Física Numérica

Julio César Avila Torreblanca

15 de diciembre del 2021

Tarea 8

Lanzamiento de martillo.

El record mundial de martillo es de 86.74 m por Yuri Sedykh y se ha mantenido desde 1986. El martillo pesa 7.26 kg, es esférico, y tiene un radio de $R = 6$ cm. La fricción en el martillo puede ser considerada proporcional al cuadrado de la velocidad del martillo relativa al aire:

$$F_D = \frac{1}{2}\rho A C_D v^2 \tag{1}$$

donde ρ es la densidad del aire ($1.2\,\mathrm{kg/m^3}$) y $A = \pi R^2$ es la sección transversal del martillo. El martillo puede experimental, en principio, un flujo laminar con coeficiente de rozamiento $C_D = 0.5$ o un flujo inestable osciante con coeficiente de rozamiento $C_D = 0.75$.

1. Resuelva la ecuación de movimiento para el lanzamiento oblicuo de martillo. Deberá transformar las EDOs para los movimientos en x y y en un sistema de cuatro ecuaciones de primer orden. Considere lanzamienos desde una posición inicial $x_0 = 0$ y $y_0 = 2\,\mathrm{m}$, para un ángulo ideal $\phi = 45$ y encuentre la velocidad que produce la distancia del lanzamiento del récord mundial.

Solución.
Sea $\boldsymbol{r} = (x, y)$ la posición del martillo. Luego, por la segunda ley de Newton tenemos la siguiente ecuación que describe el movimiento:

$$m\ddot{\boldsymbol{r}} = \boldsymbol{F}_D + \boldsymbol{F}_g \tag{2}$$

Donde $\boldsymbol{F}_D$ es la fuerza de fricción y $\boldsymbol{F}_g$ es la fuerza gravitacional. Notemos que $\boldsymbol{F}_D$ debe tener sentido opuesto a la velocidad, para ello se debe multiplicar por un vector unitario en sentido opuesto a la velocidad, es decir:

$$\boldsymbol{F}_D = \frac{1}{2}\rho A C_D v^2 \left(-\frac{\boldsymbol{v}}{|\boldsymbol{v}|}\right)$$

Donde $\boldsymbol{v} = (\dot{x}, \dot{y})$ y $v^2 = \dot{x}^2 + \dot{y}^2$. Por componentes se sigue que:

$$m\frac{d^2x}{dt^2} = \frac{1}{2}\rho A C_D(\dot{x}^2 + \dot{y}^2)\left(-\frac{\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}}\right)$$
$$m\frac{d^2y}{dt^2} = \frac{1}{2}\rho A C_D(\dot{x}^2 + \dot{y}^2)\left(-\frac{\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}}\right) - mg$$

Es decir:

$$\frac{d^2x}{dt^2} = -k\dot{x}\sqrt{\dot{x}^2 + \dot{y}^2} \tag{3}$$

$$\frac{d^2y}{dt^2} = -k\dot{y}\sqrt{\dot{x}^2 + \dot{y}^2} - g \tag{4}$$

Donde $k = \frac{\rho A C_D}{2m}$. Ahora expresemos estas ecuaciones en la forma estándar. Para ello sean:

$$y^{(0)} = x(t), \quad y^{(1)} = \frac{dx}{dt}, \quad y^{(2)} = y(t), \quad y^{(3)} = \frac{dy}{dt}$$

Así de (3) se sigue que:

$$\frac{dy^{(0)}}{dt} = y^{(1)} = f^{(0)} \quad ; \quad \frac{dy^{(1)}}{dt} = -ky^{(1)}\sqrt{(y^{(1)})^2 + (y^{(3)})^2} = f^{(1)} \tag{5}$$

Y de (4) tenemos:

$$\frac{dy^{(2)}}{dt} = y^{(3)} = f^{(2)} \quad ; \quad \frac{dy^{(3)}}{dt} = -ky^{(3)}\sqrt{(y^{(1)})^2 + (y^{(3)})^2} - g = f^{(3)} \tag{6}$$

Con estas ecuaciones procederemos a resolver la el problema para encontrar la velocidad inicial del lanzamiento del martillo.

Para el programa utilizaremos las siguientes librerías:

In [1]:
```
from pylab import *
import numpy as np
from scipy.integrate import odeint
```

Ahora necesitamos definir las constantes:

In [2]:
```
# Definición de constantes
N = 1000        #No. de pasos
phi = np.pi/4   #Ángulo de lanzamiento
x0 = 0          #Posicion inicial en x
y0 = 2          #Posicion inicial en y
C = 0.5         #Coeficiente de rozamiento
m = 7.26        #Masa del martillo
rho = 1.2       #Densidd del aire
R = 0.06        #Radio de la esfera
g = 9.81        #ac. gravedad
k = rho*np.pi*R**2*C/(m*2.)  #Constante global
```

Notemos que el coeficiente de rozamiento `C` será 0.75 para el flujo inestable oscilante, 0.5 para el flujo laminar y 0.0 para cuando no haya fuerza de fricción.
En el método de Runge-Kutta para las EDO's requerimos de valores iniciales. En la siguiente función `cond_ini`, dada una velocidad inicial v_0 se genera un arreglo de 4 valores con las condiciones iniciales de la siguiente forma: $[x_0, v_{0_x}, y_0, v_{0_y}]$. Esta función tendrá como parámetro una velocidad inicial v_0 y con ello calculará las velocidades $v_{0_x} = v_0 \cos\phi$ y $v_{0_y} = v_0 \sin\phi$ para arrojar el arreglo ya descrito.

In [3]:
```
def cond_ini(v0):
    """
    Esta función establece las condiciones iniciales para el arreglo dada
    una velocidad inicial.

    Parámetros:
        v0: es la magnitud de la velocidad inicial
    Salida:
        array : [posición en x, vel en x, posición en y, vel en y]
    """
    m0 = np.float64(x0)           #Establecer estado inicial en x
    m1 = np.float64(v0*np.cos(phi)) #Vel. inicial en x
    m2 = np.float64(y0)           #Establecer estado inicial en y
    m3 = np.float64(v0*np.sin(phi))
    return array([m0,m1,m2,m3])
```

Lo siguiente a realizar es una función que evalúe a las funciones $f^{(0)}, f^{(1)}, f^{(2)}$ y $f^{(3)}$ descritas en (5) y (6) para un punto x, y en un intervalo de tiempo t.

In [4]:
```
def f(y,t):
    """
    Esta función evalúa a las funciones f^(i) de la forma estandar para un
    y en un tiempo t.

    Parámetros:
        y: arreglo [posicion en x, vel en x, posición en y, vel en y]
        t: arreglo con los puntos t_i a ser evaluados
    Salida:
        array : las funciones f^(i) evalada en los parámetors
                [f^(0), f^(1), f^(2), f^(3)]
    """
    f0 = y[1]
    f1 = -k*y[1]*(y[1]*y[1] + y[3]*y[3])**(1/2)
    f2 = y[3]
    f3 = -k*y[3]*(y[1]*y[1] + y[3]*y[3])**(1/2) - g
    return array([f0,f1,f2,f3])
```

Ahora nuestro propósito es generar una solución para una velocidad inicial v_0. La forma en como lo haremos será fijar el tiempo t a 1 segundo y obtendremos la solución de 0 a 1 s. Luego a través de un ciclo `while` iremos aumentando el tiempo hasta generar la solución cuyo último valor para la posición en y sea muy cercano al suelo. Es decir, que la solución final tendrá el momento en que el martillo se impacta en el suelo. Esto se observa en el siguiente algoritmo.

In [5]:
```
def time(t,N):
    """
    Esta función calcula la solución para una vel. inicial v_0 dada a un tiempo
    t, y a través de un ciclo while aumenta el tiempo hasta generar la solución
    en la que el último punto de posición de y sea cero (choque con el suelo).

    Parámetros:
        t: tiempo en el que se generará la solución de 0 a t.
        N: no. de pasos en los que se dividirá el tiempo [0,t] para generar la
           solución.

    Salida:
        respuesta: arreglo que contiene la solución numérica. En la primer
                   columna se encuentras los valores para x, en la segunda
                   columna la velocidad en x, en la tercera el valor de y,
                   en la cuarta la velocidad en y. Todos estos valores son en
                   cada punto del tiempo en el arreglo llamado tiempo.
        tiempo: arreglo que contiene los puntos del tiempo donde se generó la
                solución numérica.

    """
    i=1      #Contador de iteraciones
    dy = 1.0 #Delta de y
    while dy > 0.3:
        print(f'Iteracion y -> {i}')
        if i== 10000:       #Clausula para establecer un alto en caso de que calcule de más
            print('No. maximo de iteraciones alcanzado')
            break
        y = cond_ini(v0)                #Iniciamos el arreglo con las condiciones iniciales
        tiempo = linspace(0, t, N)      #Generamos los puntos para el tiemoo
        respuesta = odeint(f,y,tiempo)  #Generamos la solución numérica
        print(respuesta)

        dy = abs(respuesta[-1][2] - 0.) #Verifica la condición para y_final

        t += 0.01   #Aumentamos el tiempo
        i += 1      #Aumentamos el contador

    return respuesta, tiempo
```

En esta última parte hay que tener cuidado con el criterio $dy > 0.3$ para que se rompa el ciclo `while` ya que es posible que la solución nunca cumpla con esto. Esa es la razón por la que se ha colocado un contador para un número máximo de iteraciones. En caso de alcanzar el número máximo de iteraciones, habría que hacer más grande el criterio colocado de 0.3.
Lo siguiente a realizar para poder estimar la velocidad a la que se lanzó el martillo y con ello lograr una longitud de 86.74 m, será usar otro ciclo `while` y al igual que en la función `time`, se generará la solución para la velocidad inicial v_0 hasta que el martillo impacte con el suelo y con ello evaluaremos la condición de que el x final esté dentro de un intervalo aceptable para la longitud del récord. Esto se realiza en el siguiente algoritmo:

```
In [12]: #Semillas y criterio
         t = 1.0         #Tiempo inicial
         v0 = 10.0        #Semilla inicial para la vel inicial


         dx = 1.
         j=1
         while dx > 0.2:
             if j== 10000:       #Clausula para establecer un alto en caso de que calcule de más
                 print('No. maximo de iteraciones alcanzado')
                 break
             respuesta, tiempo = time(t,N)
             dx = abs(respuesta[-1][0] - 86.745) #Verifica la confición del x final
             v0 += 0.1  #Aumentamos la velocidad inicial
             t = 1.     #Reseteamos el tiempo a 1 seg
             print(f'x -> {j}')
             j+=1
```

Aquí hemos usado los valores semilla $v_0 = 10\,\text{m/s}$ y $t = 1\,\text{s}$ para encontrar los que cumplen las condiciones establecidas. Cuando termine de generar la solución estas variables (v_0, t) guardaran la velocidad inicial y el tiempo del recorrido para lograr un alcance de 86.74 m. Nuevamente hay que tener cuidado con el criterio $dx > 0.2$ para romper el ciclo `while`, en caso de no alcanzar el número máximo de iteraciones habría que aumentar el 0.2.

Ya con esto es posible calcular la velocidad inicial a la que se realizó el lanzamiento de martillo para generar la longitud del récord para una fuerza de fricción. Para este inciso supondremos que el lanzamiento se hizo en condiciones muy ideales en las que no existió fricción. Para generar la solución sin fricción basta fijar $C_D = 0.0$ en la definición de las constantes. Realizando esto e imprimiendo los resultados se obtiene:

```
In [21]: print(f'Vel. inicial = {v0} m/s')
         print(f'x final = {respuesta[-1][0]} m')
         print(f'y final = {respuesta[-1][2]} m')
         print(f'Tiempo de vuelo = {tiempo[-1]} s')

         Vel. inicial = 29.000000000000163 m/s
         x final = 86.8503903992372 m
         y final = 0.25382789923661164 m
         Tiempo de vuelo = 4.249999999999953 s
```

Así hemos obtenido la solución para el lanzamiento sin fricción.Entonces la velocidad a la que se realizó en lanzamiento fue $v_0 \approx 29.0\,\text{m/s}$ para el caso ideal sin fricción. Si graficamos la posición $y(t)$ contra el tiempo obtenemos:

In [20]:
```
# Ahora graficamos
title('Gráfico de y(t) vs tiempo sin frición')
plot(tiempo,respuesta[:,2], 'g')
xlabel('$t$ [s]')
grid()
ylabel('$y(t)$ [m]')
show()
```

---

2. Calcule y grafique la dependencia en el tiempo de la altitud del martillo y su trayectoria $y = y(t)$ en los tres régimenes:

   (a) Sin fricción.

   (b) Flujo laminar.

   (c) Flujo inestable oscilante.

Solución.
Para este inciso usaremos la velocidad inicial calculada en el ejercicio anterior. Es decir, $v_0 \approx 29.0\,\mathrm{m/s}$. A partir de esto generaremos la solución numérica de la trayectoria del martillo desde que es lanzado hasta que se impacta con el suelo. Esto para cada uno de los casos descritos. En el siguiente algoritmo generamos las soluciones numéricas para cada uno de los casos.

In [32]:
```
###################### Cálculo si fricción
C = 0.     #Sin fricción
k = rho*np.pi*R**2*C/(m*2.)  #Constante global
t = 1.0        #Reseteamos el tiempo

res_noFric, tiempo_noFric = time(t,N)

print('-'*50)
print('Caso sin fricción:')
print(f' \tVel. inicial = {v0} m/s')
print(f'\tx final = {res_noFric[-1][0]} m')
print(f'\ty final = {res_noFric[-1][2]} m')
print(f'\tTiempo de vuelo = {tiempo_noFric[-1]} s')
```

```
####################### Cálculo para el flujo laminar
C = 0.5     #Flujo laminar
k = rho*np.pi*R**2*C/(m*2.)  #Constante global
t = 1.0        #Reseteamos el tiempo

res_laminar, tiempo_laminar = time(t,N)

print('-'*50)
print('Caso con flujo laminar:')
print(f' \tVel. inicial = {v0} m/s')
print(f'\tx final = {res_laminar[-1][0]} m')
print(f'\ty final = {res_laminar[-1][2]} m')
print(f'\tTiempo de vuelo = {tiempo_laminar[-1]} s')
```

```
####################### Cálculo para el flujo laminar
C = 0.75     #Flujo laminar
k = rho*np.pi*R**2*C/(m*2.)  #Constante global
t = 1.0        #Reseteamos el tiempo

res_ines, tiempo_ines = time(t,N)

print('-'*50)
print('Caso con flujo laminar:')
print(f' \tVel. inicial = {v0} m/s')
print(f'\tx final = {res_ines[-1][0]} m')
print(f'\ty final = {res_ines[-1][2]} m')
print(f'\tTiempo de vuelo = {tiempo_ines[-1]} s')
```

Estas soluciones nos producen los siguientes resultados:

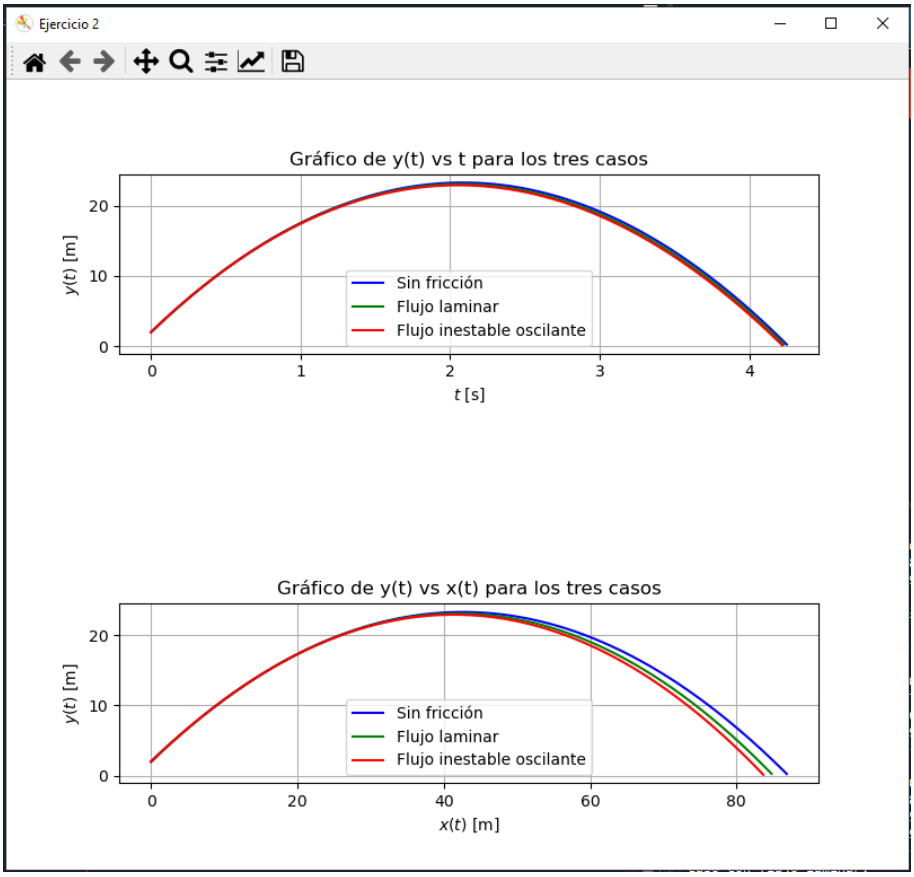
```
--------------------------------------------------
Caso sin fricción:
        Vel. inicial = 29.000000000000163 m/s
        x final = 87.56103271432971 m
        y final = 0.128658214329036 m
        Tiempo de vuelo = 4.2699999999999525 s
--------------------------------------------------
Caso con flujo laminar:
        Vel. inicial = 29.000000000000163 m/s
        x final = 84.77961562974883 m
        y final = 0.2734869175635737 m
        Tiempo de vuelo = 4.229999999999953 s
--------------------------------------------------
Caso con flujo laminar:
        Vel. inicial = 29.000000000000163 m/s
        x final = 83.6481016390504 m
        y final = 0.1447309078918786 m
        Tiempo de vuelo = 4.219999999999954 s
```

Se puede observar como al agregar fricción la longitud x recorrida es menor y a su vez el tiempo de vuelo también disminuye.
Ahora veamos gráficamente como difieren las soluciones. En el siguiente algoritmo se generan dos subgráficas: la primera muestra la superposición de los tres casos de $y(t)$ vs t para la velocidad v_0 obtenida en el ejercicio 1. La segunda subgráfica contiene la superposición de $y(t)$ vs $x(t)$ para los tres casos.

```
In [34]: # Ahora graficamos las soluciones usando subgráficas
figure('Ejercicio 2',figsize=(8,7))
subplot(3,1,1)    # Definimos la primera gráfica
plot(tiempo_noFric,res_noFric[:,2], 'b',label='Sin fricción')
plot(tiempo_laminar,res_laminar[:,2], 'g',label='Flujo laminar')
plot(tiempo_ines,res_ines[:,2], 'r',label='Flujo inestable oscilante')
title('Gráfico de y(t) vs t para los tres casos')
xlabel('$t$ [s]')
ylabel('$y(t)$ [m]')
grid(True)
legend()

subplot(3,1,3)    # Definimos la segunda gráfica
plot(res_noFric[:,0],res_noFric[:,2], 'b',label='Sin fricción')
plot(res_laminar[:,0],res_laminar[:,2], 'g',label='Flujo laminar')
plot(res_ines[:,0],res_ines[:,2], 'r',label='Flujo inestable oscilante')
title('Gráfico de y(t) vs x(t) para los tres casos')
xlabel('$x(t)$ [m]')
ylabel('$y(t)$ [m]')
grid(True)
legend()
```

Esto nos produce los siguientes dos subgráficos:



En el primer subgráfico se puede ver que el tiempo de recorrido es casi el mismo en los tres casos. Sin embargo, en el segundo gráfico se observa que existe una diferencia significativa en la longitud x recorrida. Como es de esperarse, al ser mayor la fuerza de fricción es menor la longitud total recorrida. En el siguiente ejercicio obtendremos la diferencia de longitudes.

3. En el inciso anterior, estime la cantidad en que es influenciada la distancia del lanzamiento por la fricción.

Solución.
Este inciso es muy simple. Debido a la forma en como hemos realizado nuestro programa, basta comparar la última entrada de la primer columna de los arreglos `res_noFric`, `res_laminar` y `res_ines`. En esa posición se encuentra la posición x donde el martillo chocó con el suelo al final del recorrido. Tomaremos el caso sin fricción como el que describe el lanzamiento de martillo, de esa forma obtendremos la diferencia de longitud para el flujo laminar y flujo inestable oscilante con respecto al caso sin fricción. Esto se hace en el siguiente algoritmo con su correspondiente resultado:

```
In [35]: ##################Obtención de las diferencia de longitudes
         dif_laminar = res_noFric[-1][0] - res_laminar[-1][0]
         dif_ines = res_noFric[-1][0] - res_ines[-1][0]
         print('-'*50)
         print(f'Diferencia para el flujo laminar: {dif_laminar} m')
         print(f'Diferencia para el flujo inestable oscilante: {dif_ines} m')

         --------------------------------------------------
         Diferencia para el flujo laminar: 2.7814170845808803 m
         Diferencia para el flujo inestable oscilante: 3.9129310752793174 m
```

Por lo que en presencia de flujo laminar el martillo llegará aproximadamente 2.78 m antes del récord y en presencia de flujo inestable oscilante llegará 3.91 m antes.