Parcial 2

S. Moya, S. J. Niño, N. L. Bucurú, A. F. Moreno

Herramientas Computacionales
Departamento de Física
Universidad Nacional de Colombia

22 de mayo de 2020

1. Problema (Valor 2)

Partiendo de las ecuaciones de Euler-Lagrange, figura (1), deducir las ecuaciones de movimiento para el péndulo-doble.

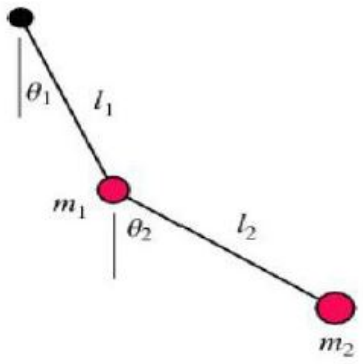


Figura 1: *Diagrama del péndulo-doble.*

Utilizar el código péndulo, dado en clase, e implementar su solución computacional, escribiendo un programa y usando el algoritmo verlet, rk4 y ode45 desarrollados en clase. Anexar las gráfica de la evolución de $\theta_1(t)$ en función del tiempo t, de la misma forma que para $\theta_2(t)$. Presentar los resultados para la parte lineal ($\sin(\theta) \approx \theta$) y no lineal. Por otra parte, graficar θ_1 contra θ_2

Solución:

Para la solución del primer punto, se uso el lenguaje de programación *JavaScript*, el cual permitió hacer diferentes simulaciones.

Al ser un nuevo lenguaje de programación, se decidió hacer un programa que implemente el método de Runnge-Kutta para resolver las ecuaciones diferenciales de movimiento en el doble péndulo. Se realizaron las gráficas de θ_1 vs t, θ_2 vs t y θ_2 vs θ_1 en tiempo real, es decir, mientras se lleva a cabo una simulación del sistema físico, estas se van graficando.

2. Problema (Valor 1.2)

Utilizar el código de la animación del péndulo simple, visto en clase, y extenderlo al péndulo doble, (punto 1 del parcial) para hacer su animación.

Solución:

Al igual que en el punto anterior, se uso el lenguaje de programación de *JavaScript* para realizar la simulación del doble péndulo, tanto amortiguado, como sin amortiguar.

3. Problema (Valor 0.8)

Utilizar el código de la animación del péndulo doble, (desarrollado en el punto 2 del parcial), e introducir el efecto de amortiguamiento en ambos péndulos de tal manera que la animación del doble péndulo muestre que el sistema llega al punto de equilibrio.

Solución:

Teniendo en cuenta lo escrito en los puntos anteriores, se encontró la dificultad de reescribir los códigos en LaTeX, puesto que, se manejan diferentes carpetas especializadas en diferentes partes del código. Por ejemplo, hay una carpeta especializada en los códigos de *JavaScript*, otra especializada en el código de *CSS* y la ultima que usa *HTML* para configurar la página web. Todas estas carpetas se entrelazan entre sí y puede ser tedioso seguir su orden a través de un archivo de texto en LaTeX.

Es así que, por las razones anteriores, como grupo, se decidió realizar un vídeo, en el cual se explican los códigos y su funcionamiento, el cómo abrir el servidor y el resultado final, además de adjuntar a este documento, los códigos para que el profesor pueda correr el programa sin ningún problema.

Para acceder al vídeo, haga clic **aquí.**

La nomenclatura que se uso, fue:

1. NN = No-Friction No-approximation.
2. NA = No-Friction Approximation.
3. FN = Friction No-approximation.
4. FA = Friction Approximation.
5. 1 = Pendulum 1 (arriba).
6. 2 = Pendulum 2 (abajo).

4. Problema (Valor 1.0)

En el péndulo de Wilberforce se muestra las oscilaciones acopladas de los modos longitudinales y torsionales de un cuerpo de forma cilíndrica que cuelga de un muelle en forma de hélice. El acoplamiento entre los dos modos de oscilación está descrito por una función lineal de la forma $\epsilon\ x\theta/2$, donde ϵ se denomina constante de acoplamiento.

La energía del sistema es la suma de la energía cinética de traslación del bloque, de rotación alrededor del eje vertical, la energía potencial del muelle cuando se deforma una longitud x, y cuando gira un ángulo θ, y la energía de acoplamiento

$$E = \frac{1}{2}m\left(\frac{dx}{dt}\right)^2 + \frac{1}{2}I\left(\frac{d\theta}{dt}\right)^2 + \frac{1}{2}k_x x^2 + \frac{1}{2}k_\theta \theta^2 + \frac{1}{2}\epsilon x\theta \tag{1}$$

El lagrangiano L = T - V, es:

Figura 2: *Diagrama del péndulo de Wilberforce.*

$$L = \frac{1}{2}m\left(\frac{dx}{dt}\right)^2 + \frac{1}{2}I\left(\frac{d\theta}{dt}\right)^2 - \frac{1}{2}k_x x^2 - \frac{1}{2}k_\theta \theta^2 - \frac{1}{2}\epsilon x\theta$$

Sus ecuaciones de movimiento son:

$$\frac{d^2x}{dt^2} + \frac{k_x}{m}x + \frac{\epsilon}{2m}\theta = 0 \tag{2}$$

$$\frac{d^2\theta}{dt^2} + \frac{k_\theta}{I}\theta + \frac{\epsilon}{2I}x = 0 \tag{3}$$

Eliminando x en el sistema de dos ecuaciones diferenciales anterior, se obtiene:

$$\frac{d^4\theta}{dt^4} + (\omega_\theta^2 + \omega_x^2)\left(\frac{d^2\theta}{dt^2}\right) + \left(\omega_\theta^2\omega_x^2 - \frac{\epsilon^2}{4mI}\right)\theta = 0 \tag{4}$$

Donde las frecuencias angulares son:

$$\omega_x^2 = \frac{k_x}{m}; \quad \omega_\theta^2 = \frac{k_\theta}{I} \tag{5}$$

Implementar su solución computacional, escribiendo un programa y usando el algoritmo rk4 y ode45 desarrollados en clase. Anexar las gráfica de la evolución de x(t) en función del tiempo t, de la misma forma que para $\theta(t)$. Demostrar como se obtiene la última ecuación diferencial de 4 orden. Por otra parte, graficar x contra θ. Graficar las energías correspondientes a las energías potencial y cinética en x y las energías potencial y cinética en θ y en el modo acoplado $\epsilon x\theta/2$, en el tiempo. Graficar la energía mecánica en el tiempo.

Solución: Para resolver este problema utilizamos el lenguaje de programación Python. Anterior al desarrollo computacional demostraremos la forma de la ecuación 4; despejando x en la ecuación 3 y utilizando la notación de Newton para las derivadas temporales tenemos que:

$$\ddot{\theta} + \frac{k_\theta}{I}\theta = -\frac{\epsilon}{2I}x$$

$$-\frac{2I}{\epsilon}\ddot{\theta} - \frac{2k_\theta}{\epsilon} = x$$

Reemplazando el lado izquierdo de la anterior expresión en 2, tenemos que:

$$\frac{d^2}{dt^2}\left(-\frac{2I}{\epsilon}\ddot{\theta}-\frac{2k_\theta}{\epsilon}\right)+\frac{k_x}{m}\left(-\frac{2I}{\epsilon}\ddot{\theta}-\frac{2k_\theta}{\epsilon}\right)+\frac{\epsilon}{2m}\theta=0$$
$$-\frac{2I}{\epsilon}\dddot{\theta}-\frac{2k_\theta}{\epsilon}\ddot{\theta}+\frac{k_x}{m}\left(-\frac{2I}{\epsilon}\ddot{\theta}-\frac{2k_\theta}{\epsilon}\right)+\frac{\epsilon}{2m}\theta=0$$

Multiplicando por $-\frac{\epsilon}{2I}$ y de 5:

$$\ddddot{\theta}+\omega_\theta^2\ddot{\theta}+\omega_x^2\ddot{\theta}+\omega_x^2\omega_\theta^2\theta-\frac{\epsilon^2}{4mI}=0$$
$$\ddddot{\theta}+(\omega_\theta^2+\omega_x^2)\ddot{\theta}+(\omega_x^2\omega_\theta^2-\frac{\epsilon^2}{4mI})\theta=0$$

Con lo que queda demostrada la forma de la ecuación 4.

Para resolver el problema 4 primero se debe plantear el sistema de ecuaciones lineales, que queda de la forma:

$$\frac{dx}{dt}=v$$
$$\frac{dv}{dt}=-\frac{k_x}{m}x-\frac{\epsilon}{2m}\theta$$
$$\frac{d\theta}{dt}=\omega$$
$$\frac{d\omega}{dt}=-\frac{k_\theta}{I}\theta-\frac{\epsilon}{2I}x$$

4.1. Ode45

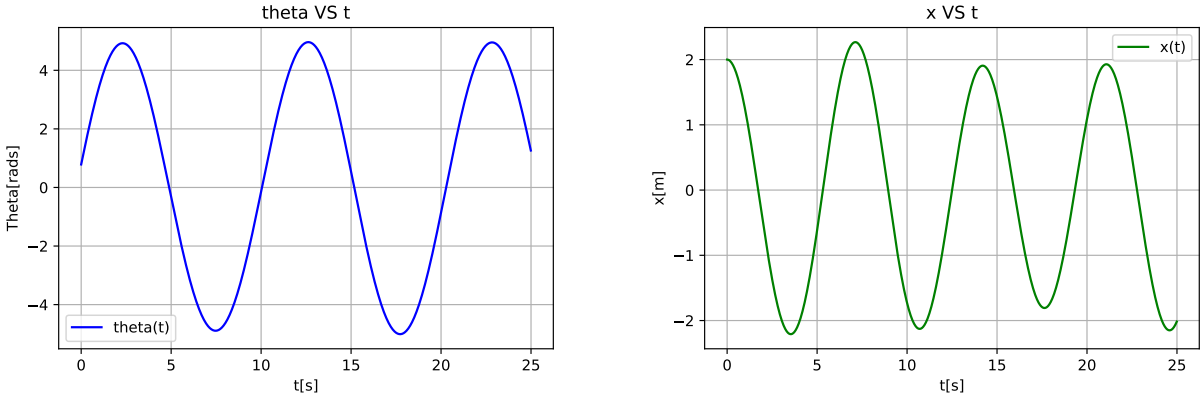
```
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np
import math as mt
#Definimos los par metros
ktheta = 1.5
kx = 2
m = 2.5
epsilon = 0.1
I = 4


def dx(x, v, theta, omega): #Ecuacion diferencial de primer orden para dx/dt=v
  return v


def dtheta(x, v, theta, omega): #Ecuacion diferencial de primer orden para dtheta/dt=w
  return omega

#Ecuacion diferencial de primer orden para dv/dt=-kx*x/m - epsilon*theta/(2*m)
```

```
def dv(x, v, theta, omega):
  return -kx*x/m - epsilon*theta/(2*m)

#Ecuacion diferencial de primer orden para dw/dt=-ktheta*theta/I - epsilon*x/(2*I)\\
def dw(x, v, theta, omega):
  return -ktheta*theta/I - epsilon*x/(2*I)

#Definicion de Energia en x
def Ex(x, theta, v, omega):
  return m/2*(v**2) + 1/2*kx*(x**2)

#Definicion de Energia en \theta
def Etheta(x, theta, v, omega):
  return 1/2*I*(omega**2) + 1/2*ktheta*(theta**2)

#Definicion energia total del sistema
def Etotal(x, theta, v, omega):
  return (m/2*(v**2) + 1/2*kx*(x**2)
          + 1/2*I*(omega**2)
          + 1/2*ktheta*(theta**2)
          + 1/2*epsilon*x*theta )

#Definimos el sistema
def system(state, t):
  x, theta, v, omega, = state #Definiendo el vector del sistema ,
  return (dx(x, v, theta, omega),
          dtheta(x, v, theta, omega),
          dv(x, v, theta, omega),
          dw(x, v, theta, omega))
#nos devuelve las funciones que establecimos anteriormente

#difinimos el tiempo para correr las graficas y su particion
t = np.arange(0, 25, 0.01)


#condiciones iniciales
state_0 = [2, mt.pi/4, 0, 3]

#Usando el integrador Python odeint, que tiene la funcion del Ode45 en
#Octave, resolvermos el sistema con la funcion del sistema y condiciones
#iniciales
states = odeint(system, state_0, t)

#Energia en x
energyx = Ex(states[:,0], states[:,1], states[:,2], states[:,3])

#Energia en \Theta
energytheta = Etheta(states[:,0], states[:,1], states[:,2], states[:,3])
```

```
#Energia total
energytotal = Etotal(states[:,0], states[:,1], states[:,2], states[:,3])

#grafica de x vs t
plt.plot(t, states[:, 0], "g", label="x(t)")
plt.title("x␣VS␣t")
plt.xlabel("t[s]")
plt.ylabel("x[m]")
plt.grid()
plt.legend()
plt.savefig("x_t.svg")

#grafica \theta  vs t
plt.clf()
plt.plot(t, states[:, 1], "b", label="theta(t)")
plt.title("theta␣VS␣t")
plt.xlabel("t[s]")
plt.ylabel("Theta[rads]")
plt.grid()
plt.legend()
plt.savefig("theta_t.svg")

#grafica de x vs \Theta
plt.clf()
plt.plot(states[:, 1], states[:, 0], "r", label="x(theta)")
plt.title("x␣VS␣theta")
plt.xlabel("Theta[rads]")
plt.ylabel("x[m]")
plt.grid()
plt.legend()
plt.savefig("theta_x.svg")

#Grafica de energia en x en el tiempo
plt.clf()
plt.plot(t, energyx, "g", label="Ex(t)")
plt.title("Energia␣longitudinal␣VS␣tiempo")
plt.xlabel("Tiempo[s]")
plt.ylabel("Energia[Joules]")
plt.grid()
plt.legend()
plt.savefig("Ex.svg")


#Grafica de energia en \theta en el tiempo
plt.clf()
plt.plot(t, energytheta, "b", label="Etheta(t)")
plt.title("Energia␣de␣torsi n␣VS␣tiempo")
```

```
plt.xlabel("Tiempo[s]")
plt.ylabel("Energia[Joules]")
plt.grid()
plt.legend()
plt.savefig("Etheta.svg")

#Grafica de energia total en el tiempo
plt.clf()
plt.plot(t, energytotal, "r", label="Etotal(t)")
plt.title("Energia total VS tiempo")
plt.xlabel("Tiempo[s]")
plt.ylabel("Energia[Joules]")
plt.grid()
plt.legend()
plt.savefig("Etotal.svg")
```

.



(a) *θ en función de t*

(b) *x en función de t*

Figura 3: *Movimiento de las coordenadas generalizadas del péndulo de Wilberforce en el tiempo*

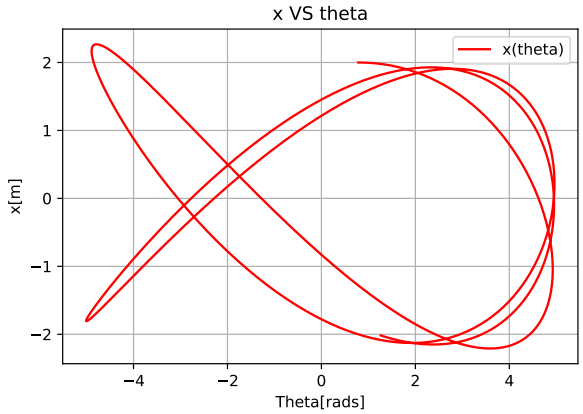
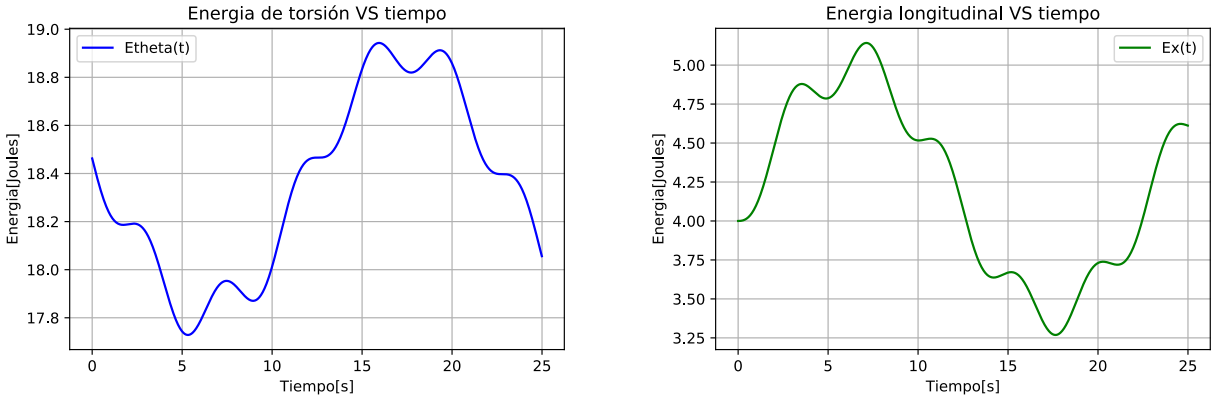


Figura 4: *Movimiento de x en función de θ*



(a) *Energía enθ en función de t*

(b) *Energia en x en función de t*

Figura 5: *Energía de las coordenadas generalizadas del péndulo de Wilberforce en el tiempo*

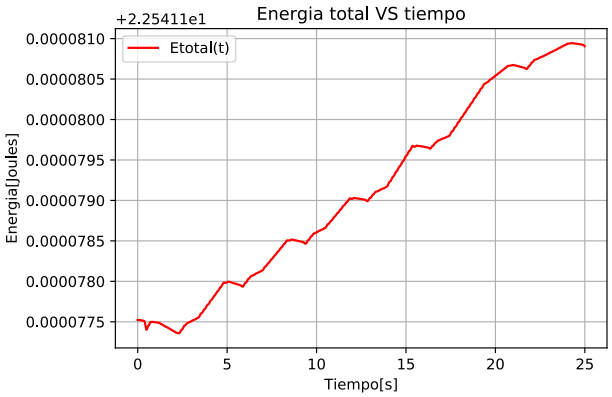


Figura 6: *Energía total en función del tiempo en el péndulo de Wilberforce*

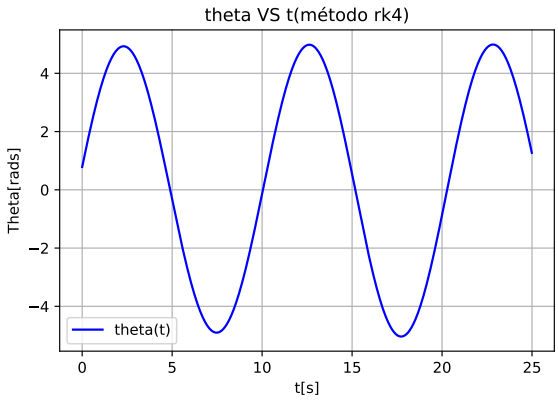
4.2. rk4

```
import matplotlib.pyplot as plt
import numpy as np
import math as mt

#Se declaran las constantes especificas del problema

ktheta = 1.5                    #Constante elastica de torsi n
kx = 2                          #Constante elastica longitudinal
m = 2.5                         #Masa del cuerpo
epsilon = 0.1                   #Constante de acoplamiento
I = 4                           #Momento de Inercia

h=0.01                          #Tama o del intervalo
t = np.arange(0, 25, h)         #Vector de tiempo de 0-25 en pasos de h
n=len(t)                        #Tama o vector de tiempo

#Se inicializan las matrices para cada variable con sus condiciones iniciales

x=np.zeros(n); x[0]= 2
theta=np.zeros(n); theta[0]= mt.pi/4
v=np.zeros(n); v[0]= 0
omega=np.zeros(n); omega[0]= 3

#Se declaran las funciones del sistema de ecuaciones lineales a resolver

def dx(v):
  return v

def dtheta(omega):
  return omega
```

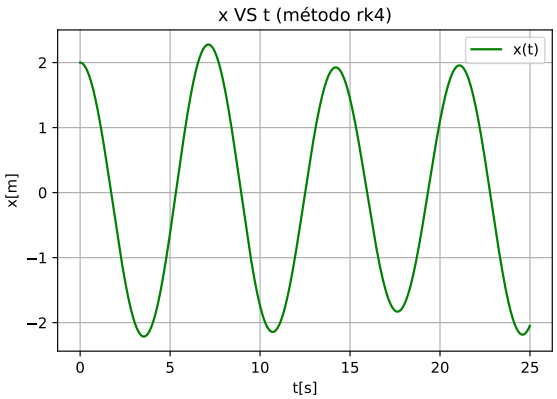
```
def dv(x,theta):
  return -kx*x/m - epsilon*theta/(2*m)

def dw(x,theta):
  return -ktheta*theta/I - epsilon*x/(2*I)

#Se declaran las funciones de Energia longitudinal, de torsi n y total

def Ex(x, theta, v, omega):
  return m/2*(v**2) + 1/2*kx*(x**2)


def Etheta(x, theta, v, omega):
  return 1/2*I*(omega**2) + 1/2*ktheta*(theta**2)


def Etotal(x, theta, v, omega):
  return (m/2*(v**2) + 1/2*kx*(x**2) + 1/2*I*(omega**2)+
          1/2*ktheta*(theta**2) + 1/2*epsilon*x*theta)

#Metodo de Runge-Kutta de cuarto orden para un sistema de 4 ecuaciones

for i in range(0,n-1):
    k1 = h*dx(v[i])
    l1 = h*dv(x[i], theta[i])
    n1 = h*dtheta(omega[i])
    m1 = h*dw(x[i], theta[i])
    k2 = h*(dx(v[i]) +1/2*l1)
    l2 = h*dv(x[i] + 1/2*k1, theta[i] +1/2*n1)
    n2 = h*(dtheta(omega[i]) + 1/2*m1)
    m2 = h*dw(x[i] + 1/2*k1, theta[i] + 1/2*n1)
    k3 = h*(dx(v[i]) +1/2*l2)
    l3 = h*dv(x[i] + 1/2*k2, theta[i] +1/2*n2)
    n3 = h*(dtheta(omega[i]) + 1/2*m2)
    m3 = h*dw(x[i] + 1/2*k2, theta[i] + 1/2*n2)
    k4 = h*(dx(v[i]) +1/2*l3)
    l4 = h*dv(x[i] + 1/2*k3, theta[i] +1/2*n3)
    n4 = h*(dtheta(omega[i]) + 1/2*m3)
    m4 = h*dw(x[i] + 1/2*k3, theta[i] + 1/2*n3)
    x[i+1] = x[i] + (1/6)*(k1 + 2*k2 + 2*k3 + k4)
    v[i+1] = v[i] + (1/6)*(l1 + 2*l2 + 2*l3 + l4)
    theta[i+1] = theta[i] + (1/6)*(n1 + 2*n2 + 2*n3 + n4)
    omega[i+1] = omega[i] + (1/6)*(m1 + 2*m2 + 2*m3 + m4)

#Creaci n de matrices de energia

energyx = Ex(x, theta, v, omega)
```

```
energytheta = Etheta(x, theta, v, omega)


energytotal = Etotal(x, theta, v, omega)

#Graficas de varias relaciones

plt.plot(t, x, "g", label="x(t)")
plt.title("x VS t")
plt.xlabel("t[s]")
plt.ylabel("x[m]")
plt.grid()
plt.legend()
plt.savefig("Rk4x_t.svg")


plt.clf()
plt.plot(t, theta, "b", label="theta(t)")
plt.title("theta VS t")
plt.xlabel("t[s]")
plt.ylabel("Theta[rads]")
plt.grid()
plt.legend()
plt.savefig("Rk4theta_t.svg")


plt.clf()
plt.plot(theta, x, "r", label="x(theta)")
plt.title("x VS theta")
plt.xlabel("Theta[rads]")
plt.ylabel("x[m]")
plt.grid()
plt.legend()
plt.savefig("Rk4theta_x.svg")


plt.clf()
plt.plot(t, energyx, "g", label="Ex(t)")
plt.title("Energia longitudinal VS tiempo")
plt.xlabel("Tiempo[s]")
plt.ylabel("Energia[Joules]")
plt.grid()
plt.legend()
plt.savefig("Rk4Ex.svg")
```

```
plt.clf()
plt.plot(t, energytheta, "b", label="Etheta(t)")
plt.title("Energia de torsi n VS tiempo")
plt.xlabel("Tiempo[s]")
plt.ylabel("Energia[Joules]")
plt.grid()
plt.legend()
plt.savefig("Rk4Etheta.svg")


plt.clf()
plt.plot(t, energytotal, "r", label="Etotal(t)")
plt.title("Energia total VS tiempo")
plt.xlabel("Tiempo[s]")
plt.ylabel("Energia[Joules]")
plt.grid()
plt.legend()
plt.savefig("Rk4Etotal.svg")
```



(a) *θ en función de t*



(b) *x en función de t*

Figura 7: *Movimiento de las coordenadas generalizadas del péndulo de Wilberforce en el tiempo- Utilizando el metodo rk4.*

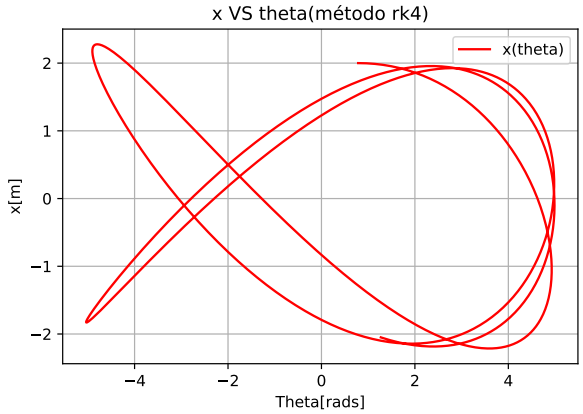
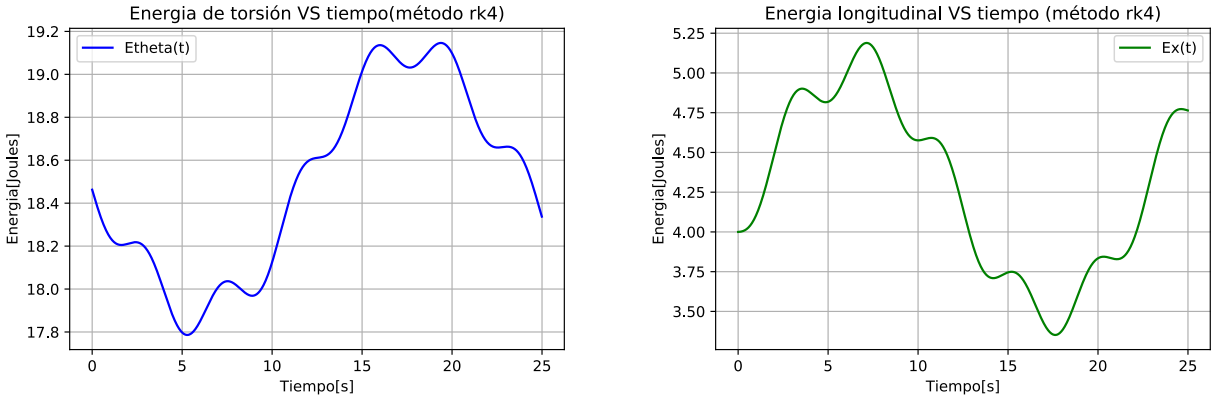


Figura 8: *Movimiento de x en función de θ- Utilizando el metodo rk4.*



(a) *Energía enθ en función de t*

(b) *Energia en x en función de t*

Figura 9: *Energía de las coordenadas generalizadas del péndulo de Wilberforce en el tiempo- Utilizando el metodo rk4.*

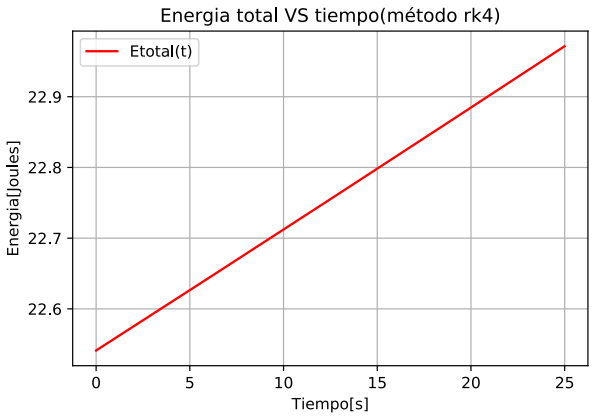


Figura 10: *Energía total en función del tiempo en el péndulo de Wilberforce - Utilizando el metodo rk4.*