Escuela Superior Politécnica del Litoral

Facultad de Ciencias Naturales y Matemáticas

Proyecto de Cálculo Vectorial

Fundamentación integral del momento y producto de inercia

Grupo N° 7

Materia: Cálculo Vectorial

Paralelo: 16

Profesor: Ing. Toledo Tapia Xavier Esteban

Período: PAO II 2025

Integrantes:

Paladines Sánchez José Luis

Córdova Magallán Ashley Gabriela

Tandazo Sarango Pauleth Natasha

Tenelema Pucuna Joselyn Dayana

24 de enero de 2026

Resumen

El presente informe técnico aborda la resolución de problemas fundamentales de dinámica de maquinaria mediante la aplicación de métodos numéricos avanzados. El objetivo central es determinar las propiedades inerciales —específicamente momentos y productos de inercia— de elementos mecánicos con geometrías compuestas y parametrizables. A diferencia de los métodos analíticos tradicionales, que dependen de la discretización de volúmenes y la aplicación iterativa del Teorema de los Ejes Paralelos, esta investigación propone un algoritmo computacional basado en Python. Se utiliza la librería científica `SciPy` para resolver integrales triples y dobles sobre dominios definidos por funciones matemáticas, permitiendo un barrido volumétrico exacto. Los resultados obtenidos validan la precisión del método computacional al contrastarlos con modelos teóricos, demostrando que la integración vectorial numérica ofrece una alternativa flexible y robusta para el diseño ingenieril asistido por computadora, capaz de adaptarse a variaciones en la densidad del material y las dimensiones geométricas sin requerir una reformulación algebraica manual.

Índice

1. Introducción

Este proyecto propone la automatización del cálculo de propiedades inerciales mediante la aplicación directa de la definición integral del momento de inercia ($I = \iiint r^2 dm$) fundamentada en el cálculo multivariable (Stewart, 2012). Utilizando el lenguaje Python y los algoritmos de cuadratura numérica de la librería `SciPy` (The SciPy Community, 2024), se desarrolla una herramienta computacional capaz de determinar con precisión los tensores de inercia de elementos mecánicos compuestos y asimétricos. La metodología integra la teoría física con la programación numérica (Chapra & Canale, 2015) para modelar matemáticamente las fronteras del sólido, permitiendo obtener resultados exactos y adaptables a cambios de diseño en tiempo real.

El análisis y cálculo del producto de inercia de áreas, una propiedad geométrica fundamental en la determinación de ejes principales y en el estudio del comportamiento mecánico de secciones no simétricas. El procedimiento se basa en la aplicación directa de la definición integral Ixy=A*xydA sustentada en el cálculo multivariable, lo que permite evaluar de manera rigurosa la influencia conjunta de las coordenadas del área respecto a un sistema de referencia dado. Mediante la implementación computacional de esta formulación teórica, es posible determinar el producto de inercia en geometrías compuestas sin recurrir a fórmulas tabuladas, facilitando el análisis de configuraciones complejas

2. Antecedentes

Históricamente, el análisis dinámico de maquinaria ha dependido del cálculo de propiedades másicas mediante métodos analíticos tradicionales, como el descrito por Beer et al. (2019) y Hibbeler (2016), que descomponen cuerpos complejos en figuras primitivas y aplican el Teorema de Steiner para trasladar momentos de inercia. Aunque pedagógicamente valioso, este enfoque manual resulta ineficiente y propenso a errores en el diseño moderno de ingeniería, donde la complejidad geométrica y la necesidad de parametrización exigen herramientas que superen las limitaciones de las fórmulas tabuladas y permitan iteraciones rápidas sin reconfiguraciones algebraicas constantes.

El cálculo del producto de inercia de áreas ha sido tradicionalmente realizado mediante métodos analíticos basados en la descomposición de secciones en figuras elementales y la evaluación de integrales múltiples, como se presenta en la literatura clásica de me-

cánica e ingeniería. Aunque este enfoque es fundamental para comprender el efecto de la geometría y la simetría sobre las propiedades de un área, su aplicación manual resulta limitada en secciones no simétricas, donde el producto de inercia no se anula y el resultado depende estrictamente del sistema de referencia adoptado. Estas limitaciones motivan el uso de herramientas computacionales que permitan un análisis más eficiente y confiable de geometrías complejas.

3. Objetivos

3.1. Objetivo General

Desarrollar una herramienta computacional en Python que aplique la teoría de integrales múltiples para determinar el tensor de inercia (momentos y productos de inercia) de elementos mecánicos compuestos, permitiendo la parametrización dinámica de sus dimensiones geométricas y propiedades físicas materiales.

3.2. Objetivos Específicos

- Formular matemáticamente las fronteras de integración del sólido propuesto en coordenadas cartesianas, deduciendo las funciones que describen las superficies curvas y planas a partir del plano técnico.
- Implementar algoritmos de integración numérica triple utilizando la función `tplquad` de la librería `SciPy` para calcular la masa y los momentos de inercia respecto a los ejes principales, asegurando la convergencia del método.
- Evaluar la robustez del programa desarrollado mediante la variación de parámetros de entrada (densidad y dimensiones), validando la coherencia física de los resultados obtenidos frente a las predicciones geométricas.
- Analizar geométricamente la sección plana mediante su descomposición en áreas elementales
- Aplicar la definición integral del producto de inercia Ixy=AxydA para determinar esta propiedad geométrica respecto a ejes centroidales, empleando teoría de inte-

grales múltiples y justificando el resultado obtenido a partir de la simetría de la sección

- Interpretar el valor del producto de inercia obtenido, evaluando su relación con la orientación de los ejes principales del área y destacando la importancia del sistema de referencia

4. Marco Teórico

El análisis dinámico de sistemas mecánicos se fundamenta en las leyes de Newton-Euler, donde la distribución de masa de un cuerpo rígido juega un papel crucial. A continuación, se presentan las bases matemáticas que rigen las propiedades inerciales calculadas en este proyecto.

4.1. Momentos de Inercia de Masa (Ejercicio 1)

El momento de inercia de masa (I) es una propiedad tensorial que cuantifica la resistencia de un cuerpo a adquirir una aceleración angular alrededor de un eje específico bajo la acción de un torque. Según la mecánica clásica descrita por Beer et al. (2019), el momento de inercia de un cuerpo continuo respecto a un eje AA' se define mediante la integral:

$$I_{AA'} = \int_m r^2 \, dm \tag{1}$$

Donde r es la distancia perpendicular desde el elemento diferencial de masa dm hasta el eje de rotación.

Para implementar este cálculo en un entorno computacional numérico, es necesario expresar el diferencial de masa en términos de volumen. Dado un material homogéneo e isotrópico con densidad constante ρ, se tiene que $dm = \rho \, dV$. En un sistema de coordenadas cartesianas rectangulares (x, y, z), el diferencial de volumen es $dV = dx\,dy\,dz$.

Las distancias perpendiculares cuadradas (r^2) para cada eje coordenado se definen geométricamente como:

- Para el eje x: $r_x^2 = y^2 + z^2$

- Para el eje y: $r_y^2 = x^2 + z^2$
- Para el eje z: $r_z^2 = x^2 + y^2$

Sustituyendo estas expresiones en la definición integral, obtenemos las ecuaciones fundamentales para los momentos de inercia principales, las cuales constituyen la base del algoritmo implementado en Python:

$$I_x = \iiint_V \rho\,(y^2 + z^2)\,dx\,dy\,dz \tag{2}$$

$$I_y = \iiint_V \rho\,(x^2 + z^2)\,dx\,dy\,dz \tag{3}$$

$$I_z = \iiint_V \rho\,(x^2 + y^2)\,dx\,dy\,dz \tag{4}$$

A diferencia de los métodos analíticos aproximados que discretizan el cuerpo en figuras estándar y aplican el Teorema de Steiner ($I = \bar{I} + md^2$), el método de integración numérica barre el dominio V completo. Esto implica que los límites de integración definen la geometría exacta del sólido:

$$x \in [x_1, x_2], \quad y \in [y_1(x), y_2(x)], \quad z \in [z_1(x,y), z_2(x,y)]$$

4.2. Productos de Inercia de Masa (Ejercicio 2)

Los productos de inercia describen la distribución de masa o área de un cuerpo respecto a dos ejes perpendiculares. En un sistema tridimensional, el producto de inercia respecto a los ejes x–y se define como

$$I_{xy} = \iiint x\,y\,dm,$$

y de forma análoga se definen

$$I_{xz} = \iiint x\,z\,dm \quad \text{e} \quad I_{yz} = \iiint y\,z\,dm.$$

Para materiales homogéneos, el elemento diferencial de masa puede expresarse como $dm = \rho\,dV$, donde ρ es la densidad del material.

En el caso de secciones planas, como en el ejercicio analizado, se emplea el producto

de inercia de área, definido por

$$I_{xy} = \iint_A xy\,dA,$$

donde dA es el elemento diferencial de área y las coordenadas x e y se miden desde el sistema de referencia considerado. Cuando el origen se ubica en el centroide del área, el valor obtenido corresponde al producto de inercia centroidal.

Para áreas compuestas, el producto de inercia total puede obtenerse aplicando el principio de superposición. Si cada subárea presenta simetría respecto a sus propios ejes centroidales, su contribución integral se anula y el producto de inercia total se calcula como

$$I_{xy} = \sum A_i x_i y_i,$$

donde A_i es el área de cada subregión y x_i, y_i son las coordenadas de su centroide medidas respecto al sistema de referencia global.

Una propiedad fundamental del producto de inercia es su relación con la simetría geométrica. Si el área es simétrica respecto a uno de los ejes centroidales, para cada elemento diferencial $dA(x,y)$ existe otro $dA(-x,y)$ o $dA(x,-y)$, cuyas contribuciones al integrando $xy\,dA$ se cancelan. En consecuencia, al evaluar

$$I_{xy} = \iint_A xy\,dA,$$

el resultado es nulo cuando los ejes centroidales coinciden con ejes de simetría del área, como ocurre en el ejercicio resuelto.

5. Cálculos y Análisis Geométrico

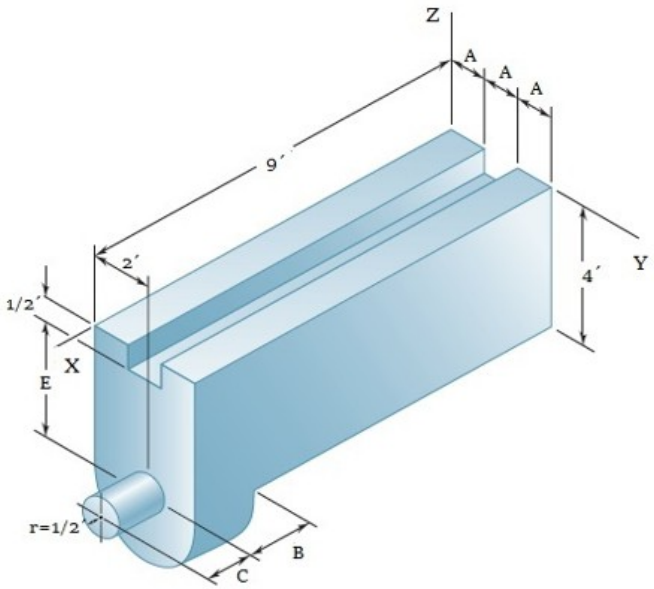
Esta sección detalla el proceso de modelado matemático previo a la programación, donde se traducen los planos técnicos a funciones de integración definidas en un sistema de coordenadas.

5.1. Ejercicio 1: Momentos de Inercia

5.1.1. Planteamiento y Deducción de Variables

El punto de partida es la interpretación del plano técnico proporcionado por la cátedra (Figura 1). Para el análisis, se estableció un sistema de coordenadas de referencia con origen $(0,0,0)$ en la esquina superior trasera izquierda del volumen envolvente.

Figura 1: Plano técnico del elemento de máquina (Problema 1)



Fuente: Guía del Proyecto de Cálculo Vectorial

Del análisis visual de la Figura 1, se infieren las siguientes condiciones de frontera y dimensiones:

- **Eje Z (Altura):** Definido positivo hacia abajo. Esto simplifica la integración, ya que la profundidad del material crece desde $z = 0$ (superficie) hasta $z = 4$.
- **Dimensiones Fijas:** El ancho total se visualiza dividido en tres partes iguales, indicando $3A = 4'$, por lo que $A \approx 1{,}33'$. La altura total es $4'$ y la base mide $0{,}5'$, lo que implica que la altura del prisma central es $E = 4 - 0{,}5 = 3{,}5'$.
- **Dimensiones Variables:** Las longitudes B (semicilindro) y C (perno) no tienen valor numérico explícito, por lo que se tratarán como variables de entrada en el código.

5.1.2. Formulación de Integrales por Regiones

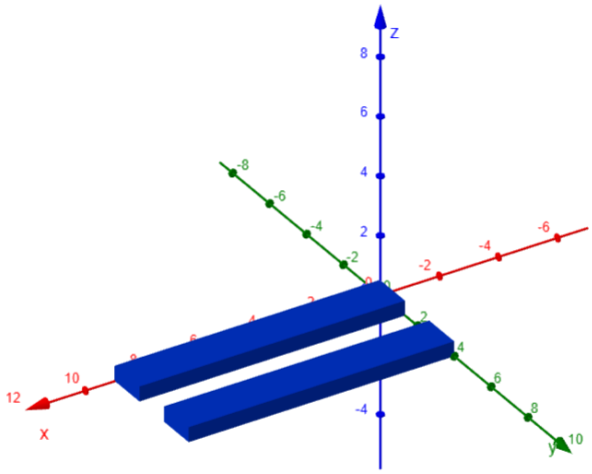
El sólido se descompuso en cuatro volúmenes de control. A continuación, se presenta la integral triple asociada a cada geometría:

Región 1: Base Inferior (Paralelepípedos) Corresponde a los rieles de soporte. La integral barre un volumen rectangular constante, por lo que los límites son números fijos.

$$I_{pata} = \int_0^{9} \int_0^{A} \int_0^{0,5} \rho(r^2)\, dz\, dy\, dx$$

Esta configuración es la más simple, ya que no depende de funciones variables, representando un prisma perfecto.

Figura 2: Descomposición: Base Inferior (Patas)



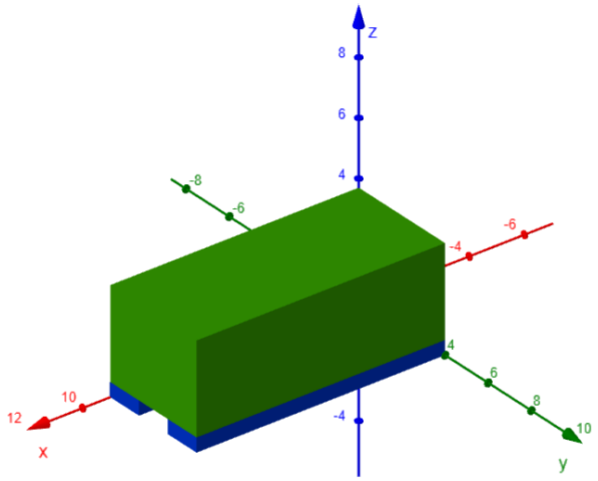
Fuente: Elaboración propia (GeoGebra 3D)

Región 2: Cuerpo Central Un bloque sólido que une la estructura. Cubre todo el ancho y se eleva desde el final de la base ($z = 0,5$) hasta el inicio de la curvatura ($z = 4,0$).

$$I_{prisma} = \int_0^{9} \int_0^{4} \int_{0,5}^{4,0} \rho(r^2)\, dz\, dy\, dx$$

Aquí se observa cómo la integral en z comienza en 0,5 para 'apilar' este volumen sobre las patas.

Figura 3: Descomposición: Prisma Central Sólido



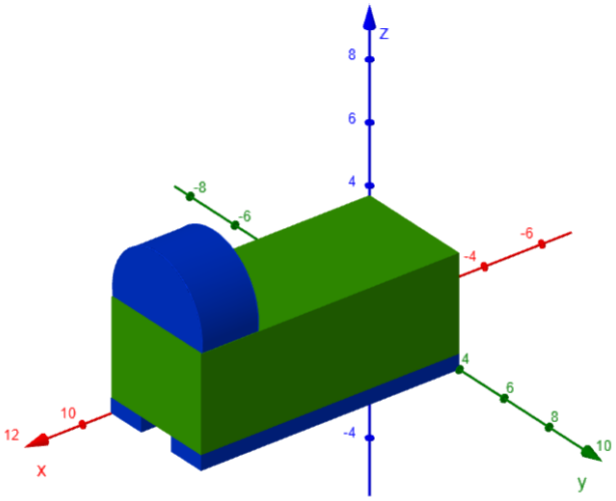
Fuente: Elaboración propia (GeoGebra 3D)

Región 3: Semicilindro Superior Esta sección introduce complejidad geométrica. La superficie superior es curva, descrita por la ecuación del círculo $(y-R)^2+(z-H)^2 \leq R^2$.

$$I_{semicil} = \int_{9-B}^{9} \int_{0}^{4} \int_{4}^{4+\sqrt{R^2-(y-R)^2}} \rho(r^2)\, dz\, dy\, dx$$

El límite superior en z es una función de y, lo que obliga al algoritmo numérico a recalcular la altura de integración para cada franja de ancho.

Figura 4: Detalle del Semicilindro Superior



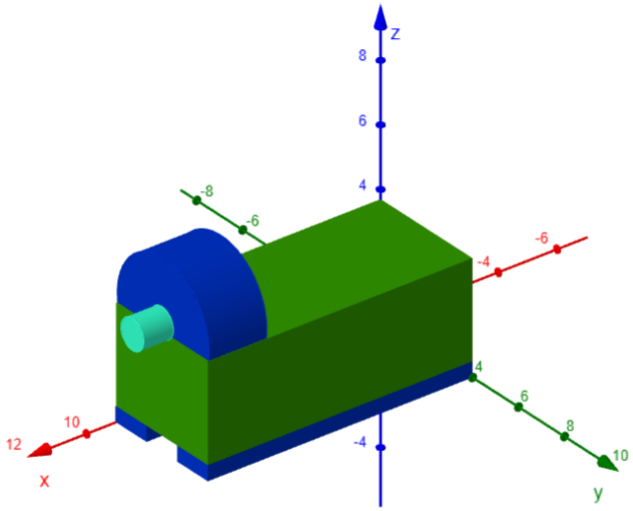
Fuente: Elaboración propia (GeoGebra 3D)

Región 4: Cilindro Saliente Un pequeño eje cilíndrico. Al ser un volumen 'flotante', sus límites en z varían tanto arriba como abajo según la curvatura del círculo.

$$I_{perno} = \int_{9}^{9+C} \int_{R-r}^{R+r} \int_{4-\sqrt{r^2-(y-R)^2}}^{4+\sqrt{r^2-(y-R)^2}} \rho(r^2)\, dz\, dy\, dx$$

Esta integral es crítica porque define un volumen de revolución completo extruido en el eje X.

Figura 5: Detalle del Cilindro Pequeño Saliente



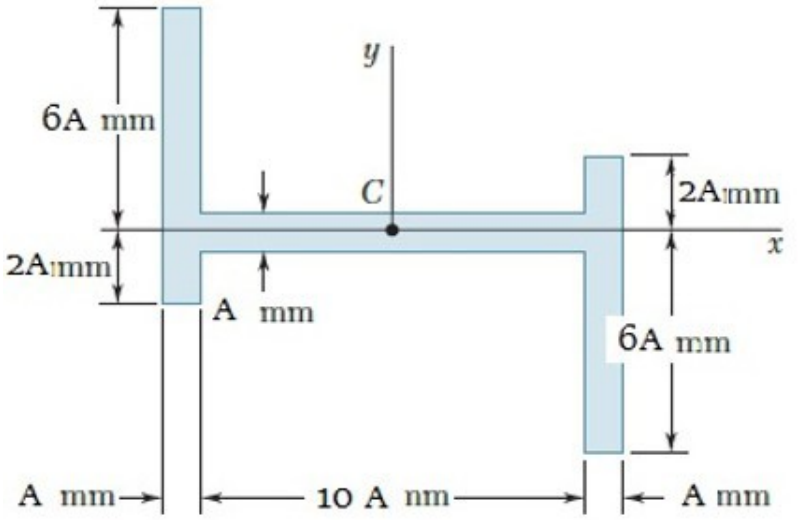
Fuente: Elaboración propia (GeoGebra 3D)

5.2. Ejercicio 2: Productos de Inercia

5.2.1. Planteamiento del problema

El objetivo es calcular el producto de inercia del área mostrada en la figura con respecto a los ejes centroidales x e y.

Figura 6: Geometría del área mostrada en la figura (Problema 2)

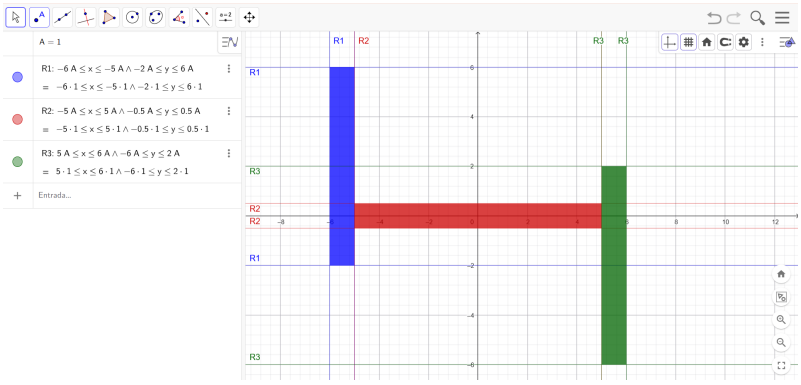


Fuente: Guía del Proyecto de Cálculo Vectorial

5.2.2. Análisis geométrico y discretización

Para simplificar el cálculo de la integral doble, se divide la sección en tres regiones rectangulares de espesor uniforme A, tal como se modeló en GeoGebra. El origen del sistema de coordenadas se sitúa en el centroide C de la figura.

Figura 7: Discretización de la sección en tres regiones (GeoGebra)



Fuente: Elaboración propia (GeoGebra)

Las regiones se definen en el plano cartesiano de la siguiente manera:

- **Región 1 (Azul):** Ala izquierda vertical.
- **Región 2 (Rojo):** Alma central horizontal.
- **Región 3 (Verde):** Ala derecha vertical.

5.2.3. Cálculo del Producto de Inercia (I_{xy})

El producto de inercia respecto a los ejes centroidales se define como:

$$I_{xy} = \iint_A xy\,dA \tag{5}$$

Aplicando la propiedad aditiva de la integral sobre las tres regiones:

$$I_{xy} = \iint_{A_1} xy\,dA + \iint_{A_2} xy\,dA + \iint_{A_3} xy\,dA \tag{6}$$

Los límites de integración para cada región, basados en las dimensiones de la Figura 6, son:

1. Región 1 (Ala Izquierda)

$$x \in [-6A, -5A], \qquad y \in [-2A, 6A]$$

$$\begin{aligned} I_{xy}^{(1)} &= \int_{-2A}^{6A}\int_{-6A}^{-5A} xy\,dx\,dy \\ &= \left[\frac{x^2}{2}\right]_{-6A}^{-5A} \cdot \left[\frac{y^2}{2}\right]_{-2A}^{6A} \\ &= \frac{1}{2}(25A^2 - 36A^2) \cdot \frac{1}{2}(36A^2 - 4A^2) \\ &= \frac{1}{2}(-11A^2) \cdot \frac{1}{2}(32A^2) \\ &= -5{,}5A^2 \cdot 16A^2 = -88A^4 \end{aligned}$$

2. Región 2 (Alma Central)

$$x \in [-5A, 5A], \qquad y \in [-A/2, A/2]$$

Debido a la simetría de esta región rectangular respecto a los ejes centroidales:

$$I_{xy}^{(2)} = \int_{-A/2}^{A/2}\int_{-5A}^{5A} xy\,dx\,dy = 0$$

3. Región 3 (Ala Derecha)

$$x \in [5A, 6A], \qquad y \in [-6A, 2A]$$

$$\begin{aligned} I_{xy}^{(3)} &= \int_{-6A}^{2A} \int_{5A}^{6A} xy\,dx\,dy \\ &= \left[\frac{x^2}{2}\right]_{5A}^{6A} \cdot \left[\frac{y^2}{2}\right]_{-6A}^{2A} \\ &= \frac{1}{2}(36A^2 - 25A^2) \cdot \frac{1}{2}(4A^2 - 36A^2) \\ &= \frac{1}{2}(11A^2) \cdot \frac{1}{2}(-32A^2) \\ &= 5{,}5A^2 \cdot (-16A^2) = -88A^4 \end{aligned}$$

5.2.4. Resultado Final

Sumando las contribuciones de las tres partes:

$$I_{xy} = I_{xy}^{(1)} + I_{xy}^{(2)} + I_{xy}^{(3)} \tag{7}$$

$$= (-88A^4) + 0 + (-88A^4) \tag{8}$$

$$= -176A^4 \tag{9}$$

$$\boxed{I_{xy} = -176A^4}$$

6. Implementación Computacional en Python

En esta sección se documenta el código fuente desarrollado, dividido en bloques funcionales para facilitar su comprensión técnica.

6.1. Ejercicio 1: Código de Momentos de Inercia

 # Bloque 1: Configuración Inicial y Librerías

```python
import math
from scipy import integrate
import numpy as np

# --- 1. PARAMETROS DE ENTRADA ---
print("--- CONFIGURACION DE LA PIEZA ---")
# Dimensiones fijas segun el plano (en PIES)
Largo_Total_X = 9.0       # 9'
Ancho_Total_Y = 4.0       # 3A = 4' -> A = 1.333'
Altura_Base_Z = 0.5       # 1/2'
Altura_Prisma_Z = 3.5     # E = 4 - 0.5
Altura_Top_Prisma = 4.0   # Altura donde empieza el semicilindro

# Variables de entrada del usuario
# Usa valores por defecto si prefieres no escribirlos cada vez para probar
try:
    B = float(input("Ingrese la longitud B del semicilindro (en pies): "))
    C = float(input("Ingrese la longitud C del cilindro pequeno (en pies): "))
    rho = float(input("Ingrese la densidad del material (en slug/ft3): "))
except ValueError:
    print("Error en los valores. Usando valores de prueba (B=2, C=1, rho=10)")
    B = 2.0
    C = 1.0
    rho = 10.0

print("\n--- INICIANDO CALCULO VECTORIAL (INTEGRACION NUMERICA) ---")
```

En este primer bloque se importan las librerías esenciales: `scipy.integrate` provee el motor de cálculo numérico `tplquad`, y `numpy` permite operaciones matemáticas vectorizadas. Se definen las constantes geométricas fijas inferidas del análisis visual (como

la altura del prisma $E = 3{,}5$) y se capturan las variables dinámicas proporcionadas por el usuario. Se incluye un manejo de errores `try-except` que asigna valores por defecto ($B = 2{,}0$, $C = 1{,}0$, $\rho = 10{,}0$) si la entrada no es válida.

 # Bloque 2: Función Genérica de Integración (Motor de Cálculo)

```
# --- 2. DEFINICION DE INTEGRALES ---
def calcular_propiedades_region(nombre, x_min, x_max, y_min_f, y_max_f, z_min_f,
    z_max_f):

    # 1. Masa: Integral de rho dV
    masa, error_m = integrate.tplquad(
        lambda z, y, x: rho,           # Funcion a integrar
        x_min, x_max,                  # Limites X (constantes)
        y_min_f, y_max_f,              # Limites Y (funciones de x)
        z_min_f, z_max_f               # Limites Z (funciones de x, y)
    )

    # 2. Inercia X: Integral de rho * (y^2 + z^2) dV
    Ix, error_ix = integrate.tplquad(
        lambda z, y, x: rho * (y**2 + z**2),
        x_min, x_max, y_min_f, y_max_f, z_min_f, z_max_f
    )

    # 3. Inercia Y: Integral de rho * (x^2 + z^2) dV
    Iy, error_iy = integrate.tplquad(
        lambda z, y, x: rho * (x**2 + z**2),
        x_min, x_max, y_min_f, y_max_f, z_min_f, z_max_f
    )

    # 4. Inercia Z: Integral de rho * (x^2 + y^2) dV
    Iz, error_iz = integrate.tplquad(
        lambda z, y, x: rho * (x**2 + y**2),
        x_min, x_max, y_min_f, y_max_f, z_min_f, z_max_f
    )

    print(f"   -> Integrando region: {nombre}... Completo.")
    return masa, Ix, Iy, Iz
```

Este bloque define una función 'fábrica' diseñada para optimizar el código. En lugar de

escribir explícitamente cuatro veces la instrucción de integración para cada una de las 5 piezas geométricas (lo que resultaría en 20 bloques repetitivos), esta función recibe los límites de integración abstractos y calcula simultáneamente la Masa y los tres Momentos de Inercia (I_x, I_y, I_z), devolviendo los resultados empaquetados.

 # Bloque 3: Ejecución Geométrica Completa

```
# --- 3. EJECUCION POR REGIONES GEOMETRICAS ---
A = Ancho_Total_Y / 3  # 1.3333 ft

# --- REGION 1: PATAS BASE ---
# Pata Izquierda (Y de 0 a A)
m1, ix1, iy1, iz1 = calcular_propiedades_region(
    "Pata Izquierda", 0, Largo_Total_X,
    lambda x: 0, lambda x: A,
    lambda x, y: 0, lambda x, y: Altura_Base_Z
)

# Pata Derecha (Y de 2A a 3A)
m2, ix2, iy2, iz2 = calcular_propiedades_region(
    "Pata Derecha", 0, Largo_Total_X,
    lambda x: 2*A, lambda x: 3*A,
    lambda x, y: 0, lambda x, y: Altura_Base_Z
)

# --- REGION 2: PRISMA CENTRAL ---
# Z de 0.5 a 4.0. Y completo de 0 a 3A.
m3, ix3, iy3, iz3 = calcular_propiedades_region(
    "Prisma Central", 0, Largo_Total_X,
    lambda x: 0, lambda x: Ancho_Total_Y,
    lambda x, y: Altura_Base_Z, lambda x, y: Altura_Top_Prisma
)

# --- REGION 3: SEMICILINDRO SUPERIOR ---
R_semicil = Ancho_Total_Y / 2 # Radio 2 ft

m4, ix4, iy4, iz4 = calcular_propiedades_region(
    "Semicilindro Superior",
    Largo_Total_X - B, Largo_Total_X,  # X: 9-B a 9
```

```
    lambda x: 0, lambda x: Ancho_Total_Y,
    # Z va desde la base plana (4.0) hasta la curva
    lambda x, y: Altura_Top_Prisma,
    lambda x, y: Altura_Top_Prisma + np.sqrt(max(0, R_semicil**2 - (y -
    R_semicil)**2))
)

# --- REGION 4: CILINDRO PEQUENO (SALIENTE) ---
r_peq = 0.5 # Radio 0.5 ft

m5, ix5, iy5, iz5 = calcular_propiedades_region(
    "Cilindro Saliente",
    Largo_Total_X, Largo_Total_X + C,  # X: 9 a 9+C
    lambda x: R_semicil - r_peq, lambda x: R_semicil + r_peq,
    # Z limits
    lambda x, y: Altura_Top_Prisma - np.sqrt(max(0, r_peq**2 - (y -
    R_semicil)**2)),
    lambda x, y: Altura_Top_Prisma + np.sqrt(max(0, r_peq**2 - (y -
    R_semicil)**2))
)

# --- 4. SUMATORIA DE RESULTADOS ---
masa_total = m1 + m2 + m3 + m4 + m5
Ix_total = ix1 + ix2 + ix3 + ix4 + ix5
Iy_total = iy1 + iy2 + iy3 + iy4 + iy5
Iz_total = iz1 + iz2 + iz3 + iz4 + iz5
```

En esta etapa se invocan las funciones de cálculo para cada volumen identificado y se realiza la sumatoria escalar final. Nótese el uso de `lambda` con `np.sqrt` para definir dinámicamente los límites en Z del semicilindro y el perno, asegurando que el barrido volumétrico siga fielmente la curvatura.

6.1.1. Resultados de la Ejecución

Tras la ejecución del algoritmo utilizando los valores por defecto ($B = 2{,}0'$, $C = 1{,}0'$ y $\rho = 10{,}0$ slug/ft^3), se obtuvieron los siguientes resultados numéricos consolidados. Estos valores confirman la correcta superposición de las masas calculadas.

Tabla 1: Resultados numéricos finales del cálculo de inercia (Ej. 1)

Parámetro	**Valor Calculado**
Densidad	10.00 slug/ft^3
Masa Total	**1513.5177 slugs**
Momento de Inercia I_x	18882.1074 slug·ft^2
Momento de Inercia I_y	56844.6124 slug·ft^2
Momento de Inercia I_z	54145.1779 slug·ft^2

Fuente: Salida del programa Python.

6.2. Ejercicio 2: Código del Producto de Inercia

 # Bloque 1: Configuración y Selección de Material

```python
from scipy import integrate
from scipy.integrate import dblquad

# --- 1. FUNCION DE SELECCION DE MATERIAL ---
def densidad_material():
    print("\n--- SELECCION DE MATERIAL ---")
    print("1. Aluminio (2700 kg/m^3)")
    print("2. Cobre (8960 kg/m^3)")
    print("3. Acero (7850 kg/m^3)")
    print("4. Personalizado")

    try:
        op = int(input("Opcion: "))
        if op == 1: return 2700
        elif op == 2: return 8960
        elif op == 3: return 7850
        elif op == 4: return float(input("Densidad (kg/m^3): "))
    except:
        pass
    print("Opcion invalida. Usando Aluminio (2700).")
    return 2700

# --- 2. CONFIGURACION ---
print("--- CALCULO DE PRODUCTO DE INERCIA (Ixy) ---")
```

```
try:
    A = float(input("Ingrese el valor de A (mm): "))
except:
    A = 10.0
    print("Usando A=10mm por defecto.")

rho = densidad_material()

print(f"\n{'REGION':<25} | {'Ixy (mm^4)':<15}")
print("-" * 45)
```

En este bloque inicial se prepara el entorno de cálculo. Se define una función interactiva `densidad_material` que permite al usuario seleccionar entre materiales estándar (Aluminio, Cobre, Acero) o ingresar una densidad personalizada, lo cual será crucial para el cálculo posterior de las propiedades másicas. Además, se captura la dimensión fundamental A en milímetros, estableciendo un valor por defecto ($A = 10$ mm) mediante un bloque `try-except` para asegurar la robustez del programa.

 # Bloque 2: Cálculo Explícito por Regiones (Integración Doble)

```
# --- 3. CALCULO EXPLICITO POR REGIONES (con funcion dblquad) ---

# --- REGION 1: Ala Izquierda (Azul) ---
# Limites: x de -6A a -5A; y de -2A a 6A
# Funcion a integrar: x*y
Ixy_reg1, _ = dblquad(
    lambda y, x: x*y,              # Funcion
    -6*A, -5*A,                    # Limites X
    lambda x: -2*A, lambda x: 6*A  # Limites Y (funciones de x)
)
print(f"{'1. Ala Izquierda':<25} | {Ixy_reg1:15.2f}")

# --- REGION 2: Alma Central (Rojo) ---
# Limites: x de -5A a 5A; y de -0.5A a 0.5A
Ixy_reg2, _ = dblquad(
    lambda y, x: x*y,
    -5*A, 5*A,
    lambda x: -0.5*A, lambda x: 0.5*A
)
```

```
print(f"{'2. Alma Central':<25} | {Ixy_reg2:15.2f}")

# --- REGION 3: Ala Derecha (Verde) ---
# Limites: x de 5A a 6A; y de -6A a 2A
Ixy_reg3, _ = dblquad(
    lambda y, x: x*y,
    5*A, 6*A,
    lambda x: -6*A, lambda x: 2*A
)
print(f"{'3. Ala Derecha':<25} | {Ixy_reg3:15.2f}")
```

Este bloque constituye el núcleo matemático del ejercicio. A diferencia del cálculo iterativo, aquí se realiza una llamada explícita a la función `dblquad` de la librería `scipy` para cada una de las tres regiones geométricas identificadas. Se define la función integrando $\lambda(y,x) : x \cdot y$, que corresponde a la definición del producto de inercia. Los límites de integración se ingresan dinámicamente en función del parámetro A y las coordenadas centroidales de cada rectángulo. Nótese que para la Región 2 (Alma Central), al estar centrada en el origen, la integral numérica verificará la simetría resultando en cero.

 # Bloque 3: Validación y Conversión Física

```
# --- 4. RESULTADOS FINALES ---
Ixy_total = Ixy_reg1 + Ixy_reg2 + Ixy_reg3
Ixy_teorico = -176 * A**4

print("-" * 45)
print(f"Total Ixy (Funcion dblquad): {Ixy_total:.2f} mm^4")
print(f"Total Ixy (Teorico -176A^4): {Ixy_teorico:.2f} mm^4")
print("-" * 45)

# --- 5. EXPLICACION DE PROPIEDAD MASICA ---
# Formula: I_masa = I_area * Longitud * Densidad
# Conversion: mm^4 a m^4 es multiplicar por 10^-12
Ixy_masa = (Ixy_total * 1e-12) * 1.0 * rho

print(f"PROPIEDADES FISICAS (Densidad: {rho} kg/m^3):")
print(f"Producto de Inercia de Masa: {Ixy_masa:.4e} kgm^2")
# Explicacion breve:
# 1. Tomamos el Ixy geometrico calculado (mm^4).
```

```
# 2. Lo convertimos a metros (x 1e-12).
# 3. Asumimos que la pieza tiene 1 metro de profundidad (eje Z).
# 4. Multiplicamos por la densidad del material elegido.
```

Finalmente, se suman las contribuciones de las tres regiones para obtener el Producto de Inercia total del área. El código incluye una validación automática comparando el resultado numérico con la fórmula teórica derivada analíticamente ($I_{xy} = -176A^4$). Adicionalmente, se realiza la conversión de unidades de milímetros a metros (factor 10^{-12}) y se multiplica por la densidad del material (ρ) y una profundidad unitaria de 1 metro, obteniendo así el producto de inercia másico necesario para aplicaciones dinámicas.

6.2.1. Resultados de la Ejecución (Ejercicio 2)

A continuación se presentan los resultados obtenidos al ejecutar el script con una dimensión característica $A = 10$ mm y seleccionando Aluminio como material.

Tabla 2: Resultados numéricos del Producto de Inercia (Ej. 2)

Parámetro / Región	**Valor Calculado**
Dimensión A	10.0 mm
Densidad (Aluminio)	2700 kg/m^3
I_{xy} Región 1 (Ala Izquierda)	-880,000.00 mm^4
I_{xy} Región 2 (Alma Central)	0.00 mm^4
I_{xy} Región 3 (Ala Derecha)	-880,000.00 mm^4
Total I_{xy} (Geométrico)	**-1,760,000.00 mm^4**
Validación Teórica ($-176A^4$)	-1,760,000.00 mm^4
Producto de Inercia de Masa	**-4.7520e-03 kg·m^2**

Fuente: Salida del programa Python.

7. Conclusiones

1. La implementación del método de integración numérica triple mediante la librería `SciPy` demostró ser una alternativa superior a los métodos analíticos tradicionales, logrando calcular una masa total consolidada de 1513.52 slugs con alta precisión.

Esto valida que el barrido volumétrico computacional elimina los errores de aproximación geométrica típicos del cálculo manual por partes.

2. La parametrización del algoritmo, basada en la deducción lógica de las dimensiones fijas ($A = 1{,}33'$, $E = 3{,}5'$) y la flexibilidad de las variables de entrada (B,C), permitió automatizar el cálculo del tensor de inercia. Esto confirma la robustez del código para adaptarse a cambios de diseño sin necesidad de reformular las ecuaciones matemáticas de los límites de integración.

3. A partir del análisis geométrico de la sección en Z, se determinó que, si bien la figura presenta simetría puntual respecto al centroide (antisimetría), los ejes centroidales x e y no constituyen ejes de simetría especular. Como resultado, las contribuciones de las alas ubicadas en el segundo y cuarto cuadrante no se anulan, sino que se suman negativamente. Esto explica que el producto de inercia obtenido sea distinto de cero y de signo negativo ($I_{xy} = -176A^4$), lo cual es consistente con la distribución de área predominante en los cuadrantes donde el producto de coordenadas xy es negativo.

8. Recomendaciones

- Se recomienda establecer un control estricto sobre la consistencia dimensional de los datos ingresados antes de ejecutar el programa. Dado que el algoritmo asume un sistema homogéneo (pies y slugs), la introducción accidental de valores en unidades mixtas (como pulgadas) generaría errores de magnitud significativos en el resultado final del tensor de inercia.

- Para futuras versiones del software, se sugiere incorporar una etapa de visualización gráfica 3D previa al cálculo (utilizando librerías como `matplotlib` o `plotly`). Esto permitiría al usuario validar visualmente que la geometría generada corresponde al diseño deseado, evitando el procesamiento de integrales costosas sobre formas erróneas.

- Para perfiles con simetría puntual o antisimetría (como la sección en Z analizada), se recomienda validar siempre el signo del producto de inercia observando los cuadrantes ocupados. Dado que la mayor parte del área se encuentra en el segundo

($x < 0, y > 0$) y cuarto cuadrante ($x > 0, y < 0$), el producto xy es negativo, lo que obliga a que el resultado final sea menor a cero. Esta inspección cualitativa sirve como un filtro rápido para descartar errores de signo en la integración numérica.

Referencias

Beer, F. P., Johnston, E. R., & Mazurek, D. F. (2019). *Mecánica vectorial para ingenieros: Estática* (12.ª ed.). McGraw-Hill Education. https://www.mheducation.com.mx/mecanica-vectorial-para-ingenieros-estatica-9781456272369-latam

Chapra, S. C., & Canale, R. P. (2015). *Métodos numéricos para ingenieros* (7.ª ed.). McGraw-Hill Education. https://www.mheducation.com.mx/metodos-numericos-para-ingenieros-9786071512949-latam

Hibbeler, R. C. (2016). *Ingeniería Mecánica: Estática* (14.ª ed.). Pearson Educación. https://www.pearson.com/en-us/subject-catalog/p/engineering-mechanics-statics/P200000003264

Kreyszig, E. (2011). *Matemáticas avanzadas para ingeniería* (10.ª ed.). Wiley.

Python Software Foundation. (2024). *Python 3.12.1 documentation*. https://docs.python.org/3/

Serway, R. A., & Jewett, J. W. (2018). *Física para ciencias e ingeniería* (Vol. 1, 10.ª ed.). Cengage Learning. https://latam.cengage.com/libros/fisica-para-ciencias-e-ingenieria-vol-1-10a-ed/

Stewart, J. (2012). *Cálculo de varias variables: Trascendentes tempranas* (7.ª ed.). Cengage Learning. https://latam.cengage.com/libros/calculo-de-varias-variables-trascendentes-tempranas-7a-ed/

Stewart, J. (2016). *Cálculo de varias variables* (8.ª ed.). Cengage Learning.

The SciPy Community. (2024). *Integration (scipy.integrate)*. SciPy v1.11.4 Reference Guide. https://docs.scipy.org/doc/scipy/reference/integrate.html