ResortesAcoplados

August 23, 2018

1 Sistema de resortes acoplados, Método de Lagrange, Resolución numérica de sistema de EDOs de 2° orden

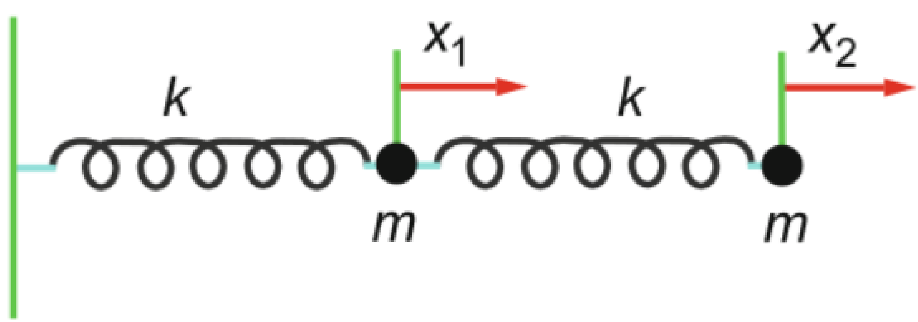
Considere un sistema como el de la imagen, con dos partículas de masa m y dos resortes de constante k

1) Obtenga el Lagrangiano del sistema
2) Desarrolle las ecuaciones de Lagrange que describen el movimiento del sistema
3) Obtenga los valores de eigenfrecuencia del sistema
4) Realice una reducción a un sistema de ecuaciones diferenciales de orden 1
5) Resuelva numéricamente para obtener gráficos de la posición en función del tiempo para dos situaciones extremas

---

1) Tenemos dos grados de libertad y definimos las coordenadas generalizadas $q_1 = x_1$, $q_2 = x_2$

La energía cinética está dada por: $T = \frac{m}{2}\dot{x_1}^2 + \frac{m}{2}\dot{x_2}^2$
La energía potencial: $U = \frac{k}{2}x_1^2 + \frac{k}{2}(x_2 - x_1)^2 = kx_1^2 + \frac{k}{2}x_2^2 - kx_2x_1$
El Lagrangiano: $\mathcal{L} = \frac{m}{2}\dot{x_1}^2 + \frac{m}{2}\dot{x_2}^2 - kx_1^2 - \frac{k}{2}x_2^2 + kx_2x_1$



dosResortes.png

2) Las ecuaciones de Lagrange para este sistema son:

x_1 : $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x_1}}\right) = \frac{\partial \mathcal{L}}{\partial x_1}$ $\qquad m\ddot{x_1} = -2kx_1 + kx_2 \qquad (1)$
x_2 : $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x_2}}\right) = \frac{\partial \mathcal{L}}{\partial x_2}$ $\qquad m\ddot{x_2} = kx_1 - kx_2 \qquad (2)$

Tenemos un sistema de dos ecuaciones diferenciales de segundo orden que describen el movimiento en este sistema

---

3) Sabemos que $x_i(t)$ es un movimiento armónico, podemos modelarlo como $x_i(t) = A_i.cos(\omega t) \Rightarrow$

$x_1(t) = A_1.cos(\omega t) \qquad x_2(t) = A_2 cos(\omega t)$
$\dot{x_1}(t) = -A_1.\omega.cos(\omega t) \qquad \dot{x_2}(t) = -A_2.\omega.cos(\omega t)$
$\ddot{x_1}(t) = -A_1.\omega^2.cos(\omega t) \qquad \ddot{x_2}(t) = -A_2.\omega^2.cos(\omega t)$

Reemplazando en (1) y (2) tenemos:
$-mA_1\omega^2 cos(\omega t) = -2kA_1 cos(\omega t) + kA_2 cos(\omega t)$
$-mA_2\omega^2 cos(\omega t) = kA_1 cos(\omega t) - 2kA_2 cos(\omega t)$

Simplificando y expresando en forma de matriz tenemos:
$\begin{bmatrix} 2k - m\omega^2 & -k \\ -k & k - m\omega^2 \end{bmatrix} . \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

La única forma de obtener una solución no trivial es que el determinante de la matriz sea igual a cero:
$\begin{vmatrix} 2k - m\omega^2 & -k \\ -k & k - m\omega^2 \end{vmatrix} = (2k - m\omega^2).(k - m\omega^2) - (-k)^2 = 0$

Resolviendo tenemos: $\omega^2 = \frac{3 \pm \sqrt{5}}{2}$
Las eigenfrecuencias serán:
$\omega_1 = 1.618\sqrt{k/m} \quad , \quad \omega_2 = 0.618\sqrt{k/m}$
Si reemplazamos estos valores en la expersión matricial obtenemos:
para $\omega_1 \rightarrow A_2 = 1,618A_1 \quad , \quad$ para $\omega_2 \rightarrow A_2 = -0,0618A_1$
El primer caso corresponde al modo simétrico donde ambas amplitudes tienen el mismo signo
El segundo caso, es el modo asimétrico con aplitudes de signo opuesto
Para entender mejor que es lo que sucede en cada caso vamos a graficar el comportamiento del sistema en cada uno de ellos

---

4) Las ecuaciones (1) y (2) forman un sistema de EDOs de segundo orden. Despejando la aceleración para x1 y x2 obtenemos:

$\ddot{x_1} = (-2kx_1 + kx_2)/m \qquad (3)$
$\ddot{x_2} = (kx_1 - kx_2)/m \qquad (4)$

Para resolverlas numéricamente es conveniente transformarlas a un sistema de primer orden. Para esto vamos a realizar un cambio de variables.
Introduciremos las funciones $y_1 = \dot{x_1}, y_2 = \dot{x_2}$, por lo tanto $\dot{y_1} = \ddot{x_1}, \dot{y_2} = \ddot{x_2}$
Entonces re escribimos el sistema de (3) y (4) como sigue:

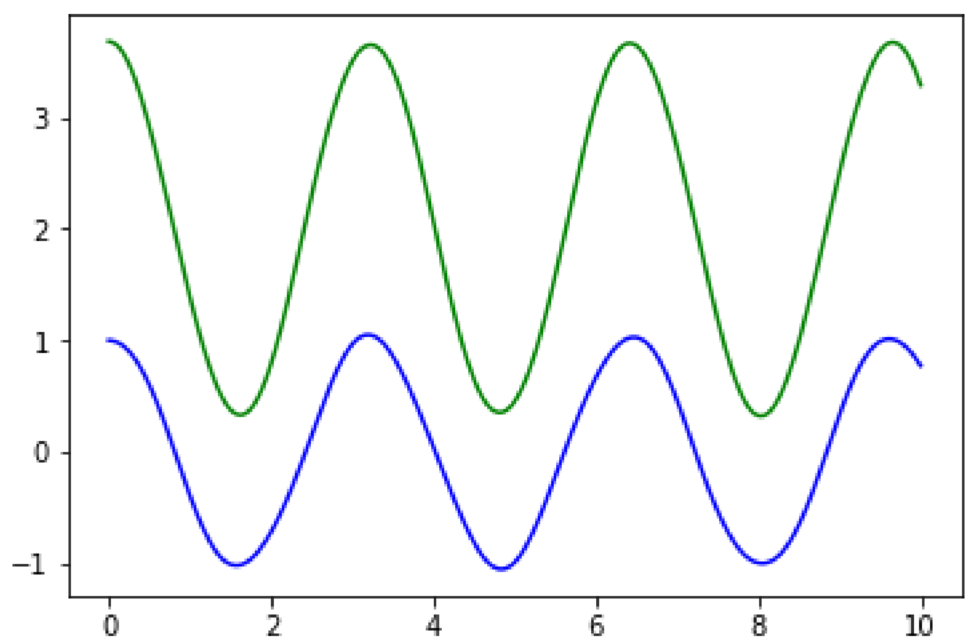
$\dot{x_1} = y_1$
$\dot{y_1} = (-2kx_1 + kx_2)/m$
$\dot{x_2} = y_2$
$\dot{y_2} = (kx_1 - kx_2)/m$
Ahora tenemos un sistema equivalente de 4 ecuaciones diferenciales de primer orden

---

5) Ahora tenemos todo lo necesario para resolver numéricamente el sistema. A continuación se muestra un ejemplo de resolución en Python 2.7

In [1]:
```
#Armo una función con el sistema de ecuaciones
def vectores (variables, tiempo, parametros):
    #toma los valores depara las variables  y los parámetros
    x1, y1, x2, y2 = variables
    m, k = parametros
    #creo las funcione x1' y1' x2' y2'
    f=[y1, (-2*k*x1+k*x2)/m, y2, (-k*x2+k*x1)/m]
    #devuelve una lista de valores x1' y1' x2' y2'
    return f

#doy los valores de los parámetros
m=1
k=10

#doy los valores de posicio y velocidad iniciales
x1=1
y1=0
x2=1.681
y2=0

# Estos son parámetros para el solver de odes
abserr = 1.0e-8
relerr = 1.0e-6
stoptime = 10.0
numpoints = 250

#defino el intervalo de tiempo
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

#empaqueto los parámetros y variables
parametros = [m, k]
variables = [x1, y1, x2, y2]

#importo la librería para resolver edos
from scipy.integrate import odeint
#ejecuto la resolución y guardo los valores con el nombre "solver"
solver = odeint(vectores, variables, t, args=(parametros,), atol=abserr, rtol=relerr)
```

```
#los valores de x1 y x2 son:
x1=solver[:,0]
x2=solver[:,2]
#Recordemos que son dos sitemas de referencia independientes

import numpy as np
#Vamos a estableces una separación entre los puntos de equilibrio de los resortes
ldist=np.full((250),2)
#Ahora definimos para un único sistema de referencia la posición
#para la masa 1 y de la masa 2, xm1 y xm2 respectivamente
xm1=x1
xm2=x2+ldist

#magic command para jupyter
%matplotlib inline

import matplotlib.pyplot as plt
plt.plot(t,xm1, 'b')
plt.plot(t,xm2, 'g')
plt.show()
```

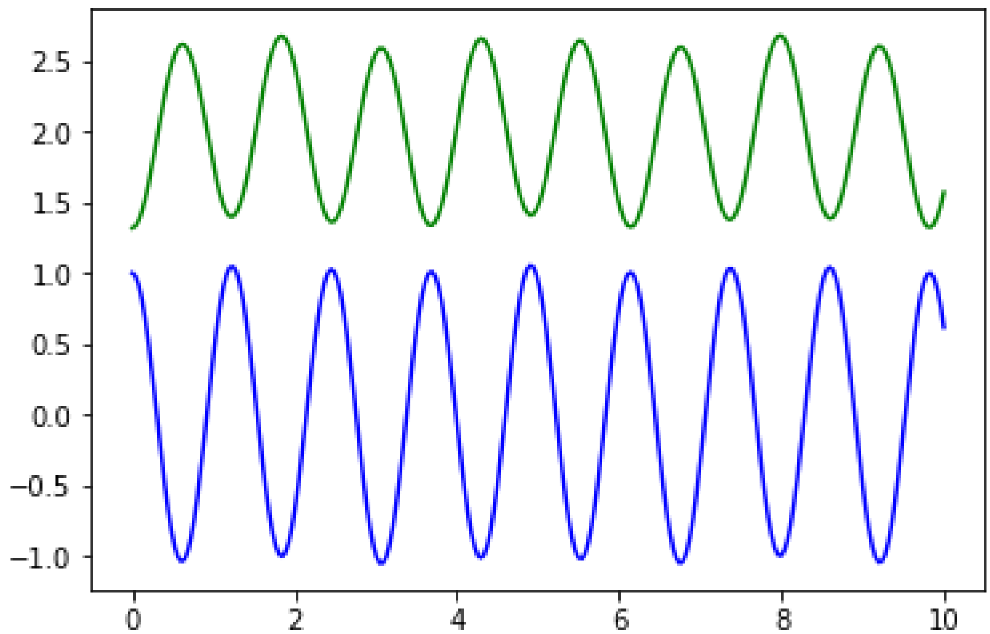


En el ejmplo anterior se dieron valores arbitrarios de $m = 1, k = 10$.

Sin embargo observe que $x1 = 1.68x2$, esto se eligió deliberadamente para mostrar un caso de modo simétrico.

Veamos ahora que pasa si $x1 = -0.68x2$

In [2]:
```
x1=1
x2=-0.681
variables = [x1, y1, x2, y2]
solverB = odeint(vectores, variables, t, args=(parametros,), atol=abserr, rtol=relerr)

x1B=solverB[:,0]
x2B=solverB[:,2]

xm1B=x1B
xm2B=x2B+ldist

plt.plot(t,xm1B, 'b')
plt.plot(t,xm2B, 'g')
plt.show()
```



En base a los resultados obtenidos se han realizado animaciones 3D

Puede ver los videos de las animaciones en:

https://mecanicaracionalfcai.github.io/practicas_archivos/modoSim.mp4

https://mecanicaracionalfcai.github.io/practicas_archivos/modoAsim.mp4

Puede correr la simulación en su computadora y visualizar que sucede cuando cambia distintos parámetros, para ello puede descargar el código completo de este notebook para Jupyter en :

https://mecanicaracionalfcai.github.io/practicas_archivos/ResortesAcoplados.zip

Para correr esta simulación necesita tener instalado Anaconda para Python 2.7 y el módulo vpython.