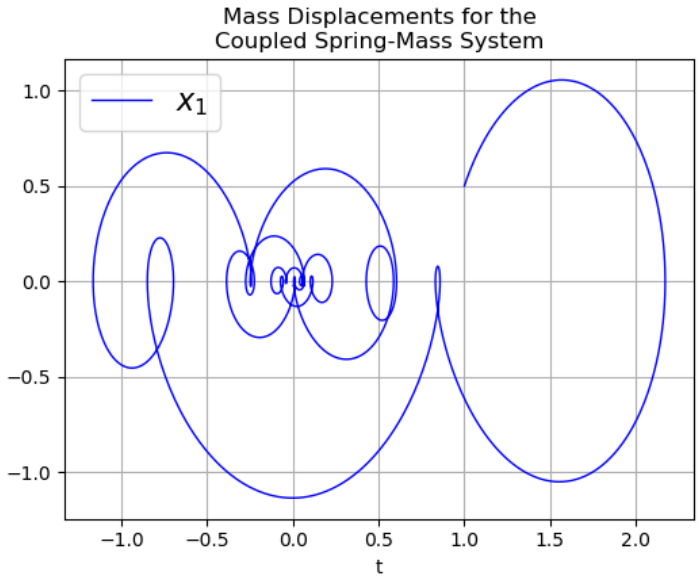


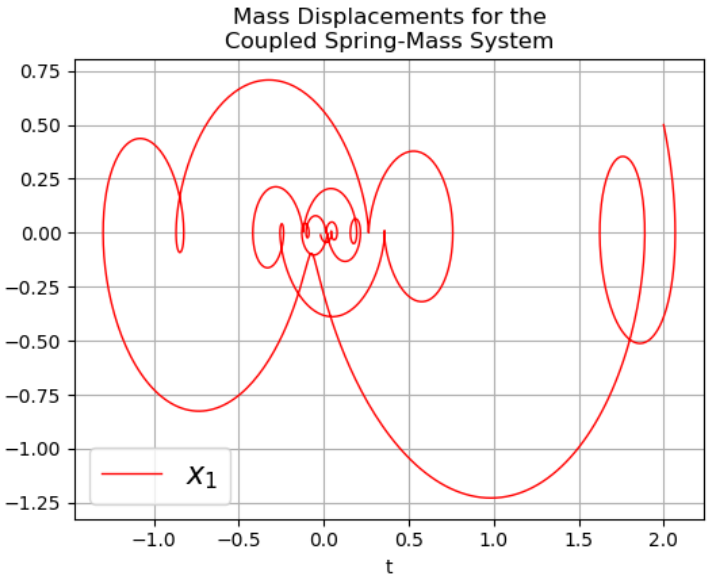

Mass Displacements for the
Coupled Spring-Mass System
1.0
0.5
0.0
−0.5
−1.0
x_1
0
10
20
30
40
50
t

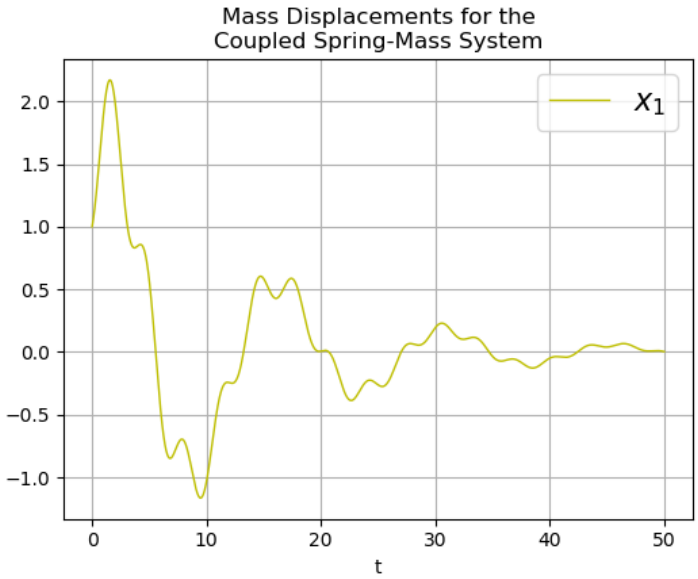

Mass Displacements for the
Coupled Spring-Mass System
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
0
10
20
30
40
50
t

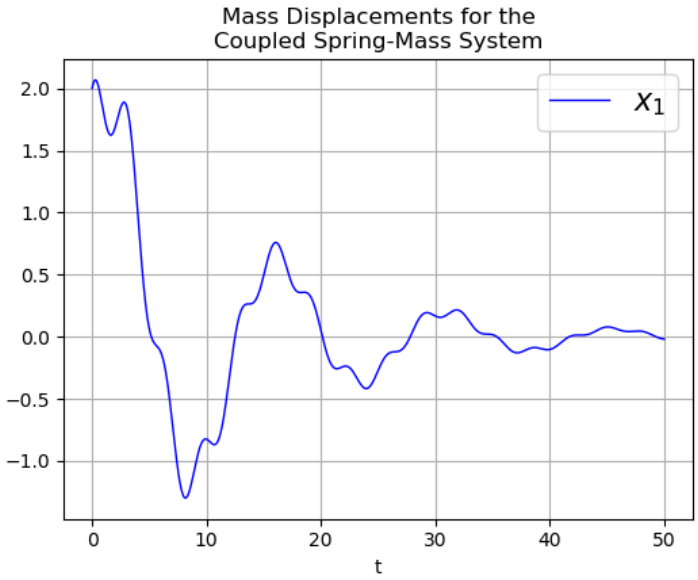

Mass Displacements for the
Coupled Spring-Mass System
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
t
x_1
x_2

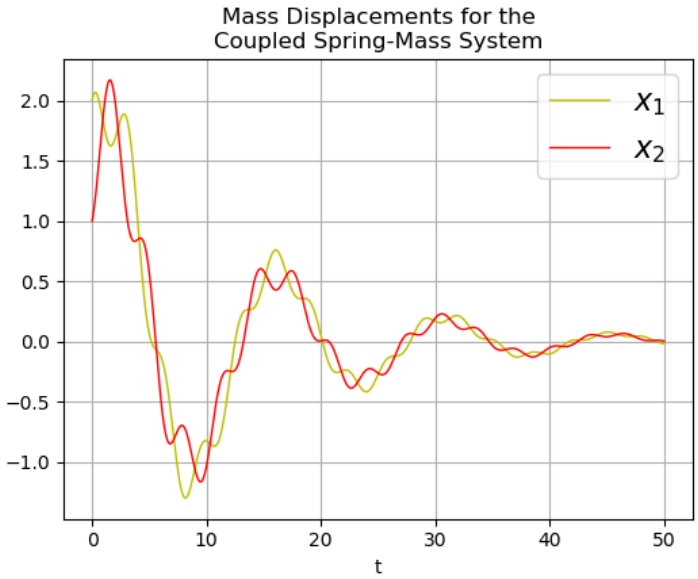

Mass Displacements for the
Coupled Spring-Mass System
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
t
x_1

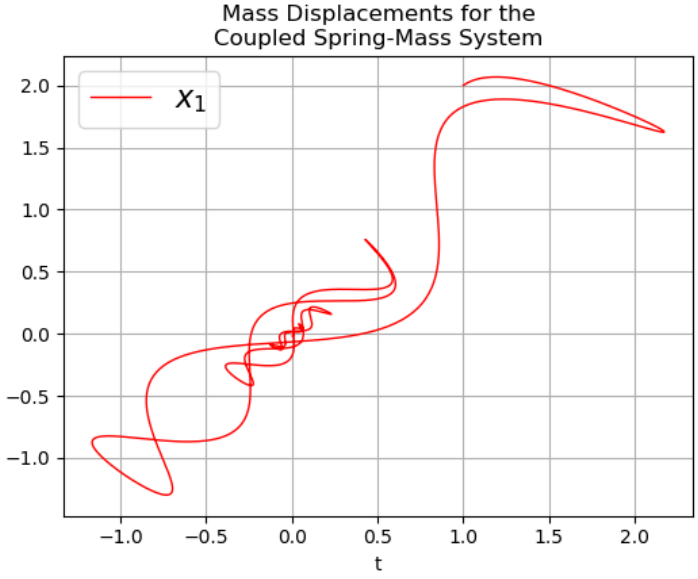

Mass Displacements for the
Coupled Spring-Mass System
x_1
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
t


Mass Displacements for the
Coupled Spring-Mass System
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
x_1
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
t


Mass Displacements for the
Coupled Spring-Mass System
x_1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40
50
t


Mass Displacements for the
Coupled Spring-Mass System
x_1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40
50
t


Mass Displacements for the
Coupled Spring-Mass System
x_1
x_2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
10
20
30
40
50
t


Mass Displacements for the
Coupled Spring-Mass System
x_1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
t