Pedro José Romero Gombau

11 de marzo de 2025

SimLaunch

SimLaunch

Procesos de simulación de la trayectoria de una proyectil en una lanzadera electromagnética y dimensionamiento de alimentación eléctrica, 11 de marzo de 2025

Tecnun
Universidad
de Navarra

1. Introducción

Este proyecto consiste en una aplicación gráfica con distintas pestañas (*tabs*), cada una dedicada a un aspecto concreto de la simulación y el dimensionamiento de una lanzadera electromagnética multipaso (*coilgun*). A lo largo de este documento se describen las partes principales y los modelos físicos asociados en el siguiente orden:

1. **geometry_viewer**: Herramientas de visualización 3D y 2D de geometrías.
2. **TabCoil**: Cálculo de inductancias y resistencias de la bobina.
3. **TabDrag**: Cálculo de coeficiente aerodinámico para la fuerza de arrastre.
4. **TabPower**: Dimensionamiento del circuito de cada etapa (condensadores) y animación del proyectil a través de n etapas.
5. **TabSearch**: Búsqueda y optimización de ángulo de lanzamiento.
6. **TabSimulator**: Simulador principal de la trayectoria (tiro parabólico con o sin rozamiento) y cálculo de la energía mecánica.

2. geometry_viewer.py

Este módulo implementa dos funciones de visualización que se utilizan como herramientas de apoyo en otras pestañas:

- `plot_geometry_in_frame(...)`: Dibuja en 3D diferentes geometrías elementales (*Prisma cuadrado*, *Cilindro*, *Esfera*) dentro de un `Frame` de Tkinter, usando `matplotlib`.
- `plot_coil_in_frame(...)`: Dibuja en 2D la sección de una bobina (vista de corte) con sus capas y espiras, representando de forma aproximada la distribución del hilo conductor.

Estas se usan en **TabCoil** (para la vista 2D de la bobina) y potencialmente en otras partes que requieran un esquema visual de la geometría.

3. TabSimulator

3.1. Objetivo

La pestaña `TabSimulator` permite simular el movimiento de un proyectil lanzado con un ángulo α, permitiendo tanto el modelo clásico de tiro parabólico sin resistencia del aire como una versión más realista que incluye un término de arrastre proporcional a la velocidad.

El análisis se basa en las ecuaciones de Newton para el movimiento en dos dimensiones, considerando:

- **Gravedad:** $g = 9{,}8\,\mathrm{m/s^2}$.
- **Velocidad inicial:** v_0.
- **Ángulo de lanzamiento:** α.
- **Coeficiente de arrastre:** b (solo en la versión con rozamiento).
- **Masa del proyectil:** m.

A continuación, se detallan los modelos físicos implementados.

3.2. Dinámica sin resistencia del aire

En el caso ideal, el proyectil solo está sometido a la gravedad y las ecuaciones del movimiento son:

$$a_x = 0, \quad a_y = -g.$$

Integrando estas ecuaciones en el tiempo, se obtienen las expresiones para la velocidad:

$$v_x(t) = v_0 \cos\alpha, \quad v_y(t) = v_0 \sin\alpha - gt.$$

Y para la posición:

$$x(t) = v_0 \cos\alpha \cdot t,$$

$$y(t) = h_0 + v_0 \sin\alpha \cdot t - \frac{1}{2} g t^2.$$

El impacto con el suelo ocurre cuando $y = 0$, lo que permite determinar el tiempo de vuelo resolviendo la ecuación cuadrática:

$$h_0 + v_0 \sin\alpha \cdot t - \frac{1}{2} g t^2 = 0.$$

Finalmente, el alcance del proyectil en este caso se obtiene evaluando $x(t)$ en el tiempo de impacto t_{impacto}.

3.3. Dinámica con resistencia aerodinámica

Si se incluye la resistencia del aire proporcional a la velocidad (*drag lineal*), la ecuación de movimiento es:

$$m\frac{d\vec{v}}{dt} = -b\vec{v} + m\vec{g}.$$

Separando en componentes:

$$m\frac{dv_x}{dt} = -bv_x, \quad m\frac{dv_y}{dt} = -mg - bv_y.$$

Esto define un sistema de ecuaciones diferenciales con solución:

$$v_x(t) = v_0 \cos\alpha \cdot e^{-\frac{b}{m}t},$$

$$v_y(t) = \left(v_0 \sin\alpha + \frac{mg}{b}\right) e^{-\frac{b}{m}t} - \frac{mg}{b}.$$

Integrando nuevamente para la posición:

$$x(t) = \frac{mv_0 \cos\alpha}{b}\left(1 - e^{-\frac{b}{m}t}\right),$$

$$y(t) = h_0 + \left(v_0 \sin\alpha + \frac{mg}{b}\right)\frac{m}{b}\left(1 - e^{-\frac{b}{m}t}\right) - \frac{mg}{b}t.$$

El tiempo de impacto t_{impacto} se obtiene numéricamente resolviendo $y(t) = 0$.

3.4. Energía Mecánica

Para cada instante, se calcula la energía mecánica total:

$$E_c = \frac{1}{2}m(v_x^2 + v_y^2),$$

$$E_p = mgy.$$

Por lo tanto, la energía total es:

$$E_{\text{tot}} = E_c + E_p.$$

Si $y \leq 0$, la energía potencial se toma como cero. Además, se introduce una versión sobredimensionada de la energía total:

$$E_{\text{sd}} = 1{,}15 \cdot E_{\text{tot}}.$$

que se utiliza en `TabPower` para compensar pérdidas de eficiencia en la transferencia de energía.

3.5. Método de Simulación

El programa integra las ecuaciones usando el método de Euler explícito con un paso de tiempo $\Delta t = 0{,}01$ s:

1. Se inicializan las condiciones $x_0, y_0, v_{x,0}, v_{y,0}$.
2. Se calculan las aceleraciones a_x, a_y.
3. Se actualizan las velocidades v_x, v_y.
4. Se actualizan las posiciones x, y.
5. Se repite hasta que $y \leq 0$ (impacto).

Este método es eficiente y permite visualizar el movimiento paso a paso, actualizando la posición del proyectil en un `Canvas`.

3.6. Visualización

El programa permite ver la trayectoria en un gráfico, y controlar el tiempo con un deslizador (*slider*) que actualiza la posición y la velocidad del proyectil en tiempo real.

3.7. Interacción con Otras Pestañas

Los resultados de la simulación se utilizan en:

- **TabPower**: La energía mecánica final se pasa a esta pestaña para calcular la capacidad y voltaje de los condensadores.
- **TabSearch**: Se usa para optimizar el ángulo de lanzamiento minimizando la velocidad inicial.

4. TabDrag

4.1. Objetivo

Permite estimar, para un proyectil de forma dada (*Prisma*, *Cilindro*, *Esfera*), el coeficiente b de un modelo de rozamiento lineal $F_{\text{drag}} = -\,b\,v$ a partir de la ecuación estándar $F_{\text{drag}} = \frac{1}{2}\,\rho\,C_D\,A\,v^2$.

4.2. Modelo principal

$$b \;=\; 0{,}5\,\rho\,C_D\,A,$$

donde:

- ρ: densidad del aire (aprox. 1.225 kg/m^3).
- C_D: coeficiente de forma (depende de la geometría).
- A: área frontal (por ejemplo, en un prisma se toma lado^2, en un cilindro $\pi\,r^2$).

4.3. Interfaz

Se selecciona la geometría, se introducen parámetros (radio, altura, etc.), y se pulsa *Calcular Coef. Rozamiento*, obteniendo b. Este valor puede asignarse a *TabSimulator* (por ejemplo, con `set_b_value`).

5. TabSearch

5.1. Objetivo

Encontrar el ángulo de lanzamiento (de 0 a 90 grados) que minimice la velocidad inicial para alcanzar cierta distancia X_{target}.

5.2. Algoritmo

1. Se barre θ de 0 a 90 grados.
2. Para cada θ, se hace una *búsqueda por bisección* en la velocidad inicial para que la distancia final (simulando las ecuaciones de movimiento) sea $\approx X_{\text{target}}$.
3. Se recoge la menor velocidad conseguida.

Luego se grafica (ángulo vs. velocidad) con `matplotlib` y se señala el mínimo.

5.3. Modelos de movimiento

- **Sin rozamiento**: $a_x = 0,\ a_y = -g$.
- **Con rozamiento lineal**: $m\,a_x = -b\,v_x,\ m\,a_y = -m\,g - b\,v_y$.

Se integra paso a paso en el tiempo (método Euler simple).

5.4. Cálculo de energía mecánica

Para cada instante se computan

$$E_c = \frac{1}{2}\,m\,(v_x^2 + v_y^2), \quad E_p = m\,g\,y \quad (\text{si } y \geq 0).$$

La suma $E_{\text{tot}} = E_c + E_p$ se presenta en la interfaz. Un método `get_energy_required()` devuelve dicha energía final, la cual se usa en *TabPower* para dimensionar la descarga de condensadores.

6. TabCoil

6.1. Objetivo

En esta pestaña, el usuario introduce los parámetros físicos de la bobina (número de espiras, radio interior, radio exterior, etc.) y el programa calcula:

- La **inductancia total** de la bobina.
- La **resistencia** aproximada del hilo.

6.2. Modelos principales

1. **Autoinductancia de una espira**:
$$L_{\text{loop}}(r) \approx \mu_0\, r \left[\ln\!\Big(\frac{8\,r}{a_{\text{eff}}}\Big) - 2\right].$$
2. **Mutua entre dos espiras coaxiales**: Se basa en integrales elípticas (K, E).
$$M_{ij} = \mu_0 \sqrt{r_i\, r_j}\, \frac{\big(2-k\big)\,K(k^2) - 2\,E(k^2)}{k}, \quad k^2 = \frac{4\,r_i\,r_j}{(r_i+r_j)^2 + z_{ij}^2}.$$
3. **Inductancia total**:
$$L_{\text{bobina}} = \sum_i L_{\text{espira},\,i} \;+\; 2\sum_{i<j} M_{ij}.$$
4. **Resistencia del hilo**:
$$R = \rho_{\text{Cu}}\, \frac{\ell_{\text{total}}}{A},$$
donde ℓ_{total} es la suma de longitudes de las espiras y A el área de la sección del conductor.

6.3. Interfaz

Se introducen parámetros en milímetros, etc. Al pulsar *Calcular*, se presenta la inductancia en microhenrios (μH) y la resistencia en ohmios (Ω). Además, se genera un dibujo 2D de la bobina (vista de corte) mediante `plot_coil_in_frame`.

7. TabPower

7.1. Objetivo

- **Dimensionar** los parámetros de cada etapa (capacidad C, tensión V_0) para suministrar la energía deseada al proyectil.
- Animar la secuencia de etapas, mostrando cómo el proyectil recorre n bobinas en un `Canvas`.

7.2. Modelos de descarga R–L–C

Cada bobina se ve como un inductor de inductancia $L(x)$ (variable con la posición) en serie con una resistencia R. El condensador (capacidad C, precargado a V_0) se descarga en un intervalo corto (p. ej., $\sim 10\,\text{ms}$). La ley de malla:

$$V_C(t) - R\,I(t) \;-\; \frac{\mathrm{d}}{\mathrm{d}t}[\,L(x(t))\,I(t)\,] \;=\; 0, \quad \frac{\mathrm{d}V_C}{\mathrm{d}t} = -\,\frac{I}{C}.$$

Se integra de forma numérica (método Euler).

7.3. Dimensionamiento por bisección

Se parte de una **energía objetivo** E_{target} proveniente de *TabSimulator*. Con un número de etapas n, cada etapa aporta aproximadamente $\frac{E_{\text{target}}}{n}$. Se hace una búsqueda sobre C o V_0 (fijando el otro) para que la energía consumida del condensador se iguale a esa fracción. De ese modo se obtiene la configuración del circuito (capacidad, tensión).

7.4. Animación de las etapas

Para ilustrar el paso del proyectil por las n bobinas, se:

- Dibuja cada bobina como un rectángulo (h_c de largo).
- Se crea una trayectoria (t, x) asumiendo, por ejemplo, velocidad constante o un tiempo fijo por etapa.
- Un *slider* en milisegundos permite moverse por la animación, desplazando un óvalo que representa el proyectil.

8. Conclusión

La aplicación se compone de las siguientes pestañas y flujos de información:

1. **geometry_viewer.py**: librería con funciones de dibujo para geometrías y bobinas.
2. **TabCoil**: diseña la bobina, calcula inductancia y resistencia.
3. **TabDrag**: obtiene el coeficiente de arrastre lineal b.
4. **TabPower**: dimensiona la alimentación (capacidad y tensión) en cada etapa, y visualiza la progresión del proyectil por n bobinas.
5. **TabSearch**: optimiza el ángulo de lanzamiento para un alcance fijo, mediante barrido y bisección.
6. **TabSimulator**: simula el movimiento del proyectil en 2D (tiro parabólico), con o sin rozamiento, y calcula la energía final.

De esta forma, cada parte del proyecto contribuye a un paso distinto del diseño y la simulación de la lanzadera electromagnética, estableciendo un flujo de datos entre ellas (por ejemplo, `TabCoil` alimenta `TabPower` con L y R; `TabSimulator` aporta la energía requerida a `TabPower`, etc.). Con ello se consigue una aplicación integral de cálculo y visualización.