PARAMETRIZACIÓN DE LA POSICIÓN DE UN MECANISMO DE THEO JANSEN MEDIANTE EL USO DE PYTHON Y MSC ADAMS

PARAMETRIZATION OF THE POSITION OF A THEO JANSEN MECHANISM USING PYTHON AND MSC ADAMS

Pedro Jorge De Los Santos Lara
Universidad Politécnica de Guanajuato, México
jdelossantos@upgto.edu.mx

Vignaud Granados Alejo
Universidad Politécnica de Guanajuato, México
vgranados@upgto.edu.mx

Diana Guadalupe Gutiérrez León
Universidad Politécnica de Guanajuato, México
dgutierrez@upgto.edu.mx

Recepción: 21/noviembre/2024 **Aceptación:** 11/septiembre/2025

Resumen

En este trabajo se presenta la parametrización de la posición de un mecanismo de Theo Jansen. El estudio se centra en el diseño e implementación de una clase en Python, para calcular y exportar de manera sencilla las longitudes y posiciones del mecanismo, con la posibilidad de escalarlo a partir de las dimensiones originales propuestas por Theo Jansen. El uso de un lenguaje de programación como Python, libre y de código abierto, y su integración con herramientas de CAD/CAE como MSC Adams permite automatizar el proceso de modelado, mediante scripts que generan las geometrías y restricciones del mecanismo. Los resultados muestran que esta estrategia puede reducir el tiempo requerido, hasta en un 90%, para explorar configuraciones del mecanismo y facilita su incorporación en flujos de trabajo de modelado y simulación asistida por computadora. Esto resulta útil tanto en aplicaciones académicas como en proyectos de investigación.

Palabras Clave: Mecanismo de Theo-Jansen, Análisis cinemático, Python, Parametrización, MSC Adams.

Abstract

This work presents the parametrization of the position of a Theo Jansen mechanism. The study focuses on the design and implementation of a Python class to easily compute and export the lengths and positions of the mechanism, with the possibility of scaling it based on the original dimensions proposed by Theo Jansen. The use of Python, a free and open-source programming language, and its integration with CAD/CAE tools such as MSC Adams, allows the automation of the modeling process through scripts that generate the geometries and constraints of the mechanism. The results show that this strategy can reduce the required time by up to approximately 90% *when exploring different configurations of the mechanism and facilitates its incorporation into computer-aided modeling and simulation workflows, which is highly useful in both academic and research applications.*

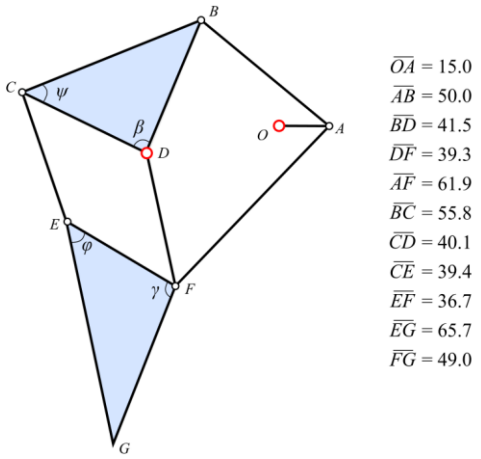
Keywords**:** *Theo-Jansen linkage, Kinematic analysis, Python, Parametrization, MSC Adams.*

1. Introducción

El mecanismo de Theo Jansen es un mecanismo muy conocido por su diseño ingenioso y eficiente, que transforma movimientos rotativos en trayectorias complejas que se asemejan a una caminata. Este mecanismo fue diseñado y optimizado por el artista holandés Theo Jansen, el cual mediante algoritmos evolutivos obtuvo las dimensiones ampliamente utilizadas en la actualidad para este mecanismo [Jansen, 2007]. En la Figura 1 se puede observar un diagrama cinemático del mecanismo, así como las longitudes de los eslabones propuestas por Theo Jansen, los círculos en rojo denotan las ubicaciones de los puntos fijos del mecanismo.

El mecanismo de Theo Jansen tiene aplicaciones potenciales en robots exploradores, sistemas de transporte no convencionales y dispositivos educativos. En los trabajos de [Singh, 2020] y [Roslee, 2021] se reporta el desarrollo de prototipos de robot cuadrúpedos que utilizan el mecanismo de Theo Jansen con el propósito de moverse por terrenos difíciles. En [Sajjan, 2021] los autores proponen un sistema híbrido que utiliza un mecanismo de Theo Jansen para automatizar las

actividades agrícolas, como alternativa de bajo costo a sistemas más industrializados.



Fuente: elaboración propia

Figura 1 Mecanismo de Theo Jansen.

La cinemática del mecanismo de Theo-Jansen ha sido ampliamente estudiada, incluyendo el enfoque analítico mediante las ecuaciones cinemáticas y de forma numérica utilizando software de CAE. En [Bhavsar, 2020] se utiliza SolidWorks para el modelado y el análisis de las trayectorias y velocidades del mecanismo, lo cual es una manera muy intuitiva de visualizar el mecanismo, sin embargo, la modificación de las relaciones de las longitudes de eslabones puede resultar muy tediosa para exploraciones iniciales del mecanismo. En [Punde, 2020] se realiza el diseño y simulación de la cinemática del mecanismo de Theo Jansen utilizando SolidWorks para el modelado geométrico y MSC Adams para el análisis de velocidades y aceleraciones.

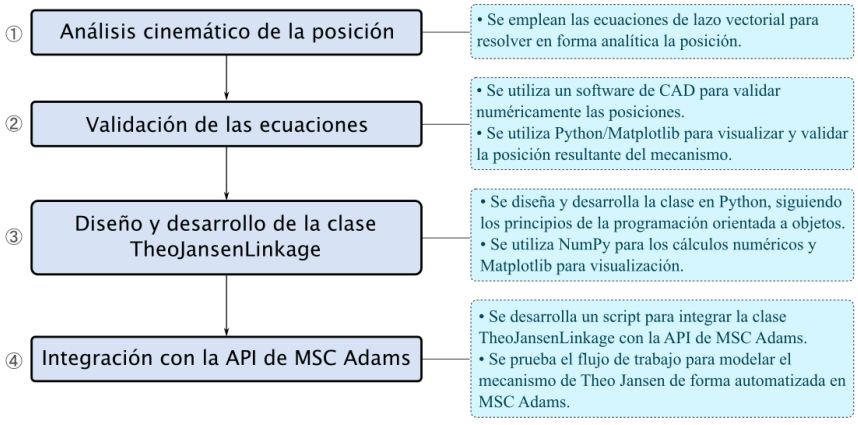
El uso de Python como herramienta de análisis cinemático y síntesis de mecanismos ha suscitado más interés en los últimos años. En GitHub se pueden encontrar librerías como Pyslvs, desarrollada por Yuan Chang [Chang, 2022], la cual es una librería que permite simular y hacer síntesis de mecanismos planos, y que incluye además una interfaz gráfica que hace más simples algunas operaciones; o la librería mechanism [Morris, 2021] que puede utilizarse para analizar mecanismos planos, incluyendo eslabonamientos, engranes y levas. Las anteriores son librerías

que abordan la simulación de mecanismos planos de manera general. Específicamente para el análisis del mecanismo de Theo Jansen utilizando Python podemos encontrar el trabajo de [Sünkün, 2022], en el cual desarrollaron una interfaz gráfica programada en Python, que permite visualizar la trayectoria del extremo del mecanismo y calcular simultáneamente la altura del paso realizando un análisis cinemático del mecanismo con las longitudes de eslabones proporcionadas por el usuario.

En este trabajo se presenta el desarrollo de una clase en Python para definir la estructura de un mecanismo de Theo-Jansen. El uso de clases permite compartir los atributos de un objeto en todos los métodos, lo cual facilita mucho realizar operaciones que modifican las propiedades de un objeto. Este enfoque también permite crear múltiples mecanismos de Theo Jansen que podrían analizarse de manera conjunta. La clase desarrollada permite la parametrización de las posiciones del mecanismo de Theo Jansen, mediante métodos que se encargan de escalar y/o desfasar las posiciones de entrada, y con esto determinar la cinemática de posición del mecanismo, para posteriormente tener la posibilidad de visualizar o exportar esta información.

2. Métodos

La parametrización de la posición del mecanismo de Theo Jansen mediante Python y MSC Adams se dividió en las etapas que se muestran en la Figura 2.

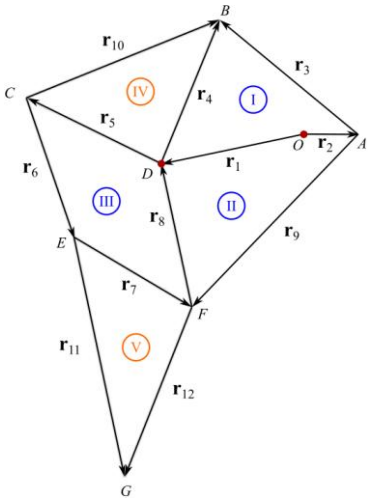


Fuente: elaboración propia

Figura 2 Metodología para la parametrización de la posición del mecanismo.

Análisis cinemático de la posición

Para plantear el problema de la posición del mecanismo de Theo Jansen se utilizó el método de ecuaciones de lazo vectorial [Norton, 2020], las ecuaciones escalares resultantes se resolvieron de forma cerrada para obtener expresiones que pudieran evaluarse en la computadora. En la Figura 3 se observa la manera en que se dispusieron cada uno de los vectores asociados a los eslabones que conforman el mecanismo completo.



Fuente: elaboración propia

Figura 3 Lazos vectoriales del mecanismo de Theo Jansen.

En el análisis de posición se asumen conocidas todas las longitudes de los eslabones, asociadas con la magnitud de cada vector, además, se consideran también conocidas las orientaciones del eslabón motriz (OA) y del eslabón fijo (OD). Los ángulos constantes β, γ, ϕ y ψ (Figura 1) se pueden determinar a partir de las longitudes de los eslabones aplicando ley de cosenos, tal como se muestra en las ecuaciones 1 a 4.

$$\beta = \cos^{-1}\left(\frac{r_4^2 + r_5^2 - r_{10}^2}{2r_4 r_5}\right) \tag{1}$$

$$\gamma = \cos^{-1}\left(\frac{r_7^2 + r_{12}^2 - r_{11}^2}{2r_7 r_{12}}\right) \tag{2}$$

$$\phi = \cos^{-1}\left(\frac{r_7^2 + r_{11}^2 - r_{12}^2}{2r_7 r_{11}}\right) \tag{3}$$

$$\psi = \cos^{-1}\left(\frac{r_5^2 + r_{10}^2 - r_4^2}{2r_5 r_{10}}\right) \tag{4}$$

En el resto de esta sección, para simplificar un poco más las expresiones resultantes se utilizan las siguientes abreviaturas: $c_i = \cos\theta_i$, $s_i = \sin\theta_i$ y $c_{ij} = \cos(\theta_i - \theta_j)$.

De la Figura 2 se puede verificar que la ecuación vectorial para el lazo I está dada por la ecuación 5. De la ecuación 5 resultan las ecuaciones escalares 6 y 7.

$$\mathbf{r}_2 + \mathbf{r}_3 - \mathbf{r}_4 - \mathbf{r}_1 = \mathbf{0} \tag{5}$$

$$r_2 c_2 + r_3 c_3 - r_4 c_4 - r_1 c_1 = 0 \tag{6}$$

$$r_2 s_2 + r_3 s_3 - r_4 s_4 - r_1 s_1 = 0 \tag{7}$$

De las ecuaciones 6 y 7 se aíslan los términos que dependen de θ_4, se elevan al cuadrado ambas ecuaciones y se suman, resultando la ecuación 8.

$$(r_4 c_4)^2 + (r_4 s_4)^2 = (r_2 c_2 + r_3 c_3 - r_1 c_1)^2 + (r_2 s_2 + r_3 s_3 - r_1 s_1)^2 \tag{8}$$

La ecuación 8 puede simplificarse para obtener la ecuación 9.

$$r_4^2 = r_1^2 + r_2^2 + r_3^2 - 2r_1 r_2 c_{12} - 2r_1 r_3 c_1 c_3 - 2r_1 r_3 s_1 s_3 + 2r_2 r_3 c_2 c_3 + 2r_2 r_3 s_2 s_3 \tag{9}$$

La ecuación 10 resulta de agrupar los términos en 9 que dependen de c_3 y s_3:

$$A_1 c_3 + B_1 s_3 = C_1 \tag{10}$$

Donde las constantes A_1, B_1 y C_1 están dadas por las ecuaciones 11, 12 y 13.

$$A_1 = -2r_1 r_3 c_1 + 2r_2 r_3 c_2 \tag{11}$$

$$B_1 = -2r_1 r_3 s_1 + 2r_2 r_3 s_2 \tag{12}$$

$$C_1 = -r_1^2 - r_2^2 - r_3^2 + r_4^2 + 2r_1 r_2 c_{12} \tag{13}$$

Una ecuación como la 10 puede resolverse haciendo la sustitución de la tangente del ángulo medio, dada por las ecuaciones 14, 15 y 16.

$$t = \tan\left(\frac{\theta_3}{2}\right) \tag{14}$$

$$\cos\theta_3 = \left(\frac{1 - t^2}{1 + t^2}\right) \tag{15}$$

$$\sin\theta_3 = \left(\frac{2t}{1 + t^2}\right) \tag{16}$$

Sustituyendo las ecuaciones 14, 15 y 16 en 10 se obtiene la ecuación 18.

$$A_1\left(\frac{1-t^2}{1+t^2}\right)+B_1\left(\frac{2t}{1+t^2}\right)=C_1 \tag{18}$$

Expandiendo y simplificando la ecuación 18 resulta en la ecuación 19.

$$-(A_1+C_1)t^2+2B_1t+(A_1-C_1)=0 \tag{19}$$

Resolviendo la ecuación 19 para t y restituyendo a θ_3 se obtiene la ecuación 20.

$$\theta_3=2\tan^{-1}\left(\frac{B_1\pm\sqrt{A_1^2+B_1^2-C_1^2}}{A_1+C_1}\right) \tag{20}$$

La ecuación 20 muestra dos soluciones que corresponden a las dos configuraciones posibles para un mecanismo de cuatro barras. De manera similar se puede proceder para resolver para θ_4, de las ecuaciones 2 y 3 se aíslan los términos que dependen de θ_3, se elevan al cuadrado y se suman, resultando en la ecuación 21. La ecuación 22 se obtiene de simplificar y agrupar la ecuación 21.

$$(r_3c_3)^2+(r_3s_3)^2=(r_4c_4+r_1c_1-r_2c_2\,)^2+(r_4s_4+r_1s_1-r_2s_2)^2 \tag{21}$$

$$A_2c_4+B_2s_4=C_2 \tag{22}$$

Donde las constantes A_2, B_2 y C_2 están dadas por las ecuaciones 23, 24 y 25.

$$A_2=-2r_2r_4c_2+2r_1r_4c_1 \tag{23}$$

$$B_2=-2r_2r_4s_2+2r_1r_4s_1 \tag{24}$$

$$C_2=-r_1^2-r_2^2+r_3^2-r_4^2+2r_1r_2c_{12} \tag{25}$$

La solución para θ_4 se determina mediante la ecuación 26. La ecuación 27 muestra la ecuación vectorial del lazo II. De la ecuación 27 resultan las ecuaciones escalares 28 y 29.

$$\theta_4=2\tan^{-1}\left(\frac{B_2\pm\sqrt{A_2^2+B_2^2-C_2^2}}{A_2+C_2}\right) \tag{26}$$

$$\mathbf{r}_2+\mathbf{r}_9+\mathbf{r}_8-\mathbf{r}_1=\mathbf{0} \tag{27}$$

$$r_2c_2+r_9c_9+r_8c_8-r_1c_1=0 \tag{28}$$

$$r_2s_2+r_9s_9+r_8s_8-r_1s_1=0 \tag{29}$$

En las ecuaciones 28 y 29 se asumen conocidas todas las longitudes de los eslabones y las direcciones de los eslabones 1 (fijo) y 2 (motriz/entrada). La solución para θ_8 está dada por la ecuación 30.

$$\theta_8 = 2\tan^{-1}\left(\frac{B_3 \pm \sqrt{A_3^2 + B_3^2 - C_3^2}}{A_3 + C_3}\right) \tag{30}$$

Las constantes A_3, B_3 y C_3 se determinan a partir de las ecuaciones 31, 32 y 33.

$$A_3 = -2r_1r_8c_1 + 2r_2r_8c_2 \tag{31}$$

$$B_3 = -2r_1r_8s_1 + 2r_2r_8s_2 \tag{32}$$

$$C_3 = -r_1^2 - r_2^2 - r_8^2 + r_9^2 + 2r_1r_2c_{12} \tag{33}$$

La solución para θ_9 está dada por la ecuación 34.

$$\theta_9 = 2\tan^{-1}\left(\frac{B_4 \pm \sqrt{A_4^2 + B_4^2 - C_4^2}}{A_4 + C_4}\right) \tag{34}$$

Las ecuaciones 35, 36 y 37 proporcionan los valores de las constantes A_4, B_4 y C_4.

$$A_4 = -2r_1r_9c_1 + 2r_2r_9c_2 \tag{35}$$

$$B_4 = -2r_1r_9s_1 + 2r_2r_9s_2 \tag{36}$$

$$C_4 = -r_1^2 - r_2^2 + r_8^2 - r_9^2 - 2r_1r_2c_{12} \tag{37}$$

La ecuación vectorial del lazo III está dada por la ecuación 38.

$$\mathbf{r}_5 + \mathbf{r}_6 + \mathbf{r}_7 + \mathbf{r}_8 = \mathbf{0} \tag{38}$$

De la ecuación 38 resultan las ecuaciones escalares 39 y 40.

$$r_5c_5 + r_6c_6 + r_7c_7 + r_8c_8 = 0 \tag{39}$$

$$r_5s_5 + r_6s_6 + r_7s_7 + r_8s_8 = 0 \tag{40}$$

En las ecuaciones 39 y 40 se asumen conocidas todas las longitudes de los eslabones involucrados ($r_5, r_6, r_7\ y\ r_8$), así como la dirección del eslabón 8 (θ_8), que fue calculada en el lazo II. El valor de θ_5 se puede determinar mediante la ecuación 41, a partir del valor ya conocido de θ_4, puesto que ambos difieren únicamente por un valor constante β, tal como se observa en la Figura 1. La ecuación 42 muestra la solución para θ_6 a partir de las ecuaciones 39 y 40.

$$\theta_5 = \theta_4 + \beta \tag{41}$$

$$\theta_6 = 2\tan^{-1}\left(\frac{B_5 \pm \sqrt{A_5^2 + B_5^2 - C_5^2}}{A_5 + C_5}\right) \tag{42}$$

Donde las constantes A_5, B_5 y C_5 están dadas por las ecuaciones 43, 44 y 45.

$$A_5 = -2r_5r_6c_5 + 2r_6r_8c_8 \tag{43}$$

$$B_5 = -2r_5r_6s_5 + 2r_6r_8s_8 \tag{44}$$

$$C_5 = -r_5^2 - r_6^2 + r_7^2 - r_8^2 - 2r_5r_8c_{58} \tag{45}$$

La ecuación 46 muestra la solución para θ_7 a partir de las ecuaciones 39 y 40.

$$\theta_7 = 2\tan^{-1}\left(\frac{B_6 \pm \sqrt{A_6^2 + B_6^2 - C_6^2}}{A_6 + C_6}\right) \tag{46}$$

Las constantes A_6, B_6 y C_6 se determinan a partir de las ecuaciones 47, 48 y 49.

$$A_6 = 2r_5r_7c_5 + 2r_7r_8c_8 \tag{47}$$

$$B_6 = 2r_5r_7s_5 + 2r_7r_8s_8 \tag{48}$$

$$C_6 = -r_5^2 + r_6^2 - r_7^2 - r_8^2 - 2r_5r_8c_{58} \tag{49}$$

Las orientaciones de los eslabones 10, 11 y 12 se pueden determinar escribiendo ecuaciones para los lazos IV y V, sin embargo, también se pueden obtener geométricamente. De las Figuras 1 y 2 se observa que θ_{10}, θ_{11} y θ_{12} se pueden calcular mediante las ecuaciones 50, 51 y 52. Con esto quedan determinadas todas las orientaciones de los eslabones que conforman el mecanismo. Las ecuaciones obtenidas se programaron en Python y se visualizaron utilizando la librería Matplotlib [Hunter, 2007]. Posteriormente se validaron utilizando un software de CAD para analizar de forma gráfica la posición del mecanismo de Theo Jansen. Las expresiones ya programadas y validadas se utilizaron posteriormente en el desarrollo de la clase.

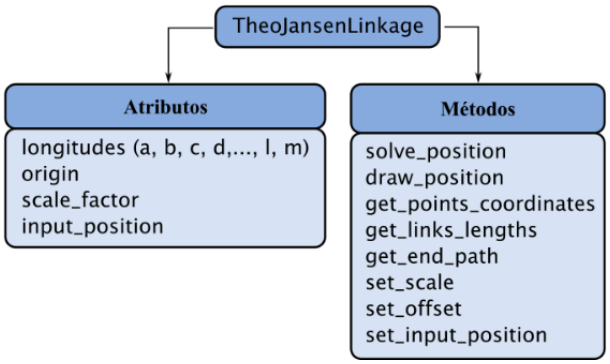
$$\theta_{10} = \theta_5 + \psi - 180° \tag{50}$$

$$\theta_{11} = \theta_7 - \phi \tag{51}$$

$$\theta_{12} = \theta_7 + \gamma - 180° \tag{52}$$

Diseño y desarrollo de la clase *TheoJansenLinkage*

Una vez realizado el análisis cinemático se procedió al diseño y desarrollo de una clase en Python denominada *TheoJansenLinkage*, la cual está estructurada de la manera que se muestra en la Figura 4. Se utilizó la librería NumPy [Harris, 2020] para implementar el cálculo de la posición mediante las ecuaciones obtenidas en la sección previa. Para el trazo del diagrama del mecanismo se utilizó Matplotlib.



Fuente: elaboración propia

Figura 4 Estructura de la clase TheoJansenLinkage

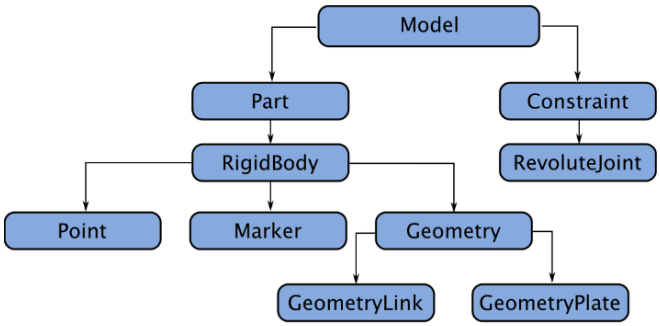
La clase cuenta con un atributo para cada una de las longitudes del mecanismo, las cuales pueden escalarse por un factor utilizando el método set_scale. El mecanismo también puede desfasarse de su posición original con el método set_offset, por defecto se considera que el punto O estará en el origen de coordenadas. El método solve_position permite resolver el problema de la posición del mecanismo para un valor específico del eslabón motriz establecido mediante set_input_position. Es posible también mostrar un diagrama simplificado del mecanismo utilizando el método draw_position.

El método get_points_coordinates permite obtener las coordenadas de cada uno de los puntos que conforman el mecanismo. El método get_links_lengths permite obtener las longitudes de cada uno de los eslabones. Utilizando get_end_path se puede obtener la trayectoria descrita por el punto G. Tanto las longitudes como las coordenadas de los puntos se almacenan en un diccionario, de tal manera que el usuario puede acceder a cada valor con el nombre correspondiente a cada elemento. El método export_points_coordinates permite exportar las coordenadas de los puntos del mecanismo a un archivo CSV, que luego puede ser importado en cualquier software de CAD/CAE.

Probando la clase con la API Python de MSC Adams

MSC Adams es un software de simulación cinemática y dinámica multicuerpo, que proporciona una API Python para realizar las operaciones de modelado y simulación, lo cual posibilita la creación de scripts para automatizar dicho proceso

[Hexagon, 2024]. En la Figura 5 se muestra la estructura jerarquizada de las clases involucradas para poder crear un mecanismo plano articulado utilizando la API de MSC Adams.

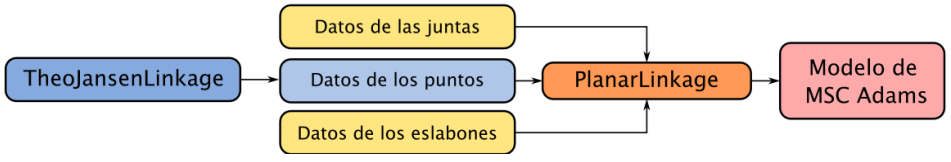


Fuente: elaboración propia

Figura 5 Estructura de clases de la API de MSC Adams para un modelo básico.

Teniendo en cuenta la estructura de la Figura 5 se programó una clase denominada PlanarLinkage que automatiza la creación de un mecanismo plano articulado, pasando como datos de entrada la información de los puntos, eslabones y juntas. Esta clase está construida encima de las clases proporcionadas por la API de MSC Adams. En la Figura 6 se observa el diagrama del proceso de automatización en la creación del modelo de un mecanismo de Theo Jansen.

Los datos de las juntas y los eslabones se pasan como diccionarios previamente definidos por el usuario, y para un mecanismo específico son valores constantes. La información de los puntos es un dato que se toma directamente del cálculo de las coordenadas proporcionadas por la clase TheoJansenLinkage. Es importante considerar que en este caso se definieron de forma automatizada las geometrías y restricciones, un proceso más completo incluye las definiciones de movimientos y fuerzas, así como la simulación y el postprocesado de variables de interés.



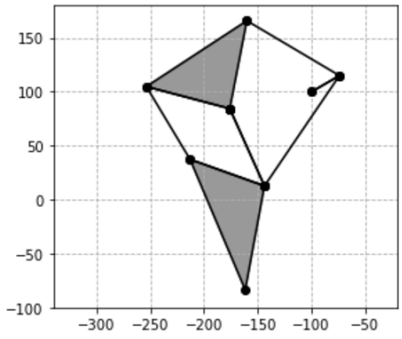
Fuente: elaboración propia

Figura 6 Proceso para automatizar el modelado de un mecanismo de Theo Jansen

3. Resultados

En la Figura 7 se muestra un ejemplo de cómo utilizar la clase desarrollada, se puede notar que el mecanismo está representado para la posición establecida ($\pi/6\ rad$) mediante el método set_input_position. Las longitudes originales han sido escaladas por un factor de 2 y el mecanismo ha sido desfasado -100 unidades en la dirección de x y 100 unidades en la dirección de y.

```
In [1]: import numpy as np
        from theo_jansen import TheoJansenLinkage

In [2]: tj = TheoJansenLinkage()
        tj.set_input_position(np.pi/6)
        tj.set_scale(2)
        tj.set_offset(-100,100)
        tj.draw_position()
```



Fuente: elaboración propia

Figura 7 Ejemplo de uso de la clase TheoJansenLinkage.

La Figura 8 muestra la obtención mediante el método get_points_coordinates las coordenadas de cada punto del mecanismo. Este método devuelve un diccionario con la información requerida, de tal manera que el usuario puede acceder enseguida a un valor específico utilizando la clave del punto correspondiente.

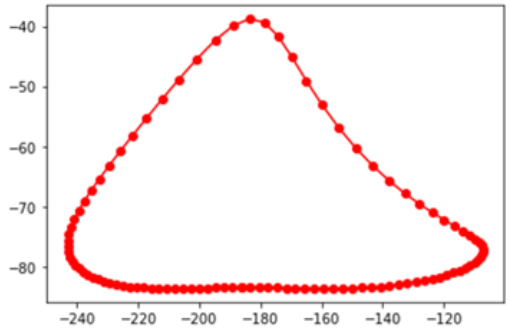
```
In [4]: tj.get_points_coordinates()

Out[4]: {'O': array([-100.,  100.]),
         'A': array([-74.01923789, 115.        ]),
         'B': array([-160.11602199,  165.86593916]),
         'C': array([-253.55186891,  104.8388754 ]),
         'D': array([-176. ,   84.4]),
         'E': array([-212.91418677,   37.32577453]),
         'F': array([-143.76313506,   12.71496294]),
         'G': array([-161.61269909,  -83.64578148])}
```

Fuente: elaboración propia

Figura 8 Obteniendo las coordenadas de los puntos.

En la Figura 9 se muestra cómo obtener la trayectoria del extremo (pie) del mecanismo. La cantidad de puntos a evaluar en la trayectoria de salida se puede especificar mediante un entero pasado como argumento al método get_end_path. La información de esta trayectoria se puede graficar o bien almacenar en arreglos de NumPy para posteriores análisis.

```
In [5]: import matplotlib.pyplot as plt

        x,y = tj.get_end_path().T
        plt.plot(x,y, "r-o")
Out[5]: [<matplotlib.lines.Line2D at 0x254e24a3a00>]
```



Fuente: elaboración propia

Figura 9 Trayectoria del extremo del mecanismo.

En la Figura 10 se muestra el script utilizado para crear el modelo utilizando la API Python de MSC Adams, por cuestiones del espacio se omite la importación de los módulos y la definición de la clase PlanarLinkage; se observa que la información de los puntos se obtiene mediante el método get_points_coordinates de la clase TheoJansenLinkage.

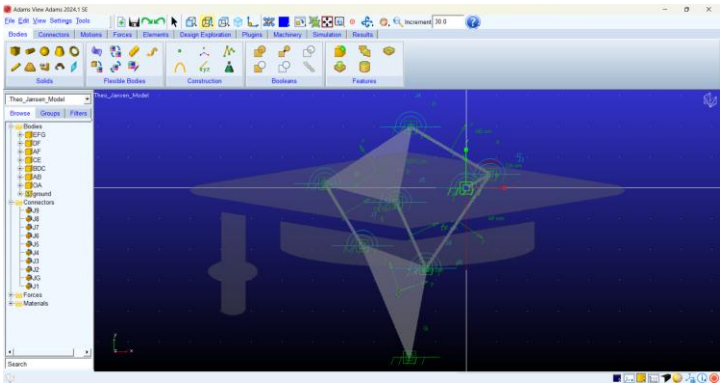
En la Figura 11 se observa el modelo de MSC Adams View creado con este script. En este modelo quedan definidos los eslabones y las juntas del mecanismo, cuyas posiciones están también parametrizadas mediante los puntos de diseño utilizados en MSC Adams View.

El uso de la clase desarrollada, integrada con la API Python de MSC Adams, permite reducir de manera considerable los tiempos de modelado de un mecanismo de este tipo. A continuación, se presentan los tiempos estimados, para un usuario promedio, siguiendo tanto el flujo de trabajo manual como el automatizado mediante la clase desarrollada en Python.

```
MODEL_NAME = "Theo_Jansen_Model"
tj = TheoJansenLinkage()
tj.set_input_position(np.pi/6)
tj.scale_lengths(3)

points = tj.get_points_coordinates()
POINTS = { "O": list(points["O"]) + [0], "A": list(points["A"]) + [0],
           "B": list(points["B"]) + [0], "C": list(points["C"]) + [0],
           "D": list(points["D"]) + [0], "E": list(points["E"]) + [0],
           "F": list(points["F"]) + [0], "G": list(points["G"]) + [0],
}

LINKS = {"OA": ("O","A"), "AB": ("A","B"),
        "BDC": ("B","D","C"), "CE": ("C","E"),
        "AF": ("A","F"),"DF": ("D","F"),
        "EFG": ("E","F","G"),
}

JOINTS = {
    "J1": ("OA","ground","O"), "JG": ("BDC","ground","D"),
    "J2": ("AB","OA","A"), "J3": ("AF","OA","A"),
    "J4": ("BDC","AB","B"), "J5": ("BDC","DF","D"),
    "J6": ("BDC","CE","C"), "J7": ("CE","EFG","E"),
    "J8": ("DF","EFG","F"), "J9": ("AF","EFG","F"),
}

pl = PlanarLinkage(MODEL_NAME, POINTS, LINKS, JOINTS)
pl.create_parametrized_linkage()
```

Fuente: elaboración propia

Figura 10 Script para modelar el mecanismo en MSC Adams.



Fuente: elaboración propia

Figura 11 Modelo en MSC Adams View del mecanismo.

Flujo de trabajo clásico de MSC Adams:

- Análisis de la posición utilizando un método gráfico en un software CAD (~5 minutos)
- Obtención de las coordenadas de los puntos del mecanismo (~2 minutos)
- Creación de los puntos que definen la posición del mecanismo (~3 minutos)
- Creación de los eslabones (~4 minutos)
- Creación de las juntas (~6 minutos)

Flujo de trabajo asistido mediante la parametrización en Python:

- Ajuste de los parámetros en el script (~1 minuto)
- Ejecución del script en MSC Adams (~1 minuto)

Se observa que la reducción del tiempo es significativa, pasando de proximadamente 20 minutos para modelar manualmente un mecanismo de este tipo en MSC Adams, a menos de dos minutos utilizando el script de la parametrización.

4. Discusión

La clase desarrollada en este trabajo ha demostrado ser útil para la parametrización de la posición del mecanismo de Theo Jansen, permitiendo escalar y desplazar la configuración original del mecanismo. Esto puede resultar necesario cuando se requiere analizar la cinemática del mecanismo para varias configuraciones, o bien trabajar con algún tipo de optimización. En comparación con el uso de software CAD para el análisis de posición y/o de la trayectoria del extremo ([Bhavsar, 2020] y [Punde, 2020]), la ventaja es que explorar diversas configuraciones del mecanismo de Theo Jansen es mucho más rápido y sencillo, con una reducción estimada del tiempo de hasta un 90%. Respecto al trabajo de [Sünkün, 2022], la ventaja radica en que no es necesario introducir las longitudes de los eslabones de forma manual y, que además del cálculo y visualización, se tiene la posibilidad de exportar los datos a un archivo de texto plano o bien integrarlo directamente en el flujo de trabajo de un software de CAD/CAE que cuente con una API en Python. El uso de Python como lenguaje de programación tiene la ventaja de que es fácilmente integrable con muchos programas de análisis en ingeniería que lo utilizan como lenguaje de scripting, particularmente aquí se mostró que integrar la clase con la API Python de MSC Adams es sencillo y permite la automatización del proceso de modelado.

5. Conclusiones

En este trabajo se ha presentado el diseño e implementación de una clase en Python que permite la parametrización de la posición del mecanismo de Theo Jansen, lo que la hace adecuada para tareas relacionadas con el diseño cinemático de este mecanismo. Esta herramienta puede utilizarse en el ámbito educativo o en actividades de investigación, su uso permite ahorrar tiempo al momento de explorar las dimensiones y configuraciones del mecanismo, además, es posible modificar de

forma independiente las longitudes de cada eslabón, lo cual permitiría analizar las trayectorias de salida sin mantener las proporciones originales propuestas por Theo Jansen. En el estatus actual la clase permite analizar la posición del mecanismo, no obstante, sería deseable expandir las capacidades para incluir el análisis de las velocidades y aceleraciones, lo cual podría lograrse sin sobresaltos a partir de las ecuaciones de posición. Incluir capacidades de análisis dinámico sería también una característica formidable.

6. Referencias y Bibliografía

[1] Bhavsar, K., Darji, P., Gohel, D., Modi, J., & Parmar, U. (2020). Comparison of kinematic analysis of robot made of conventional Theo Jansen mechanism, modified Theo Jansen mechanism of PLA and modified Theo Jansen mechanism of mild steel. In Recent Trends in Mechanical Engineering: Select Proceedings of ICIME 2020 (pp. 545-559). Singapore: Springer Singapore.

[2] Chang, H. (2022). Pyslvs-UI: An open-source planar linkage mechanism simulation and mechanical synthesis system (v22.07.0). GitHub. https://github.com/KmolYuan/Pyslvs-UI

[3] Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

[4] Hexagon (2024). Adams 2024.1 Python Interface Help.

[5] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(03), 90-95.

[6] Jansen, T. (2007). The great pretender.

[7] Morris, G. (2021). Mechanism: A visual tool to aid engineers with the design process for mechanisms, cams, and gears (v1.1.10). GitHub. https://github.com/gabemorris12/mechanism

[8] Norton, R. (2020). Diseño de Maquinaria; 6ta Edición; México DF.

[9] Punde, Y., Dhande, Y., Chopde, A., & Bagade, S. (2020). Design and Linkage Analysis of Theo Jansen Mechanism. *International Journal of Engineering Research & Technology (IJERT)*, *9*(09), 259-263.

[10] Roslee, H. H., Tew, J. C., Ismail, M. A. U., Adli, A. A. A., Othman, W. A. F. W., Wahab, A. A. A., & Alhady, S. S. N. (2021). Development of Theo Jansen inspired all-terrain quadruped mini mobile robot. In *Journal of Physics: Conference Series* (Vol. 1969, No. 1, p. 012009). IOP Publishing.

[11] Sajjan, S. C., Nadaf, S., Deshpande, S., Rehan, N. M. M., & Sushma, N. R. (2021). Mechanical ox–Replacement of tractors and farm animals for agriculture practices. *Materials Today: Proceedings*, *47*, 2529-2536.

[12] Singh, R., & Bera, T. K. (2020). Walking model of Jansen mechanism-based quadruped robot and application to obstacle avoidance. Arabian Journal for Science and Engineering, 45(2), 653-664.

[13] Sünkün, S., Parlak, B. O., Yıldırım, A., & Yavaşoğlu, H. A. (2022). Design of a Toolbox for Kinematic Analysis of Jansen's Linkage. Journal of Computer Science, 112-119.