20/10/2023

Reto

Programación de trayectorias de proyectiles en 2D utilizando un lenguaje de alto nivel

Integrantes:

Diego Iván Rodríguez Núñez A01644772

David Martínez Martín del Campo A01645721

Gonzalo Flores García A01644771

Arturo Gomez Gomez A07106692

Instituto Tecnológico y de Estudios Superiores de Monterrey

F1004B.212 Modelación computacional del movimiento

Docente: Carlos Santana Velásquez

20 de octubre de 2023

Fórmulas y consideraciones físicas:

Fuerzas que actúan sobre el proyectil:

- Peso
- Resistencia al aire

Cabe recalcar que la resistencia al aire depende de la dirección del movimiento, la fricción siempre tendrá dirección contraria al movimiento, razón por la cuál se encuentran funciones trigonométricas dentro de las fórmulas.

Segunda ley de Newton:

$$\Sigma F = ma$$
$$a = (\Sigma F)/m$$
$$a = (Vf - Vi)/\Delta t$$
$$(Vf - Vi)/\Delta t = (\Sigma F)/m$$
$$Vf = Vi + \Delta t\,((\Sigma F)/m)$$

Esta es un uso del método numérico de Euler:

$$y1 = y0 + h\,f(x, y)$$

Fuerzas aplicadas en x:

Resistencia al aire

$$Vxf = Vxi + \Delta t\,((\Sigma Fx)/m)$$
$$Vxf = Vxi + \Delta t\,((-Fa * cos(Vxi, Vyi))/m)$$

Fuerzas aplicadas en y:

- Resistencia al aire
- Peso

$$Vyf = Vyi + \Delta t\,((\Sigma Fy)/m)$$
$$Vyf = Vyi + \Delta t\,((-Fa * sen(Vxi, Vyi) - W)/m)$$

Donde:

$$W = mg$$
$$Fa = (D * \rho * A * Vi * Vi)/2$$

D = Coeficiente de arrastre
ρ = Densidad del fluido
A = Área de contacto del proyectil

Posición en y

$$yf = (Vfy * \Delta t) + yi$$

Posición en x

$$xf = (Vfx * \Delta t) + xi$$

$$Vxf = Vxi + \Delta t\,((- Fa * cos(Vxi, Vyi))/m)$$

$$Vyf = Vyi + \Delta t\,((- Fa * sen(Vxi, Vyi) - mg)/m)$$

$$xf = (Vfx * \Delta t) + xi$$

$$yf = (Vfy * \Delta t) + yi$$

1. Inicialización de Variables: El código comienza inicializando algunas variables y listas, como trayectorias_x y trayectorias_y, que se utilizarán para almacenar las coordenadas X e Y de las trayectorias de los proyectiles.

2. Entrada de Datos del Usuario: Utiliza bucles while anidados para solicitar al usuario la entrada de datos relacionados con el proyectil, como la velocidad inicial (vi), el ángulo de lanzamiento (ang), y el coeficiente de arrastre (cd). Estos bucles se utilizan para asegurarse de que los valores ingresados sean válidos.

3. Cálculo de Parámetros del Proyectil: Después de obtener los valores de vi, ang, y cd, el código calcula varios parámetros necesarios para la simulación, como la fuerza de arrastre (f_arrastre), la altura inicial (altura), y el paso de tiempo (dt).

4. Simulación de la Trayectoria: El código simula la trayectoria del proyectil utilizando bucles while. Mientras la posición en Y (pos_y) sea mayor o igual a cero, se actualizan las posiciones en X e Y en cada iteración. Para hacerlo, se utilizan las ecuaciones de movimiento en dos dimensiones y se tiene en cuenta la fuerza de arrastre.

5. Animación de Trayectorias: A medida que se calculan las posiciones en cada paso de tiempo, el código agrega estas posiciones a las listas trayectoria_x_actual y trayectoria_y_actual. Luego, dibuja la trayectoria actual en una figura de animación utilizando la función plot, lo que crea una animación de la trayectoria del proyectil en tiempo real.

6. Visualización de Datos: Después de que el proyectil aterriza, el código muestra la velocidad en X e Y y la posición final en X. Luego, almacena la trayectoria actual en las listas trayectorias_x y trayectorias_y.

7. Pregunta al Usuario: El código pregunta al usuario si desea simular otra erupción. Si el usuario responde "no" (ingresa 0), se establece la bandera flag en false, lo que finaliza la simulación.