Proyecto N1

Métodos Numéricos

Integrantes:

Andino Padilla José Luis

Alexander Alexander Paillacho

Luis Enrique Pérez

Fabian Simbaña

Docente:

Fecha de entrega: 18 DE JUNIO DE 2024

Índice De Contenido

Índice De Figuras

Introducción:

Este proyecto se enfoca en la simulación y visualización de la trayectoria de una bomba lanzada desde un avión hacia una montaña, utilizando técnicas interactivas y visuales para mejorar la comprensión de los principios físicos subyacentes. Al implementar una interfaz de usuario basada en sliders, se permite al usuario ajustar de manera intuitiva parámetros críticos como la velocidad del avión, la altura sobre el nivel del mar, la distancia horizontal al pico de la montaña, la altura de la montaña y el ángulo de la montaña.

El objetivo principal de este proyecto es proporcionar una herramienta educativa interactiva que facilite la exploración y el análisis del movimiento parabólico bajo la influencia de la gravedad. Para lograr esto, se utiliza la biblioteca ipywidgets en combinación con matplotlib para crear gráficos que se actualizan en tiempo real conforme el usuario modifica los parámetros. Además, se incorpora una imagen personalizada para representar el punto de impacto de la bomba, mejorando la experiencia visual y haciendo la simulación más atractiva y comprensible. Este proyecto no solo ayuda a visualizar conceptos físicos, sino que también sirve como un ejemplo práctico de cómo combinar matemáticas, física y programación para resolver problemas complejos y crear aplicaciones interactivas. La solución implementada ofrece una plataforma robusta y

adaptable que puede extenderse para incluir análisis más detallados y simulaciones más complejas en el futuro.

Metodología:

1. Descripción de la solución.

La solución implementada permite simular y visualizar la trayectoria de una bomba lanzada desde un avión hacia una montaña utilizando sliders interactivos para ajustar los parámetros de entrada. Utilizando la biblioteca *ipywidgets*, se crean sliders para modificar la velocidad del avión, la altura sobre el nivel del mar, la distancia horizontal al pico de la montaña, la altura de la montaña y el ángulo de la montaña.

Al ajustar estos sliders, la gráfica se actualiza automáticamente para mostrar la nueva trayectoria de la bomba y el punto de impacto, el cual es representado por una imagen personalizada en lugar de un simple punto. La implementación asegura que solo una imagen se actualice en cada cambio de los sliders, proporcionando una interfaz de usuario intuitiva y visualmente atractiva en Google Colab.

2. Desarrollo matemático.

El desarrollo matemático para calcular la trayectoria de la bomba lanzada desde un avión hacia una montaña se basa en los principios de la física clásica, específicamente en el movimiento parabólico bajo la influencia de la gravedad. A continuación, se detalla el desarrollo matemático:

- **Movimiento en el eje horizontal (x)**

En el eje horizontal, la bomba se mueve con una velocidad constante igual a la velocidad inicial del avión (v_0),ya que no hay aceleración horizontal (ignoramos la resistencia del aire).
La posición horizontal de la bomba en cualquier instante t se da por:

$$x(t) = v_0 \cdot t$$

- **Movimiento en el eje vertical (y)**

En el eje vertical, la bomba se ve afectada por la aceleración debida a la gravedad (g), que actúa hacia abajo. La ecuación de movimiento para la posición vertical es:

$$y(t) = h_a - \frac{1}{2} g t^2$$

Donde:

- h_a es la altura inicial del avión sobre el nivel del mar.
- g es la aceleración debida a la gravedad ($9.81\ m/s^2$).
- t es el tiempo en segundos desde que la bomba es soltada.

- **Tiempo de impacto**

El tiempo que tarda la bomba en impactar el suelo (nivel del mar) se calcula resolviendo la ecuación

$$y(t) = 0:$$

$$0 = h_a - \frac{1}{2} g t^2$$

$$t = \sqrt{\frac{2h_0}{g}}$$

- **Distancia horizontal de impacto**

La distancia horizontal recorrida por la bomba hasta el impacto se determina usando el tiempo de impacto t_{mpacto} :

$$x_{impacto} = v_0 \cdot t_{impacto}$$

$$x_{impacto} = v_0 \cdot \sqrt{\frac{2h_0}{g}}$$

- **Coordenadas de la montaña**

La montaña se modela como un triángulo isósceles con altura h_m y un ángulo de inclinación α respecto al horizonte. La base de la montaña se puede calcular usando trigonometría:
La distancia horizontal desde el pie de la montaña al pico, en ambos lados, es:

$$base_{parcial} = \frac{h_m}{\tan(\alpha)}$$

La montaña entonces tiene vértices en:

$$(d - base_{parcial}, 0)$$
$$(d, h_m)$$
$$(d + base_{parcial}, 0)$$

- **Trayectoria de la bomba**

La trayectoria de la bomba se obtiene evaluando las ecuaciones $x(t)$ y $y(t)$ para valores de t desde 0 hasta $t_{impacto}$

- **Resumen del desarrollo matemático:**

1. Calcular el tiempo de impacto:

$$t_{impacto} = \sqrt{\frac{2h_0}{g}}$$

2. Calcular la distancia horizontal de impacto

$$x_{impacto} = v_0 \cdot t_{impacto}$$

3. Determinar las coordenadas de la montaña.
4. Evaluar las ecuaciones

$$x(t) = v_0 \cdot t$$
$$y(t) = h_a - \frac{1}{2} g t^2$$

para graficar la trayectoria de la bomba.

3. Diagrama de flujo

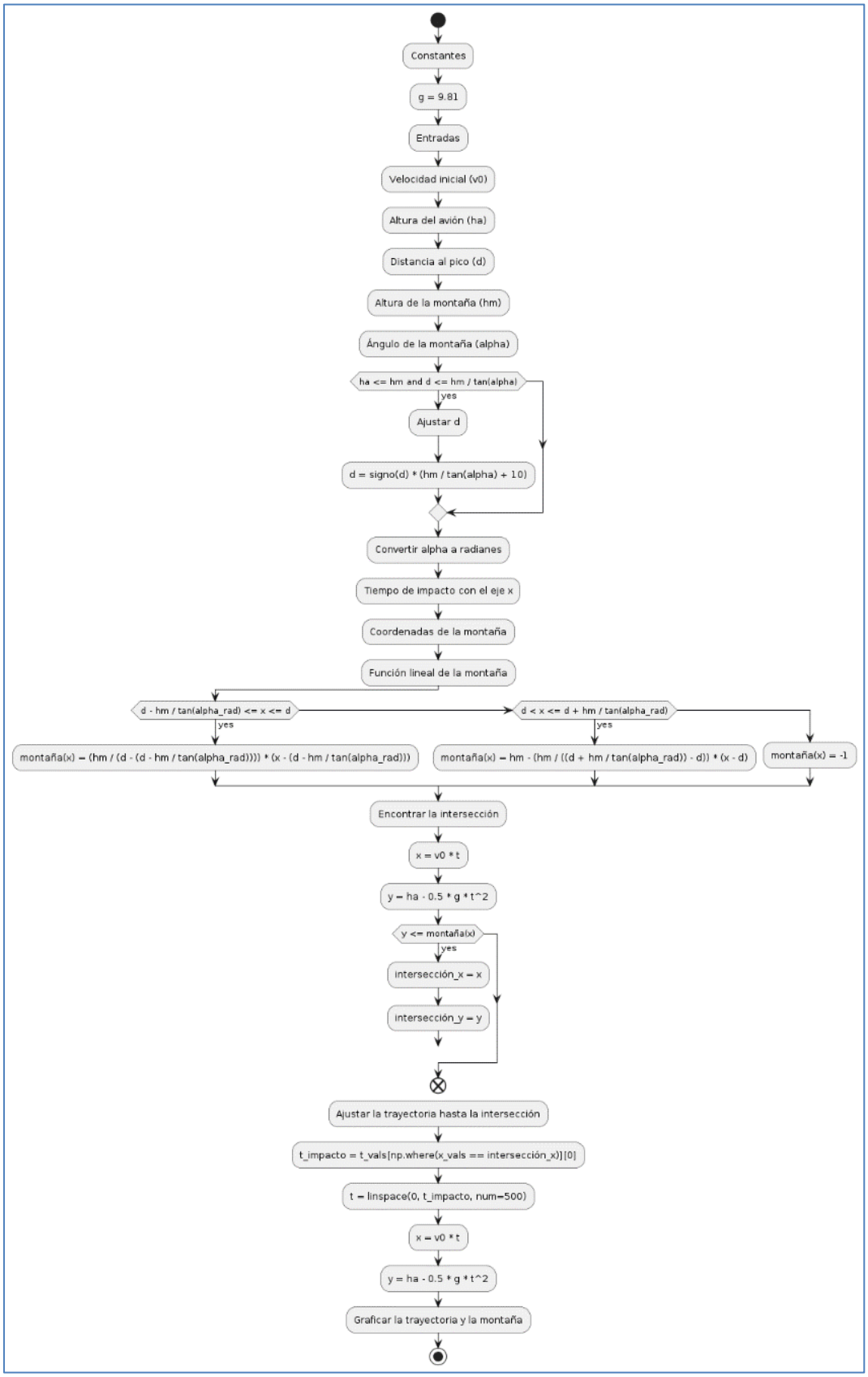

Figura 1 Diagrama de Flujo

4. Pseudocódigo.

```
# Importar bibliotecas necesarias
importar numpy como np
importar matplotlib.pyplot como plt
importar ipywidgets como widgets
desde IPython.display importar display, clear_output
desde matplotlib.offsetbox importar OffsetImage, AnnotationBbox

# Constante de gravedad
g = 9.81 # m/s^2

# Definir funciones
función calcular_trayectoria(v0, ha, d, hm, alpha):
    alpha_rad = convertir_a_radianes(alpha)
    t_impacto = sqrt(2 * (ha - hm) / g)
    x_impacto = v0 * t_impacto
    t = linspace(0, t_impacto, num=500)
    x = v0 * t
    y = ha - 0.5 * g * t^2
    devolver x, y, x_impacto

función graficar_trayectoria(v0, ha, d, hm, alpha):
    x, y, x_impacto = calcular_trayectoria(v0, ha, d, hm, alpha)
    alpha_rad = convertir_a_radianes(alpha)
    montaña_x = calcular_montaña_x(d, hm, alpha_rad)
    montaña_y = calcular_montaña_y(hm)
    crear_figura()
    graficar_montaña(montaña_x, montaña_y)
    graficar_trayectoria_bomba(x, y)
    agregar_imagen_en_punto_impacto(x_impacto, 0)
    mostrar_grafica()

# Crear y configurar sliders
crear_slider_velocidad()
crear_slider_altura_avion()
crear_slider_distancia_pico()
crear_slider_altura_montaña()
crear_slider_angulo_montaña()

# Definir función de actualización
función update_plot(cambio):
    obtener_valores_desde_sliders()
    clear_output()
    mostrar_sliders()
    graficar_trayectoria()

# Conectar sliders con la función de actualización
conectar_sliders_con_update_plot()

# Mostrar sliders y gráfica inicial
mostrar_sliders()
update_plot(None)

# Fin del programa
```

Figura 2 Pseudocódigo Principal

5. Detalles importantes de la implementación.

La implementación de la simulación de la trayectoria de una bomba lanzada desde un avión hacia una montaña se realizó utilizando Python en combinación con bibliotecas como numpy, matplotlib e ipywidgets. A continuación, se detallan los aspectos más importantes:

Uso de Sliders Interactivos:

- Se crearon sliders para permitir al usuario ajustar parámetros críticos como la velocidad del avión, la altura sobre el nivel del mar, la distancia horizontal al pico de la montaña, la altura de la montaña y el ángulo de inclinación de la montaña.
- Los sliders se implementaron utilizando la biblioteca ipywidgets y se conectaron a funciones de actualización que recalculan y actualizan la gráfica de la trayectoria de la bomba en tiempo real cada vez que se modifica un parámetro.

Funciones Matemáticas y de Visualización:

- Cálculo de Trayectoria: Se definió una función calcular trayectoria que utiliza ecuaciones del movimiento parabólico para determinar la posición de la bomba en el tiempo.
- Graficación de la Trayectoria: La función graficar_trayectoria se encargó de representar gráficamente tanto la montaña como la trayectoria de la bomba, incluyendo el punto de impacto, utilizando matplotlib.
- Coordenadas de la Montaña: La montaña se modeló como un triángulo isósceles y se calcularon sus coordenadas utilizando trigonometría para determinar su base y altura.

Interactividad y Visualización:

- La interfaz de usuario se diseñó para ser intuitiva, permitiendo una exploración visual interactiva. Al mover los sliders, la gráfica se actualiza automáticamente, mostrando la nueva trayectoria y el punto de impacto de la bomba.
- Se incorporó una imagen personalizada para representar el punto de impacto, mejorando la experiencia visual y haciéndola más atractiva y comprensible.

Consideraciones y Restricciones:

- Para simplificar los cálculos, se ignoró la resistencia del aire.
- Se establecieron restricciones para asegurar la validez de los datos de entrada, como límites en los sliders para evitar valores no válidos.
- Se garantizó que la bomba no excediera los límites de la simulación, asegurando así una representación precisa y controlada del fenómeno.

Resultados:

```python
import numpy as np
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import display, clear_output

# Constantes
g = 9.81  # Aceleración debida a la gravedad en m/s^2

# Función para calcular la trayectoria de la bomba
def calcular_trayectoria(v0, ha, d, hm, alpha):
    alpha_rad = np.radians(alpha)
    t_impacto = np.sqrt(2 * (ha - hm) / g)
    x_impacto = v0 * t_impacto

    # Trayectoria de la bomba
    t = np.linspace(0, t_impacto, num=500)
    x = v0 * t
    y = ha - 0.5 * g * t**2

    return x, y, x_impacto
```

```
# Función para graficar la montaña y la trayectoria de la bomba
def graficar_trayectoria(v0, ha, d, hm, alpha):
    x, y, x_impacto = calcular_trayectoria(v0, ha, d, hm, alpha)
    alpha_rad = np.radians(alpha)

    # Coordenadas de la montaña
    montaña_x = [d - hm / np.tan(alpha_rad), d, d + hm / np.tan(alpha_rad)]
    montaña_y = [0, hm, 0]

    plt.figure(figsize=(10, 6))
    plt.plot(montaña_x, montaña_y, label='Montaña')
    plt.plot(x, y, label='Trayectoria de la bomba')
    plt.scatter([x_impacto], [0], color='red', zorder=5, label='Punto de impacto')
    plt.xlabel('Distancia horizontal (m)')
    plt.ylabel('Altura (m)')
    plt.legend()
    plt.grid()
    plt.show()

# Crear sliders para los parámetros
v0_slider = widgets.FloatSlider(value=100, min=10, max=500, step=10, description='Vel. 🛩
(m/s):')
ha_slider = widgets.FloatSlider(value=80, min=10, max=500, step=10, description='Alt. 🛩
(m):')
d_slider = widgets.FloatSlider(value=50, min=10, max=500, step=10, description='Dist al pico
(m):')
hm_slider = widgets.FloatSlider(value=75, min=10, max=500, step=10, description='Alt. ⛰
(m):')
alpha_slider = widgets.FloatSlider(value=60, min=1, max=89, step=1, description='Ángulo
(grados):')

# Función que se llama cuando se actualiza cualquier slider
def update_plot(change):
    v0 = v0_slider.value
    ha = ha_slider.value
    d = d_slider.value
    hm = hm_slider.value
    alpha = alpha_slider.value

    clear_output(wait=True)
    display(v0_slider, ha_slider, d_slider, hm_slider, alpha_slider)
    graficar_trayectoria(v0, ha, d, hm, alpha)

# Conectar sliders con la función de actualización de la gráfica
v0_slider.observe(update_plot, names='value')
ha_slider.observe(update_plot, names='value')
d_slider.observe(update_plot, names='value')
hm_slider.observe(update_plot, names='value')
alpha_slider.observe(update_plot, names='value')

# Mostrar sliders y la gráfica inicial
display(v0_slider, ha_slider, d_slider, hm_slider, alpha_slider)
update_plot(None)
```

Figura 3 Código de implementado en Python

Conclusiones:

- En este proyecto, se utilizaron varios métodos de cálculos algebraicos, incluyendo diversas fórmulas del cálculo del Movimiento Parabólico. No se consideró la resistencia del aire para simplificar los cálculos y obtener resultados más precisos. Además, se aplicaron nuestros conocimientos en física y programación para lograr los resultados deseados.
- Para la realización de este proyecto, fue necesario considerar varias restricciones. Al implementar sliders, se debía tener cuidado para evitar la entrada de datos no válidos. Además, se establecieron restricciones esenciales para asegurar el correcto funcionamiento del programa, tales como la delimitación del suelo y la montaña. Esto garantizó que, al caer la bomba, no excediera estos límites, asegurando así una simulación precisa y controlada.

Referencias

Parabólico, M. (1996). Movimiento parabólico. *LABORATORIO DE FISICA*, 10.
Fernandez, A. (2013). *Python 3 al descubierto*. Alfaomega Grupo Editor.