



Taller Script Python

COLOQUE SU NOMBRE

2025-01-19

1. Objetivos

- Desarrollar un script de python con paradigma estructurado

2. Problema

Usted es el científico de datos para un sistema de lanzamiento de cohetes. Para esto, se le solicita que determine gráficamente el alcance del cohete según al menos tres ángulos de entrada, un tiempo máximo de vuelo t . Se considera que existe una fuerza de resistencia al movimiento debida al viento y representada mediante un coeficiente b . Escribir un programa que permita obtener el alcance del cohete gráficamente, mediante la resolución de ecuaciones diferenciales ordinarias. Los parámetros como Velocidad inicial V , coeficiente de resistencia B , tiempo máximo t_{\max} y ángulos de lanzamiento se ingresan desde la línea de comando. Siga las instrucciones proporcionadas en la Sección 3 para completar el taller.

3. Instrucciones

1. Obtener las expresiones matemáticas para las ecuaciones diferenciales ordinarias. Vea la Sección 5.1
2. Investigar cómo funciona la librería `solve_ivp` del paquete `scipy.integrate`
3. Investigar cómo funciona la librería `argparse` para controlar el ingreso de valores y opciones desde la línea de comandos.
4. Cree una carpeta con los archivos del proyecto denominada `cohete`.
5. Dentro de la carpeta cree un archivo `main.py` con el código proporcionado en la Sección 5.2
6. Reorganizar el código proporcionado de ejemplo para conformar un proyecto de código estructurado. Las funciones estarán dentro de una carpeta `utils` en el archivo `projectile_functions.py`. El código principal se coloca en el archivo `main.py` conforme al siguiente esquema de archivos.
 - a) Con YASnippet habilitado escriba `imp` seguido de C-TAB para escribir `import`
 - b) Escriba `main` seguido de la secuencia de comandos C-TAB. Esto genera una función llamada `main`
 - c) escriba `ifm` seguido de la secuencia C-TAB para generar la condición de ejecución del programa Python como programa principal
 - d) Use `fd` seguido de C-TAB para escribir una función con *docstrings* y use `r` seguido de C-TAB para escribir `return`
 - e) Utilice `pars` C-TAB para crear un parser y `args` C-TAB para crear un argumento de entrada del programa. Importe la librería `argparse`

```

/
├── main.py
├── utils
│   ├── __init__.py
│   └── projectile_functions.py

```

4. Entregables del taller

1. Archivo .zip con el archivo `main.py` y la librería desarrollada `projectile_functions.py` y el gráfico obtenido de una carrera.
2. Completar en la sección [] de este documento una breve descripción de lo que hacen las librerías revisadas en este taller.
3. Una vez redactado el punto anterior suba el archivo .ORG y el .PDF

5. Recursos

Puede consultar más información en el siguiente vídeo y en GitHub

5.1. Desarrollo Matemático

La fuerza de resistencia del viento se puede considerar que es proporcional al cuadrado de la velocidad y en sentido opuesto a ésta. De ahí que las fuerzas actuando en el sistema quedarían determinadas por la ecuación

$$\vec{F}_{\text{net}} = \vec{F}_g + \vec{F}_f = -mg\hat{y} - b|\vec{v}|\vec{v} \quad (1)$$

La velocidad está dada por $\vec{v} = \dot{x}\hat{i} + \dot{y}\hat{j}$

En consecuencia la Fuerza neta del sistema queda:

$$\vec{F}_{\text{net}} = -mg\hat{y} - b\sqrt{\dot{x}^2 + \dot{y}^2}(\dot{x}\hat{i} + \dot{y}\hat{j}) \quad (2)$$

Que en forma vectorial se puede escribir como

$$\vec{F}_{\text{net}} = \begin{bmatrix} -b\sqrt{\dot{x}^2 + \dot{y}^2}\dot{x} \\ -mg - b\sqrt{\dot{x}^2 + \dot{y}^2}\dot{y} \end{bmatrix} \quad (3)$$

Dado que $\vec{F}_{\text{net}} = m\vec{a} = m\langle\ddot{x}, \ddot{y}\rangle$, se tiene

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} -b\sqrt{\dot{x}^2 + \dot{y}^2}\dot{x} \\ -mg - b\sqrt{\dot{x}^2 + \dot{y}^2}\dot{y} \end{bmatrix} \quad (4)$$

De donde se obtienen el par de ecuaciones diferenciales

$$\ddot{x} = -\frac{b}{m}\sqrt{\dot{x}^2 + \dot{y}^2}\dot{x} \quad (5)$$

$$\ddot{y} = -g - \frac{b}{m}\sqrt{\dot{x}^2 + \dot{y}^2}\dot{y} \quad (6)$$

Definiendo $x' = x/g$ and $y' = y/g$, se tiene entonces

$$\ddot{x}' = -\frac{bg}{m}\sqrt{\dot{x}'^2 + \dot{y}'^2}\dot{x}' \quad (7)$$

$$\ddot{y}' = -1 - \frac{bg}{m} \sqrt{\dot{x}'^2 + \dot{y}'^2} \dot{y}' \quad (8)$$

Sea $B \equiv bg/m$, entonces considerando el cambio de notación y eliminado la ' para simplificar, se propone resolver un sistema de ecuaciones de primer orden ODE siguiente:

$$\dot{x} = v_x \quad (9)$$

$$\dot{v}_x = -B\sqrt{\dot{x}^2 + \dot{y}^2} \dot{x} \quad (10)$$

$$\dot{y} = v_y \quad (11)$$

$$\dot{v}_y = -B\sqrt{\dot{x}^2 + \dot{y}^2} \dot{y} \quad (12)$$

5.2. Código Inicial Python

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

def dSdt(t,S,B):
    """
    Use fd seguido de C-TAB para generar la funcion con docstrings
    Define el sistema de ecuaciones diferenciales para el movimiento del proyectil
    con resistencia al aire
    Parametros:
        t (float): Tiempo
        S (list): Estado actual [x, vx, y, vy]
        B (float): Coeficiente de resistencia del aire. Varía desde 0.0 a 1.0
    Retorna:
        list: Derivadas [dx/dt, dvx/dt, dy/dt, dvdy/dt]
    """
    x, vx, y, vy = S
    return [vx,
            -B*np.sqrt(vx**2+vy**2)*vx,
            vy,
            -1-B*np.sqrt(vx**2+vy**2)*vy]

B = 0
V = 1
t1 = 40*np.pi/180
t2 = 45*np.pi/180
t3 = 50*np.pi/180

sol1 = solve_ivp(dSdt, [0, 2],
                  y0=[0,V*np.cos(t1),0,V*np.sin(t1)],
                  t_eval=np.linspace(0,2,1000),
                  args=(B,)) # atol=1e-7, rtol=1e-4
sol2 = solve_ivp(dSdt, [0, 2],
                  y0=[0,V*np.cos(t2),0,V*np.sin(t2)],
                  t_eval=np.linspace(0,2,1000),
                  args=(B,)) #atol=1e-7, rtol=1e-4
sol3 = solve_ivp(dSdt, [0, 2],
```

```

y0=[0,V*np.cos(t3),0,V*np.sin(t3)],
t_eval=np.linspace(0,2,1000),
args=(B,)) #atol=1e-7, rtol=1e-4

plt.plot(sol1.y[0],sol1.y[2], label=r'$\theta_0=40^\circ$')
plt.plot(sol2.y[0],sol2.y[2], label=r'$\theta_0=45^\circ$')
plt.plot(sol3.y[0],sol3.y[2], label=r'$\theta_0=50^\circ$')
plt.ylim(bottom=0)
plt.legend()
plt.xlabel('$x/g$', fontsize=20)
plt.ylabel('$y/g$', fontsize=20)
plt.show()

```

6. Descripción librerías usadas

6.1. SOLVE_{IVP}

6.2. ARGPARSE

6.3. NUMPY