ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

METODOS NUMÉRICOS (ICCD412)

PROYECTO TELESCOPIO JAMES WEBB

Integrantes:

- Mateo Sebastián Cumbal Guasgua
- Daniel Ismael Flores Espín
- Johann Vladimir Pasquel Montenegro
- Luis Rafael Tipán Tandalla

DOCENTE: Ing. Jonathan A. Zea

ESCUELA POLITÉCNICA NACIONAL

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN 3
2. OBJETIVOS 4
3. METEDOLOGÍA 4
3.1. Descripción de la solución. 4
3.2. Desarrollo matemático. 6
3.3. Diagrama de flujo. 10
4. RESULTADOS Y CONCLUSIONES: 21

Índice de Tablas

Figura 1.: Modelo de pistones Telescopio. 3
Figura 2. Diagrama de Flujo 12
Figura 3. Librerías. 13
Figura 4.Clase Aplicación Telescopio 14
Figura 5. Método crear_widgets. 15
Figura 6. Método Calcular_y_Dibujar 1. 16
Figura 7. Método Calcular_y_Dibujar 2. 17
Figura 8.Método Calcular_y_Dibujar 3. 18
Figura 9. Método Dibujar_sistema. 19
Figura 10. Caso por defecto. 21
Figura 11. Caso Extremo.. 22

ESCUELA POLITÉCNICA NACIONAL

1. INTRODUCCIÓN

El telescopio espacial James Webb representa uno de los avances tecnológicos más significativos en la historia de la exploración del universo, marcando un hito en nuestra capacidad para comprender los orígenes del cosmos y los procesos que lo rigen. Este instrumento, diseñado para observar las profundidades del universo en longitudes de onda infrarrojas, combina ingeniería de vanguardia con una arquitectura óptica altamente sofisticada. Su sistema óptico incluye un espejo primario segmentado de 6.5 metros de diámetro y un espejo secundario ajustable, cuya precisión milimétrica es fundamental para garantizar imágenes claras y detalladas de galaxias, exoplanetas y otros fenómenos celestes.

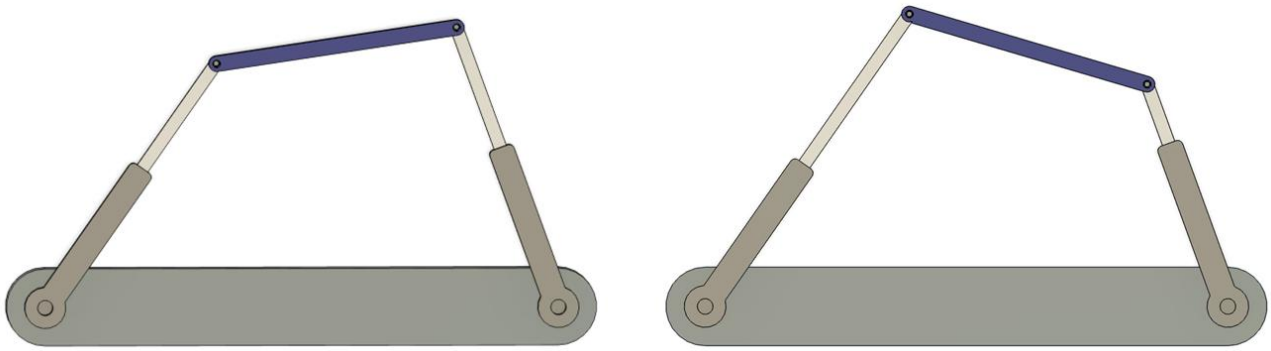


Figura 1.: Modelo de pistones Telescopio.

El espejo secundario, pieza clave en el sistema, emplea pistones lineales controlados con extrema exactitud para orientarse y mantener el alineamiento necesario en condiciones extremas del espacio profundo. Inspirados en este fascinante mecanismo, el presente proyecto propone la implementación de un sistema simplificado que modela el comportamiento de un espejo único controlado mediante pistones. Este sistema busca replicar, a pequeña escala, las complejas operaciones de orientación y ajuste, destacando la importancia de la precisión y la lógica computacional en la exploración espacial. A través de simulaciones y cálculos geométricos avanzados, este proyecto no solo rinde homenaje a la tecnología detrás del James Webb, sino que también introduce principios

ESCUELA POLITÉCNICA NACIONAL

fundamentales aplicables a sistemas ópticos modernos.

2. OBJETIVOS

- Desarrollar un modelo matemático que describa el funcionamiento de un sistema de orientación basado en pistones lineales, similar al utilizado en el espejo secundario del telescopio James Webb. Dicho modelo permitirá ajustar con precisión la posición y la inclinación de un espejo en función de un punto de interés específico (punto P), asegurando la alineación precisa del eje óptico con el objetivo deseado.

- Realizar una simulación visual del sistema para validar el modelo matemático y analizar su comportamiento en diversas configuraciones. Esto incluye estudiar cómo las variaciones en la longitud de los pistones afectan la orientación del espejo, proporcionando información clave sobre las tolerancias del sistema y sus límites operativos.

3. METEDOLOGÍA

3.1. Descripción de la solución.

La solución consiste en una aplicación en Python que permite visualizar y analizar la orientación y configuración de un telescopio basado en pistones lineales. Mediante el uso de la biblioteca Tkinter para la interfaz gráfica y Matplotlib para la representación visual, se logra un sistema interactivo e informativo.

Validación de Parámetros

El diseño y funcionamiento del telescopio secundario en este proyecto requiere cumplir ciertas restricciones geométricas y mecánicas. Estas restricciones aseguran que los componentes del sistema trabajen dentro de límites factibles y que las configuraciones proporcionadas por el usuario sean coherentes con las especificaciones del diseño.

a. Validación de Parámetros del Sistema

El sistema implementa varias condiciones, así garantizando que las entradas

ESCUELA POLITÉCNICA NACIONAL

sean válidas antes de proceder con los cálculos.

Los parámetros del espejo L, la separación entre los pistones B, y las longitudes de los pistones $(D\ y\ d\_max)$ deben cumplir:

$$(0 < L < B < d_{max})$$

Esto asegura que el espejo pueda ajustarse físicamente entre los pistones y que las longitudes de estos sean compatibles con la estructura. Por ejemplo, si *B* es menor que L, el espejo no tendría espacio para montarse, lo que haría imposible la configuración del sistema

b. Coordenadas del punto $P(x, y)$

El punto P representa la ubicación del objetivo donde el telescopio va a proyectar para eso debe pasar algunas restricciones:

- $(Py \geq 0)$: Esto asegura que el punto este en los primeros dos cuadrantes, dado que el espejo secundario opera en un plano superior al eje X.
- En caso de que $P = (0,0)$, se genera una advertencia al usuario porque este punto no tiene significado geométrico válido en el sistema. Se corrige automáticamente a $P = (1,1)$.

c. Longitudes de los pistones

Durante los cálculos, se verifica que las longitudes de los pistones $(x_1\ y\ x_2)$ estén dentro del rango permitido:

$$\mathrm{D} \leq \mathrm{x1}, \mathrm{x2} \leq \mathrm{d}_{max}$$

Esto garantiza que los pistones no excedan su capacidad de elongación o contracción, respetando las restricciones mecánicas.

d. Advertencias en caso de valores no alcanzables

ESCUELA POLITÉCNICA NACIONAL

Cuando las condiciones geométricas no permiten una solución válida (por ejemplo, si P está fuera del rango alcanzable del sistema), se emite una advertencia al usuario. En estos casos, el sistema ajusta automáticamente las posiciones para reflejar el límite más cercano.

Configuración del Cálculo

El algoritmo busca una posición válida para el espejo que permita proyectar P en su punto medio.

Utiliza un bucle iterativo para ajustar la posición vertical del espejo (M_y) y calcula:

- La dirección perpendicular hacia el punto P.
- Las longitudes de los pistones izquierdo y derecho (x_1, x_2).

Si no se encuentra una configuración válida, se ajusta al límite más cercano permitido por los pistones.

3.2. Desarrollo matemático.

El programa está diseñado para calcular y simular la orientación de un espejo secundario en un telescopio con base en coordenadas específicas (P_x, P_y), restricciones geométricas, y la configuración de los pistones que soportan el espejo. Los cálculos matemáticos subyacentes están fundamentados en principios de geometría plana y trigonometría.

A continuación, se describen las relaciones y ecuaciones principales empleadas en el código:

PARÁMETROS INICIALES DEL SISTEMA

El sistema parte de los siguientes parámetros geométricos:

- **Espejo Secundario:** Una estructura rígida de longitud L, orientada de forma tal que el objetivo $P(x, y)$ proyecta perpendicularmente sobre su punto medio $M(x, y)$.

ESCUELA POLITÉCNICA NACIONAL

- **Pistones de Soporte:** Dos elementos que sostienen los extremos del espejo, ubicados a una distancia fija *B* entre sus bases en el eje x. Las longitudes de los pistones están limitadas al intervalo $[D, d_{max}]$.

- **Punto Objetivo $\boldsymbol{P(x, y)}$:** Coordenadas que representan el objetivo hacia el cual se orienta el espejo. El programa asume que $P_y \geq 0$, es decir, que el objetivo está ubicado en los primeros dos cuadrantes.

Los parámetros deben cumplir ciertas restricciones geométricas para garantizar la configuración válida del sistema:

$$(0 < L < B < d_{max})$$

Estas relaciones aseguran que el espejo sea lo suficientemente pequeño para ser soportado por los pistones y que los pistones puedan extenderse o contraerse según los límites físicos.

Orientación del Espejo y Proyección del Punto *P*

El objetivo principal es orientar el espejo para que el punto $P(x, y)$ proyecte perpendicularmente sobre su punto medio $M(x, y)$. Este proceso implica determinar la dirección del espejo (D) y sus extremos (E_{izq} y E_{der}) en función de $M(x, y)$.

Vector Normal al Espejo

La orientación del espejo está determinada por la normal al segmento que conecta $M(x, y)$ con $P(x, y)$. El vector normal se calcula como:

ESCUELA POLITÉCNICA NACIONAL

$$N = (-P_x, M_y - P_y)$$

Dirección del Espejo (D)

La dirección del espejo (D) es perpendicular a N. Se obtiene normalizando el vector:

$$D = \left(-\frac{M_y - P_y}{||N||}, \frac{P_x}{||N||}\right), \qquad con\ ||N|| = \sqrt{P_x^2 + (M_y - P_y)^2}$$

Extremos del Espejo

La posición de los extremos del espejo se calcula en función de su punto medio $M(x, y)$ y la dirección *D*. Los extremos izquierdos ((E_{izq}) y derecho (E_{der}) se definen como:

$$x_{izq} = M_x - \frac{L}{2}D_x, y_{izq} = M_y - \frac{L}{2}D_y,$$

$$x_{der} = M_x - \frac{L}{2}D_x, y_{der} = M_y - \frac{L}{2}D_y.$$

Cálculo de las Longitudes de los Pistones

Cada pistón conecta un extremo del espejo con su respectiva base ($-B/2{,}0$ para el izquierdo y $B/2{,}0$ para el derecho). Las longitudes de los pistones se calculan utilizando la fórmula de distancia euclidiana:

- Pistón Izquierdo (x_1):

$$x_1 = \sqrt{(x_{izq} + \frac{B}{2})^2 + y_{izq}^2}$$

- Pistón Derecho (x_2):

ESCUELA POLITÉCNICA NACIONAL

$$x_2 = \sqrt{(x_{der} + \frac{B}{2})^2 + y_{der}^2}$$

Las longitudes deben cumplir las restricciones físicas:

$$D \leq x_1, x_2 \leq d_{max}$$

Barrido y Validación de Soluciones

El programa evalúa diferentes posiciones posibles M_y para en el intervalo permitido por la longitud del espejo:

Para cada posible valor de M_y, se verifican las siguientes condiciones:

$$M_y \in \left[P_y - \frac{L}{2}, P_y + \frac{L}{2}\right]$$

$y_{izq} \geq 0$ y $y_{der} \geq 0$: Los extremos del espejo deben permanecer en la región válida.

$D \leq x_1, x_2 \leq d_{max}$: Las longitudes de los pistones deber ser alcanzables.

Si no se encuentra una solución válida, el programa ajusta el sistema para que los pistones alcancen sus límites (uno en máxima elongación y otro en mínima contracción) y grafica esta configuración como la solución más cercana.

Cálculo del Ángulo del Espejo

El ángulo de orientación del espejo respecto a la horizontal se calcula a partir de la dirección D:

ESCUELA POLITÉCNICA NACIONAL

$$\theta = \arctan\left(\frac{D_y}{D_x}\right).$$

Este valor se convierte a grados para su presentación:

$$Ángulo\ en\ grados = \theta * \frac{180}{\pi}.$$

3.3. Diagrama de flujo.

En el desarrollo del programa, se identificó que el método **"Calcular_y_Dibujar"** representa el núcleo del control lógico y computacional del sistema.

3.3.1 Control Central del Programa

Este método maneja los cálculos principales necesarios para la operación del telescopio, incluyendo:

- La validación de los parámetros de entrada, garantizando que sean consistentes con las limitaciones físicas y matemáticas del modelo.
- La búsqueda de una solución para la posición y orientación del espejo, que es el objetivo principal del sistema.
- La ejecución de cálculos complejos como coordenadas, proyecciones y ángulos, los cuales determinan la funcionalidad óptima del telescopio.

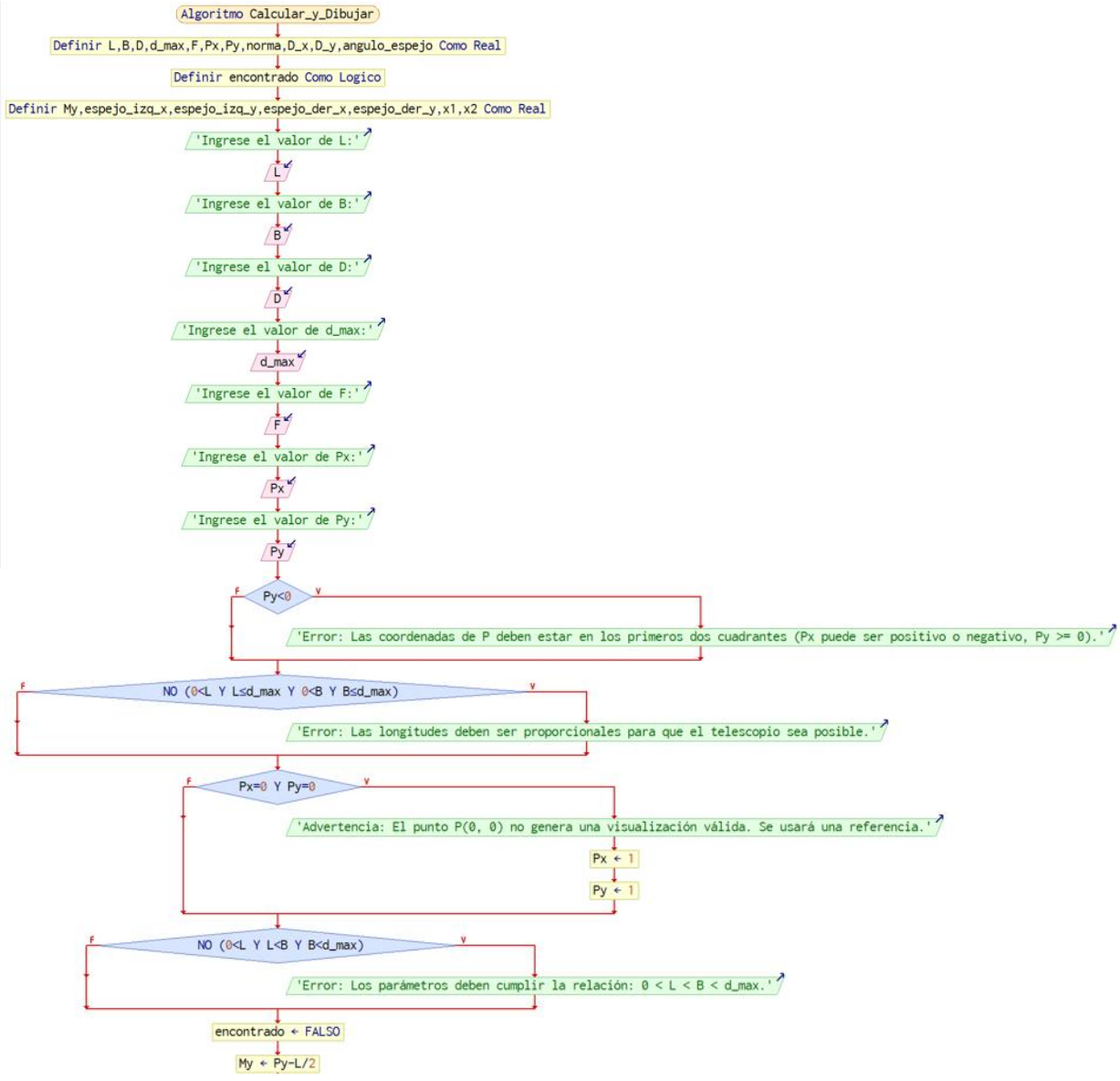
3.3.2 Cálculos geométricos y de validación

Este método es responsable de interpretar los datos proporcionados por el usuario y convertirlos en resultados matemáticos que determinan la funcionalidad del telescopio. Sin él, no sería posible determinar posiciones válidas del espejo ni garantizar que los pistones operen dentro de los límites físicos establecidos.

Representar todo el programa en un único diagrama de flujo sería redundante e

ESCUELA POLITÉCNICA NACIONAL

innecesariamente complejo. Al enfocarse en este método, se consigue una representación más clara y directa de la lógica principal, evitando elementos secundarios que no aportan al objetivo del análisis.



ESCUELA
POLITÉCNICA
NACIONAL

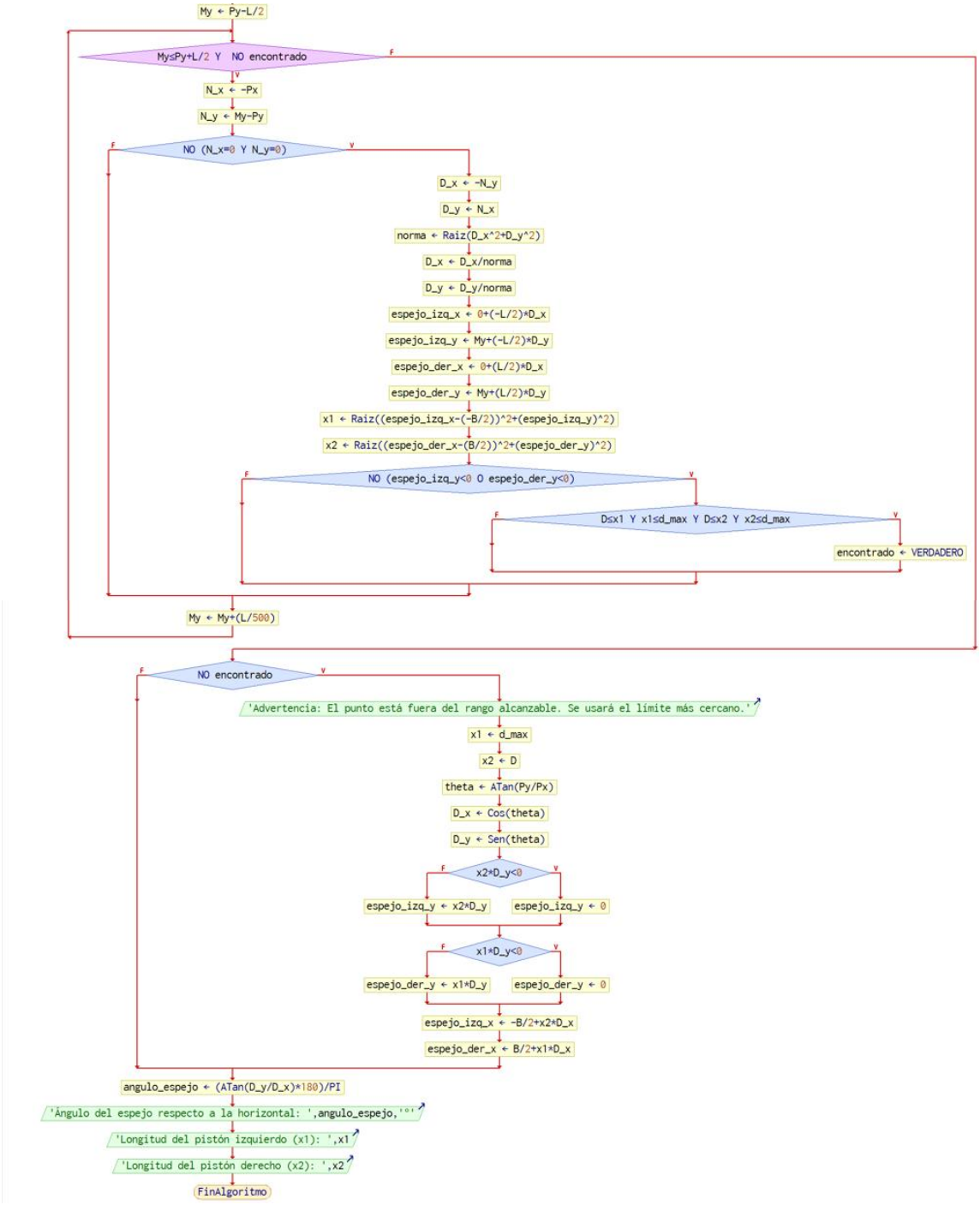


Figura 2. Diagrama de Flujo

ESCUELA POLITÉCNICA NACIONAL

3.4 Detalles importantes de la implementación

3.4.1 Importación de librerías

```
import tkinter as tk
from tkinter import messagebox
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import math
import numpy as np
```

Figura 3. Librerías.

- Tkinter: Librería estándar para crear interfaces gráficas en Python. Gestiona la entrada de datos del usuario, muestra mensajes y construye la ventana principal del programa.
- Matplotlib.pyplot: Dibuja el sistema del telescopio, pistones, espejo y punto
- Matplotlib.backends.backend_tkagg: Integra gráficos directamente en las ventas creadas con Tkinter.
- Math: Proporciona funciones matemáticas esenciales que se utilizan para realizar cálculos precisos en el programa.
- Numpy: Librería para manejo de matrices y cálculos avanzados.

3.4.2 Clase AplicacionTelescopio (def __init__)

ESCUELA
POLITÉCNICA
NACIONAL

```python
class AplicacionTelescopio:
    def __init__(self, raiz):
        self.raiz = raiz
        self.raiz.title("Telescopio James Webb")

        # Definir constantes (Longitudes del espejo y pistones)
        self.L = tk.DoubleVar(value=12.0)  # Largo del espejo
        self.B = tk.DoubleVar(value=14.0)  # Base del espejo (separación entre pistones)
        self.D = tk.DoubleVar(value=4.0)  # Longitud mínima de los pistones
        self.d_max = tk.DoubleVar(value=20.0)  # Longitud máxima de los pistones
        self.F = tk.DoubleVar(value=6.0)  # Distancia del foco al extremo izquierdo del espejo principal

        # Variables de entrada para las coordenadas de P
        self.Px_var = tk.DoubleVar(value=8.0)
        self.Py_var = tk.DoubleVar(value=12.0)

        self.crear_widgets()
        self.calcular_y_dibujar()
```

Figura 4.Clase Aplicación Telescopio

- Se almacena la ventana principal (raiz) que se pasa como argumento al crear una instancia de la clase.
- Se establece el título de la ventana principal, identificando claramente que el programa simula el telescopio James Webb.

Definición de constantes:

- **L :** Longitud del espejo principal, con un valor inicial de 12.0.
- **B :** Separación entre los pistones, con un valor inicial de 14.0.
- **D :** Valor mínimo de los pistones, inicializado en 4.0.
- **d_max :** Valor máximo permitido para los pistones, inicializado en 20.0.
- **F :** Distancia entre el foco y el extremo izquierdo del espejo, con un valor inicial de 6.0.
- **Px_var :** Coordenada X de P, valor inicializado en 8.0.
- **Py_var :** Coordenada Y de P, valor inicializado en 12.0.

ESCUELA POLITÉCNICA NACIONAL

3.4.3 Crear_widgets

```python
def crear_widgets(self):
    # Marco de entrada
    marco_entrada = tk.Frame(self.raiz)
    marco_entrada.pack(side=tk.TOP, fill=tk.X, padx=5, pady=5)

    # Entradas para los parámetros del telescopio
    tk.Label(marco_entrada, text="L (Largo del espejo):").grid(row=0, column=0, sticky=tk.W)
    tk.Entry(marco_entrada, textvariable=self.L).grid(row=0, column=1)

    tk.Label(marco_entrada, text="B (Base del espejo):").grid(row=1, column=0, sticky=tk.W)
    tk.Entry(marco_entrada, textvariable=self.B).grid(row=1, column=1)

    tk.Label(marco_entrada, text="D (Longitud mínima de los pistones):").grid(row=2, column=0, sticky=tk.W)
    tk.Entry(marco_entrada, textvariable=self.D).grid(row=2, column=1)

    tk.Label(marco_entrada, text="d_max (Longitud máxima de los pistones):").grid(row=3, column=0, sticky=tk.W)
    tk.Entry(marco_entrada, textvariable=self.d_max).grid(row=3, column=1)

    tk.Label(marco_entrada, text="F (Distancia del foco al extremo izquierdo del espejo principal):").grid(row=4, column=0, sticky=tk.W)
    tk.Entry(marco_entrada, textvariable=self.F).grid(row=4, column=1)

    # Entradas para las coordenadas de P
    tk.Label(marco_entrada, text="Px (Coordenada x de P):").grid(row=5, column=0, sticky=tk.W)
    tk.Entry(marco_entrada, textvariable=self.Px_var).grid(row=5, column=1)

    tk.Label(marco_entrada, text="Py (Coordenada y de P):").grid(row=6, column=0, sticky=tk.W)
    tk.Entry(marco_entrada, textvariable=self.Py_var).grid(row=6, column=1)

    # Botón para calcular y dibujar
    tk.Button(marco_entrada, text="Calcular y Dibujar", command=self.calcular_y_dibujar).grid(row=7, column=0, columnspan=2, pady=5)

    # Área de salida para mostrar las longitudes de los pistones y el ángulo
    self.etiqueta_salida = tk.Label(self.raiz, text="", font=('Arial', 12))
    self.etiqueta_salida.pack()

    # Área de dibujo
    self.figura = plt.Figure(figsize=(8, 8))
    self.ax = self.figura.add_subplot(111)
    self.canvas = FigureCanvasTkAgg(self.figura, master=self.raiz)
    self.canvas.get_tk_widget().pack()
```

Figura 5. Método crear_widgets.

- Campos de entrada para los parámetros geométricos y las coordenadas ***P.***
- **Botón** para ejecutar los cálculos y actualizar el gráfico.
- **Etiqueta** para mostrar los resultados del cálculo.
- **Gráfico** para visualizar el sistema en función de los datos ingresados.
- Interacción sencilla y eficiente entre el usuario y el programa.

ESCUELA POLITÉCNICA NACIONAL

3.4.4 Calcular_y_dibujar

```python
def calcular_y_dibujar(self):
    try:
        # Obtener los parámetros de entrada
        L = self.L.get()
        B = self.B.get()
        D = self.D.get()
        d_max = self.d_max.get()
        F = self.F.get()
        Px = self.Px_var.get()
        Py = self.Py_var.get()
    except tk.TclError:
        messagebox.showerror("Error", "Por favor, ingrese valores numéricos válidos.")
        return

    # Validaciones iniciales
    if Py < 0:
        messagebox.showerror("Error", "Las coordenadas de P deben estar en los primeros dos cuadrantes (Px puede ser positivo o negativo, Py ≥ 0).")
        return

    if not (0 < L <= d_max and 0 < B <= d_max):
        messagebox.showerror("Error", "Las longitudes deben ser proporcionales para que el telescopio sea posible.")
        return

    # PARCHE PARA CASO ESPECIAL (0,0)
    # Caso especial: Px = 0 y Py = 0
    if Px == 0 and Py == 0:
        messagebox.showwarning("Advertencia", "El punto P(0, 0) no genera una visualización válida. Se mostrará una posición de referencia.")
        Px = 1  # Establecer un valor mínimo para Px
        Py = 1  # Establecer un valor mínimo para Py

    # Validación de la relación 0 < L < B < d_max
    if not (0 < L < B < d_max):
        messagebox.showerror(
            "Error",
            "Los parámetros deben cumplir la relación: 0 < L < B < d_max.\n"
            "Por favor, ajusta los valores para que sean válidos."
        )
        return
```

Figura 6. Método Calcular_y_Dibujar 1.

- Los valores de los parámetros del telescopio (L, B, D, d_max, F) y las coordenadas del punto objetivo (Px, Py) son obtenidos mediante el método .get() de las variables definidas en el constructor.
- **Try-catch** garantiza que solo se procesen valores numéricos válidos antes de continuar con los cálculos.
- Se verifica que la coordenada Y del punto objetivo no sea negativa, ya que el telescopio solo opera en los primeros dos cuadrantes.
- Se valida que las dimensiones del telescopio (largo del espejo y separación entre pistones) sean proporcionales y estén dentro de los límites permitidos.
- Se asegura que las dimensiones del telescopio cumplan con una relación geométrica específica: el largo del espejo debe ser mejor que la base y ambos deben ser menores que la longitud máxima permitida.

ESCUELA POLITÉCNICA NACIONAL

```python
# Intentar encontrar la posición del espejo donde la proyección perpendicular cae en su punto medio
encontrado = False
for My in np.linspace(Py - L / 2, Py + L / 2, 500):
    N_x = -Px
    N_y = My - Py
    # Evitar división por cero
    if N_x == 0 and N_y == 0:
        continue

    # Dirección del espejo (normalizada)
    D_x = -N_y
    D_y = N_x
    norma = math.sqrt(D_x**2 + D_y**2)
    D_x /= norma
    D_y /= norma

    # Extremos del espejo
    espejo_izq_x = 0 + (-L/2) * D_x
    espejo_izq_y = My + (-L/2) * D_y
    espejo_der_x = 0 + (L/2) * D_x
    espejo_der_y = My + (L/2) * D_y

    # Longitudes de los pistones
    x1 = math.sqrt((espejo_izq_x - (-B/2))**2 + (espejo_izq_y - 0)**2)
    x2 = math.sqrt((espejo_der_x - (B/2))**2 + (espejo_der_y - 0)**2)

    # Validar que los pistones no se crucen
    if espejo_izq_y < 0 or espejo_der_y < 0:
        continue

    if D <= x1 <= d_max and D <= x2 <= d_max:
        encontrado = True
        break
```

Figura 7. Método Calcular_y_Dibujar 2.

- Búsqueda exhaustiva: Explora múltiples configuraciones posibles para encontrar una solución válida.
- Determina la orientación y posición del espejo, así como las longitudes de los pistones.
- Asegura que la solución respetará los límites físicos del sistema.

ESCUELA
POLITÉCNICA
NACIONAL

```python
# Si no se encontró una solución válida, extender o contraer al máximo permitido
if not encontrado:
    messagebox.showwarning(
        "Advertencia",
        "El punto está fuera del rango alcanzable. Se graficará el telescopio en su límite más cercano hacia la dirección del punto."
    )
    # Determinar las posiciones del pistón: uno en máxima elongación y otro en mínima
    x1 = d_max  # Pistón derecho en elongación máxima
    x2 = D      # Pistón izquierdo en elongación mínima

    # Dirección hacia el punto P
    theta = math.atan2(Py, Px)
    D_x = math.cos(theta)
    D_y = math.sin(theta)

    # Posición de los pistones en los límites máximos/mínimos
    espejo_izq_x = -B / 2 + x2 * D_x
    espejo_izq_y = max(0, x2 * D_y)  # Asegurarse de que y no sea negativo
    espejo_der_x = B / 2 + x1 * D_x
    espejo_der_y = max(0, x1 * D_y)  # Asegurarse de que y no sea negativo

    # Ajustar el espejo para que respete su longitud fija L
    espejo_centro_x = (espejo_izq_x + espejo_der_x) / 2
    espejo_centro_y = (espejo_izq_y + espejo_der_y) / 2

    espejo_izq_x = espejo_centro_x - (L / 2) * D_y
    espejo_izq_y = max(0, espejo_centro_y + (L / 2) * D_x)  # Asegurar que y no sea negativo
    espejo_der_x = espejo_centro_x + (L / 2) * D_y
    espejo_der_y = max(0, espejo_centro_y - (L / 2) * D_x)  # Asegurar que y no sea negativo

    # Ángulo del espejo respecto a la horizontal
    angulo_espejo = math.degrees(theta)

# Ángulo del espejo respecto a la horizontal
angulo_espejo = math.degrees(math.atan2(D_y, D_x))

# Mostrar las longitudes de los pistones y el ángulo
texto_salida = (
    f"Ángulo del espejo respecto a la horizontal: {angulo_espejo:.2f}°\n"
    f"Longitud del pistón izquierdo (x1): {x1:.2f}\n"
    f"Longitud del pistón derecho (x2): {x2:.2f}"
)
self.etiqueta_salida.config(text=texto_salida)

# Dibujar el sistema
self.dibujar_sistema(x1, x2, Px, Py, L, B, angulo_espejo, espejo_izq_x, espejo_izq_y, espejo_der_x, espejo_der_y)
```

Figura 8.Método Calcular_y_Dibujar 3.

- Se informa al usuario que el punto objetivo (P) está fuera del rango alcanzable y que el telescopio se ajustará al límite más cercano posible.
- Garantizar que las longitudes de los pistones respeten los límites físicos establecidos.

- Se calcula la dirección unitaria hacia el punto objetivo.
- Asegura que las coordenadas y sean siempre no negativas.
- Se ajustan para garantizar que el espejo mantenga su longitud fija L.
- Convierte el ángulo entre la dirección del espejo y la horizontal de radianes a grados.
- Muestra los resultados al usuario en la interfaz gráfica.
- Actualiza la visualización gráfica del telescopio con los nuevos parámetros calculados.

3.4.5 Dibujar_sistema

```
    def dibujar_sistema(self, x1, x2, Px, Py, L, B, angulo_espejo, espejo_izq_x, espejo_izq_y, espejo_der_x, espejo_der_y):
        self.ax.clear()

        # Coordenadas de las bases de los pistones
        A = (-B / 2, 0)
        B_punto = (B / 2, 0)

        # Dibujar los pistones
        self.ax.plot([A[0], espejo_izq_x], [A[1], espejo_izq_y], 'b-', linewidth=3, label='Pistón Izquierdo')
        self.ax.plot([B_punto[0], espejo_der_x], [B_punto[1], espejo_der_y], 'b-', linewidth=3, label='Pistón Derecho')

        # Dibujar el espejo
        self.ax.plot([espejo_izq_x, espejo_der_x], [espejo_izq_y, espejo_der_y], 'r-', linewidth=4, label='Espejo')

        # Dibujar el punto P
        self.ax.plot(Px, Py, 'go', label='Punto P')

        # Dibujar la proyección perpendicular del punto P sobre el espejo (cae en el punto medio del espejo)
        mx = (espejo_izq_x + espejo_der_x) / 2
        my = (espejo_izq_y + espejo_der_y) / 2
        self.ax.plot([Px, mx], [Py, my], 'g--', linewidth=1, label='Proyección Perpendicular')

        # Configuración del gráfico
        self.ax.set_xlabel('Eje X')
        self.ax.set_ylabel('Eje Y')
        self.ax.set_title('Orientación del Espejo Secundario')
        self.ax.legend()
        self.ax.grid(True, which='both', linestyle='--', linewidth=0.5)
        self.ax.axis('equal')

        # Centrar el telescopio en el origen
        x_min = min(-B / 2, espejo_izq_x, Px) - 2  # Margen izquierdo
        x_max = max(B / 2, espejo_der_x, Px) + 2   # Margen derecho
        y_min = 0                                  # No permitir valores negativos en Y
        y_max = max(espejo_izq_y, espejo_der_y, Py) + 2  # Margen superior

        self.ax.set_xlim(x_min, x_max)
        self.ax.set_ylim(y_min, y_max)

        # Dibujar
        self.canvas.draw()

# Crear la ventana principal
raiz = tk.Tk()
aplicacion = AplicacionTelescopio(raiz)
raiz.mainloop()
```

Figura 9. Método Dibujar_sistema.

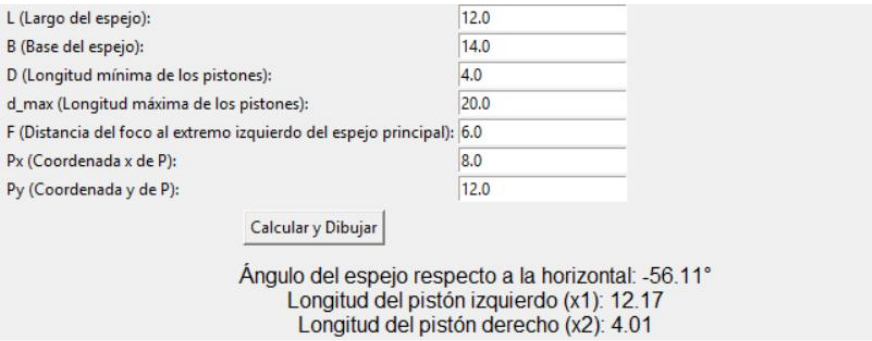
- Limpia el área de dibujo para preparar la visualización de una nueva configuración.
- Coordenadas de la base derecha del pistón.
- Une la base izquierda del pistón (A) con el extremo izquierdo del espejo (espejo_izq_x, espejo_izq_y) usando una línea azul.
- Une la base derecha del pistón (B_punto) con el extremo derecho del espejo (espejo_der_x, espejo_der_y).
- Dibuja una línea roja que conecta los extremos del espejo, representando su orientación.
- Confirmar visualmente la posición relativa del punto objetivo P.

ESCUELA POLITÉCNICA NACIONAL

4. RESULTADOS Y CONCLUSIONES:

4.1 RESULTADOS

4.1.1 Caso por defecto (Valores Iniciales)



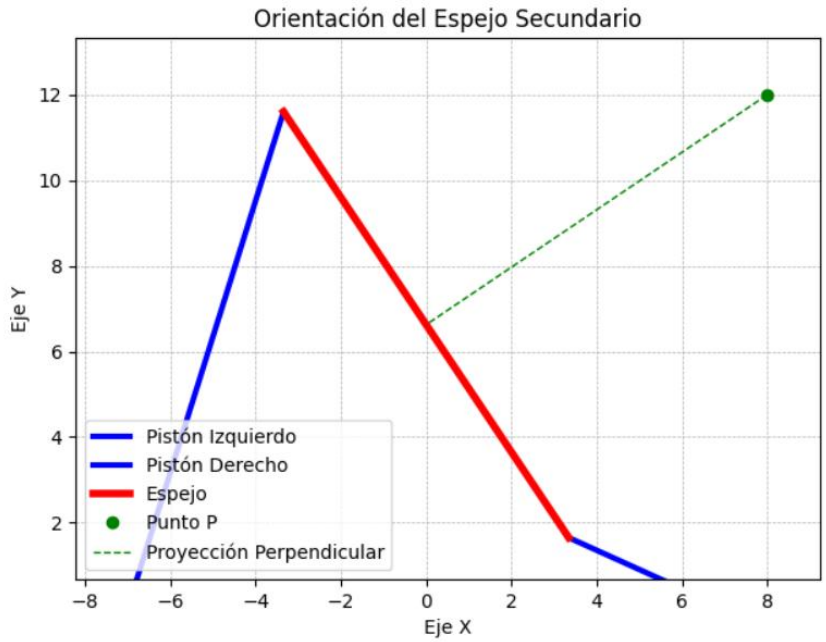
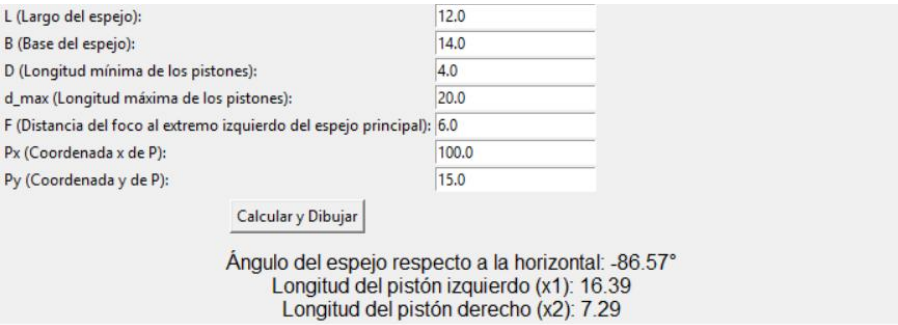


Figura 10. Caso por defecto.

ESCUELA POLITÉCNICA NACIONAL

4.1.2 Caso Extremo



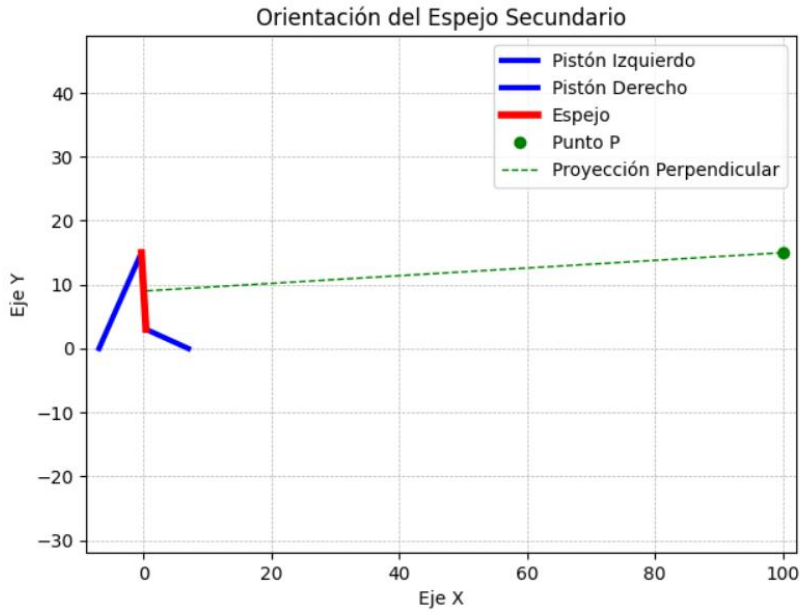


Figura 11. Caso Extremo..

4.2 CONCLUSIONES

- Se logró desarrollar un modelo matemático que describe con exactitud el funcionamiento del sistema de orientación basado en pistones lineales. Este modelo permitió calcular la posición y orientación del espejo en función de un punto de interés, garantizando la alineación precisa del eje óptico. La implementación incluyó restricciones físicas y geométricas que aseguran la operatividad del sistema dentro de los límites establecidos.

- La simulación gráfica realizada no solo valida el modelo matemático, sino que también facilita la comprensión intuitiva del sistema. A través de las visualizaciones, se pueden identificar de manera efectiva las configuraciones de los pistones, los ángulos del espejo y las restricciones geométricas involucradas.