Proyecto: Mecanismo Servo-Gripper

Dinámica Computacional de Sistemas Multicuerpo

Camilo Andres Vera Ruiz

caverar@unal.edu.co

1. Resumen

En este informe se presenta el planteamiento de un mecanismo de Gripper, incluyendo su respectivo modelado CAD, su simulación dinámica mediante Autodesk Inventor y la comparación de la cinemática obtenida mediante Inventor, con una implementación computacional realizada en Python.

Inicialmente se selecciona un mecanismo preexistente a partir del cual se aproximan las magnitudes, por medio de una simulación dinámica preliminar en el plano mediante Inventor. A partir de allí se define un alcance para la simulación cinemática y dinámica completa en Inventor, donde se llega a la conclusión de que lo más conveniente es trabajar el mecanismo únicamente con los elementos que se mueven en el plano, despreciando la fricción, de tal forma que sea posible realizar una comparación satisfactoria con los resultados de la cinemática de una implementación computacional. A continuación se desarrolla el diagrama cinemático así como el cálculo de movilidad, luego se desarrolla el CAD del mecanismo en 3D, con el fin de obtener las magnitudes de las masas y los momentos de Inercia de cada eslabón.

Se plantea una ecuación de gobierno cinemático y se realiza la simulación del mecanismo mediante un ensamble en 3D en Inventor, rescatando los resultados para las cargas resultantes en las juntas rotacionales, y la fuerza necesaria para la imposición del gobierno cinemático. Finalmente se desarrolla una implementación computacional de la cinemática del mecanismo en el lenguaje de programación Python, comparando los resultados de: posición, velocidad y aceleración, de uno de los eslabones al final de las cadenas cinemáticas; y el ángulo, velocidad y aceleración de la rotación de un eslabón, al comienzo de otra de las cadenas cinemáticas; con los resultados de Inventor, lo cual permitió evidenciar un excelente desempeño en la formulación computacional, presentando errores relativos muy pequeños, que se pueden evidenciar en graficas comparativas al final del documento.

2. Planteamiento y Funcionamiento del Mecanismo

Se plantea desarrollar el análisis y simulación del mecanismo de un efector final pensado para un manipulador serial, el cual es utilizado para la sujeción y transporte de piezas. El diseño original tiene su origen en una animación de un canal en YouTube especializado en la simulación de mecanismos [1]. En la Figura 1 se presenta una captura de la animación.

Figura 1: Mecanismo Original [1]

El mecanismo consta de un servomotor conectado a un tornillo de avance, el cual se encarga de mover el eslabón de color verde que está restringido en una corredera a la bancada que se muestra en color amarillo. El desplazamiento del eslabón verde acciona el mecanismo de tal forma que al avanzar el eslabón con respecto al motor, las mordazas de color negro se abren, de igual forma en caso de retroceder las mordazas se cierran, de esta forma el movimiento del servomotor es traducido en la apertura y cierre del mecanismo.

Cabe añadir que el tornillo de avance es bastante compacto, probablemente debido a que el mecanismo está pensado para ser utilizado con servomotores de bajo perfil, que usualmente tienen un rango de movimiento restringido, por lo tanto el mecanismo debe estar dispuesto para que el eslabón verde pueda avanzar y retroceder completamente, sin que el motor alcance el rango máximo de posición del motor, que suele estar en un rango típico de unos 180° aproximadamente, para los servomotores más típicos del mercado, sin embargo dado que no se conocen las dimensiones, se tiene libertad en la selección del paso del tornillo de avance, motivo por el cual no es necesario imponer una distancia para la junta prismática del eslabón verde, ya que selo se requiere escoger el paso, una vez se conozca la distancia del recorrido completo de la junta.

Debido a que no se conocían las dimensiones del diseño original, y tampoco se contaba con un diseño CAD, se desarrolló un diseño preliminar en el plano, por medio del software Autodesk Inventor 2024, allí se realizó una simulación dinámica en 2D con el fin de ajustar las dimensiones de los eslabones de tal forma que el mecanismo se comporte de la misma forma en que lo hace la animación. En la figura 2 se muestra una captura de dicha simulación donde se muestran las dimensiones de los eslabones en milímetros.

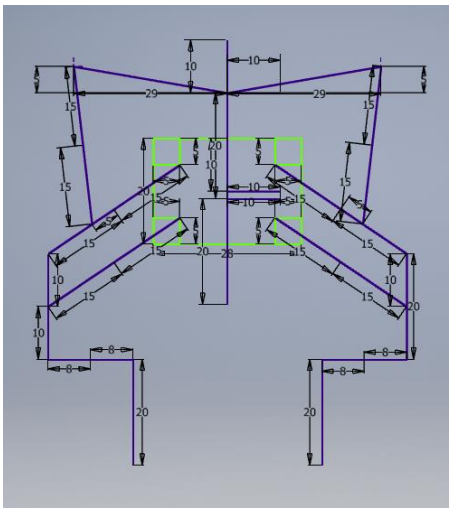

Figura 2: Simulación Dinámica y dimensiones del mecanismo en 2D.

3. Alcance de Proyecto

Dado que el mecanismo cuenta con varios eslabones, y se pretende realizar una comparación con una implementación computacional, se plantean algunas simplificaciones que faciliten la implementación a nivel de programación, por este motivo se decide ignorar la fricción en las juntas rotacionales, y se toman únicamente los eslabones cuyo movimiento se da en el plano, de tal forma que la entrada de movimiento al mecanismo, se encuentra directamente en el eslabón central de color verde, que se muestra en la figura 1. Se puede decir que esta simplificación es bastante razonable, ya que a partir de las fuerzas en el eslabón central, es posible calcular el torque requerido para el motor, conociendo las características del tornillo de avance. Por otro lado se espera que la fricción en las juntas rotacionales, no sea significativa en los valores resultantes de las cargas en las juntas y en la fuerza impuesta en el eslabón central, dado que en un diseño real, se puede disponer de lubricación que permita reducir la fricción a valores mínimos, además debido al tamaño del mecanismo, se espera que las cargas resultantes sobre las juntas rotacionales, tengan valores muy pequeños.

Teniendo en cuenta que se trata de un Gripper pensado para la tarea de pick and place, se plantea una condición de carga simplificada, en la que un peso constante de 1N (aproximadamente 100 gramos) es aplicado en las mordazas del mecanismo (0.5N en cada mordaza) con el objetivo de simular el peor caso de carga para el mecanismo, un objeto flexible que únicamente será depositado una vez que las mordazas se abran completamente. De esta manera en la simulación solo se tendrán las fuerzas de gravedad y los dos pesos de 0.5N impuestos sobre cada mordaza.

De la simulación se plantean como relevantes: las magnitudes absolutas de las fuerzas en cada junta rotacional, la fuerza requerida para mover el eslabón central, y la posición, velocidad y aceleración de una de las mordazas, y ya que el mecanismo es simétrico, solo es necesario analizar una junta o mordaza de cada uno de los pares de simetria.

4. Diagrama cinemático y cálculo de movilidad

A partir de las simplificaciones planteadas se infiere que un diagrama cinemático en el plano, es suficiente para describir el mecanismo. En la Figura 3 se muestra el diagrama cinemático desarrollado en el software draw.io, donde se muestran 10 eslabones, 12 juntas rotacionales enumeradas de R1 a R12, y una junta prismática P1 que gobierna al mecanismo por medio del eslabón 5, el cual corresponde al eslabón central color verde de la Figura 1, es decir aquel cuyo movimiento es producido por el tornillo de avance y por lo tanto por el servomotor. También hay dos puntos de interés A y B que representan el punto medio de las mordazas color negro en la figura 1, las cuales nunca llegan a tocarse puesto que el mecanismo no está pensado para la sujeción de elementos muy delgados, y que se toman como los puntos de aplicación del peso de la carga levantada por el Gripper.

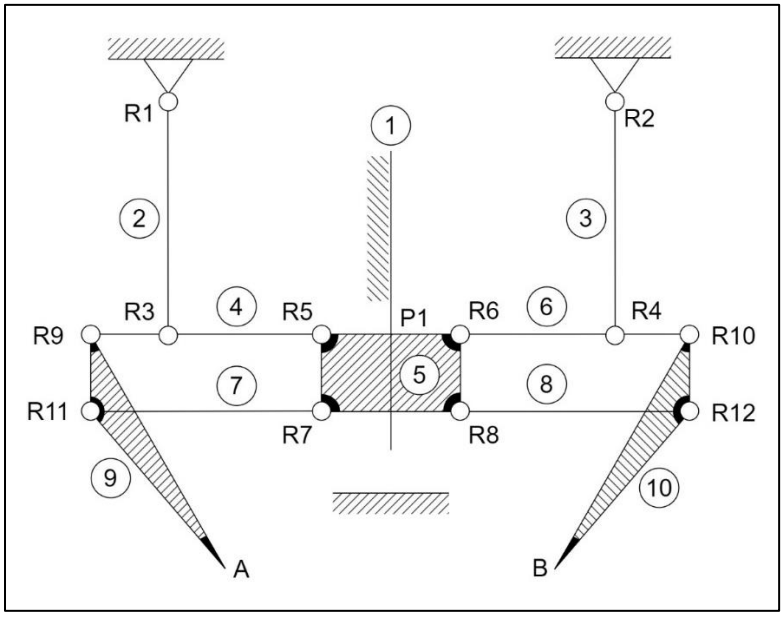

Figura 3: Diagrama Cinemático del mecanismo plano.

Con el diagrama se plantea el cálculo de la movilidad mediante la ecuación de Grübler:

- # Eslabones (n): 10
- # Juntas 1 GDL (J1): 12 Rotacionales + 1 Prismática
- # Juntas 2 GDL (J2): 0

$$m = \ 3(n-1) - 2J_1 - J_2 = 3(10-1) - 2(13) - (0)$$
$$m = 27 - 26 = 1$$

Como se puede evidenciar en la expresión matemática anterior, la movilidad es la esperada teniendo en cuenta que solo hay un elemento de gobierno en el mecanismo, por lo cual es razonable pensar que el diagrama cinemático está planteado apropiadamente.

5. Diseño CAD y Dimensiones

A partir del boceto simplificado del mecanismo, se procede a desarrollar un modelo completo en 3D, con el fin de asignar un material, calcular masas, inercias, y ajustar adecuadamente la sujeción de los eslabones para que el modelo represente un prototipo realista del mecanismo. No se incluyen ni el eje roscado ni el motor, dado que no son relevantes para la simulación.

En la Figura 4 se muestra un ensamble del mecanismo, en color amarillo se muestra la bancada, en color blanco el eslabón 5 que gobierna el mecanismo, en color rojo los acopladores que transmiten el movimiento, en color gris las mordazas, y en color negro los pines que actúan como las uniones de las juntas rotacionales.

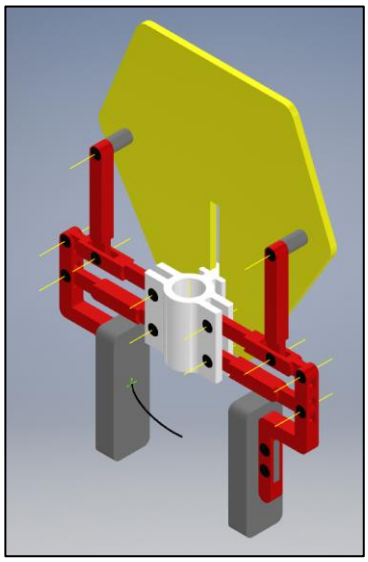

Figura 4: Ensamble del mecanismo en 3D.

En la Figura 5 se muestra un subensamble de unos de los eslabones, evidenciando un diseño a detalle, en donde se tiene en cuenta el espacio necesario para que todos eslabones rojos encajen apropiadamente entre sí, y así mismo permitan el movimiento sin contacto entre las superficies del eslabón 5 en color blanco, con el resto de los eslabones en color rojo.

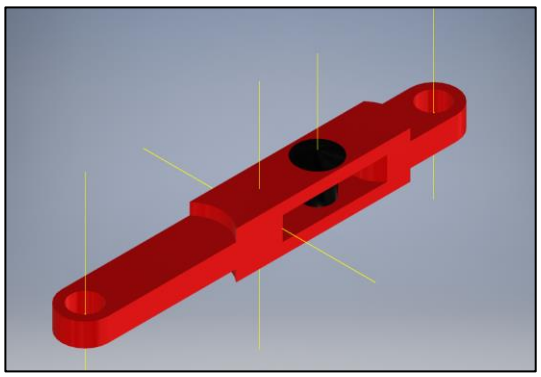

Figura 5: Subensamble de Eslabón 4.

El material seleccionado fue aluminio 6061, dado que se trata de un Gripper, el cual debe ser lo mas ligero posible para maximizar la carga máxima que puede transportar el mecanismo, y por tanto un posible robot que lo utilice. Las juntas rotacionales son tan pequeñas que no es posible el uso de rodamientos, además de no ser representativo de un producto real puesto que se incrementarían los costos innecesariamente, motivo por el cual los bujes de color negro, serian tornillos o remaches metálicos de 2.5mm, a los que también se les asigno aluminio como material.

Los eslabones de color rojo tienen un perfil cuadrado de 4mm x 4mm, con aberturas para el eslabonamiento y ensamble.

En la tabla 1 se presentan las magnitudes relevantes de cada eslabón, su masa, su momento de inercia con respecto al eje z perpendicular al plano del mecanismo, así como su magnitud en milímetros.

Tabla 1: Magnitudes de los Eslabones del Mecanismo

Eslabón	Magnitud (mm)	Masa (g)	I_{zz} $(g\ mm^2)$
1		*Bancada*	
2	30	1.2216	89.0954
3	30	1.2216	89.0954
4	30	0.8591	62.3255
5	23 x 20	8.1075	55.0160
6	30	0.8591	62.3255
7	30	0.9878	61.4218
8	30	0.9878	61.4218
9	20 x 12	9.1413	1535.6818
10	20 x 12	9.1413	1535.6818

6. Ecuación de Gobierno

Dado que el mecanismo es controlado por un servomotor de bajo perfil, se intuye que se trabaja con velocidad constante, por lo tanto la condición de gobierno de carácter cinemático, se plantea como una rampa ascendente y luego descendente sobre la posición, la cual desplaza el eslabón 20 mm hacia abajo, y luego lo retrocede en 20mm, durante un tiempo total de 1s. Matemáticamente esto se puede expresar con una función a trozos como la que se muestra a continuación, donde y_{P1}^5, representa la posición relativa de la junta prismática con respecto al centro de coordenadas del eslabón 5.

$$R_y^5(t) + y_{P1}^5 = \begin{cases} -40t\ si\ t \leq\ 0.5s \\ 40t - 40\ si\ t > 0.5s \end{cases}$$

Una representación gráfica de la ecuación de gobierno planteada en Inventor, se muestra en la figura 6.

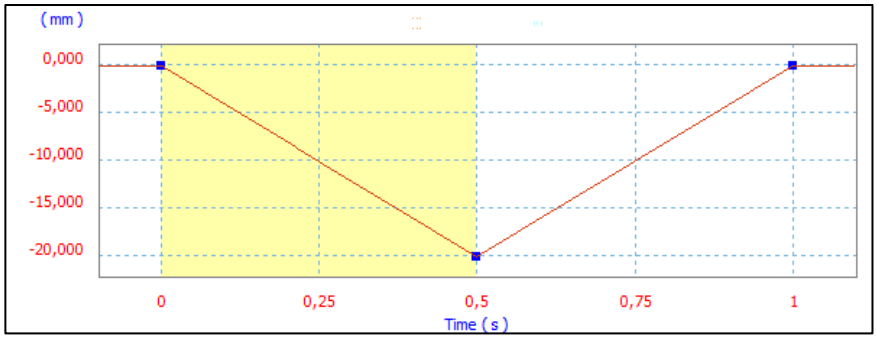

Figura 6. Grafica de la ecuación de gobierno cinemático.

7. Simulación en Inventor

Con todas las condiciones de carga y gobierno cinemático definidas, se procede a ejecutar la simulación dinámica con 100 pasos para 1 segundo de tiempo. Desde la Figura 7 a la Figura 12, se muestran los resultados para las fuerzas resultantes en las Juntas rotacionales. Teniendo en cuenta que el mecanismo tiene simetría con respecto al eje y, estas fuerzas serán iguales en pares de juntas simétricas. Estas graficas evidencias un comportamiento esperado, teniendo en cuenta la gráfica de la ecuación de gobierno, además presentan valores pico muy pequeños, 0.9N para R1, R2, R3, R4, 2.25N para R5, R6, 2.5N para R7, R8, y 2N para R9, R10, R11, R12, cargas equivalentes a máximo unos 250 gramos en el peor de los casos, lo cual representa no debería representar un problema para un tornillo o remache de 2.5mm de diámetro.

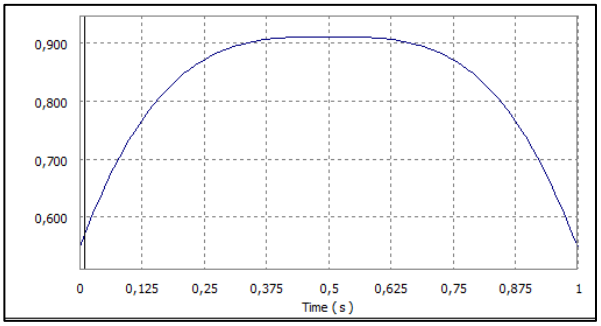

Figura 7: Fuerza total en la Juntas R1 y R2 (Newtons).

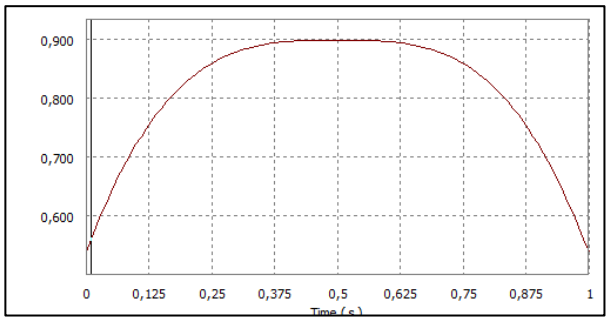

Figura 8: Fuerza total en la Juntas R3 y R4 (Newtons).

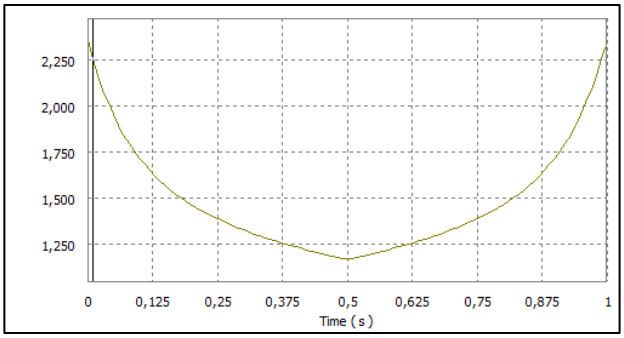

Figura 9: Fuerza total en la Juntas R5 y R6 (Newtons).

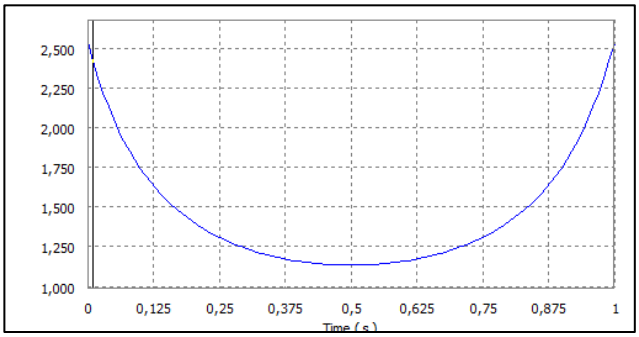

Figura 10: Fuerza total en la Juntas R7 Y R8 (Newtons).

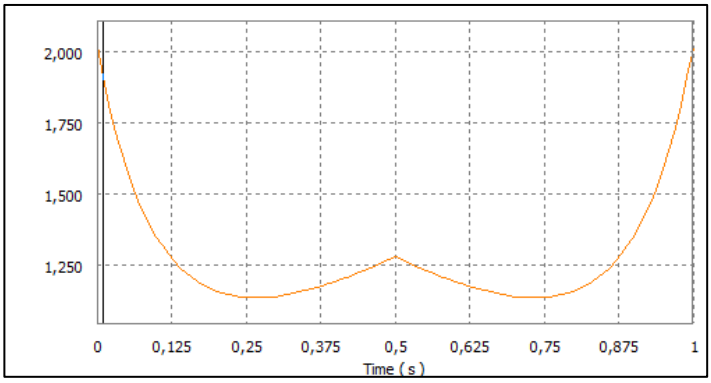

Figura 11: Fuerza total en la Juntas R9 y R10 (Newtons).

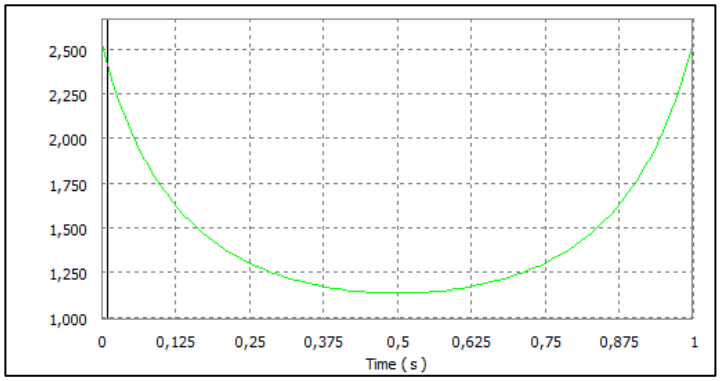

Figura 12 : Fuerza total en la Juntas R11 y R12 (Newtons).

En la Figura 13 se muestra el resultado de la fuerza requerida para ejecutar el movimiento en la junta prismática. Esta fuerza con un pico máximo de 0.5N sería el resultado necesario para la posible selección de un servomotor, ya que seleccionando un eje roscado con un paso por vuelta adecuado para el rango de movimiento de 40 mm de la junta prismática, seria posible calcular el requerimiento máximo para el torque del motor como una función de los 0.5N, sin embargo dicha tarea no entra en el alcance de este proyecto.

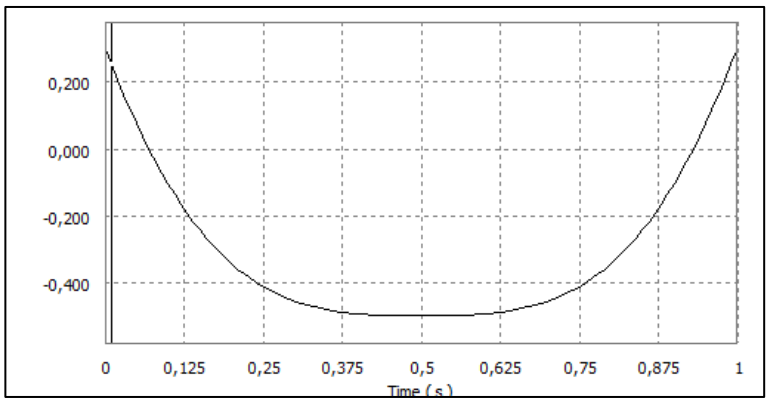

Figura 13 : Fuerza ejercida en la junta prismática (Newtons).

No se presentan las gráficas de la cinemática, puesto que se reservan para una comparación de los resultados con la implementación computacional.

8. Cinemática mediante implementación computacional

En el lenguaje de programación Python con ayuda de las librerías SymPy [3] y NumPy [4], se implementó la formulación computacional para la cinemática del mecanismo basándose en el libro de Shabana [2]. Para ello se construye un sistema compuesto por 30 incógnitas, que corresponden a la posición en el eje x, la posición en el eje y, y la rotación en el eje z, del centro de coordenadas de cada eslabón; y 30 ecuaciones de restricción, donde se tienen 2 ecuaciones por cada junta rotacional para un total de 24, 2 ecuaciones para la junta prismática, 3 ecuaciones para la bancada, y una ecuación de gobierno, para un total de 30 ecuaciones de restricción.

Para las juntas rotacionales, se utilizaron las siguientes dos expresiones generalizadas para dos eslabones i, j que están restringidos entre sí por un junta rotacional Rn, donde los ángulos θ representan la inclinación de cada eslabón, y los valores L_x, L_y representan la posición del centro de coordenadas del eslabón.

$$-L_x^i + L_x^j - \bar{x}_{Rn}^i \cos(\theta^i) + \bar{x}_{Rn}^j \cos(\theta^j) + \bar{y}_{Rn}^i \sin(\theta^i) - \bar{y}_{Rn}^j \sin(\theta^j) = 0$$

$$-L_y^i + L_y^j - \bar{x}_{Rn}^i \sin(\theta^i) + \bar{x}_{Rn}^j \sin(\theta^j) - \bar{y}_{Rn}^i \cos(\theta^i) + \bar{y}_{Rn}^j \cos(\theta^j) = 0$$

Para la junta prismática se utilizaron las dos siguientes expresiones que limitan el ángulo relativo entre el eslabón 5 y la bancada a un valor constante, y fijan la posición en x del eslabón 5, dado que la junta rotacional únicamente ejercerse su movimiento en él eje y.

$$\left(-\theta^{1} + \theta^{5},\ L_{x}^{5} - \overline{x}_{P1}^{1}\right)$$

Mediante SymPy que es una librería de cálculo simbólico para Python, es posible construir un vector de ecuaciones de restricción, como el que se muestra a continuación, donde se tienen las 3 ecuaciones de la bancada en las primeras 3 filas, las 24 ecuaciones de las juntas rotacionales y las 2 ecuaciones de la junta prismática en el centro, así como la ecuación de gobierno en la ultima columna, como una función a trozos.

$$\begin{bmatrix}
L_{x}^{1} \\
L_{y}^{1} \\
\theta^{1} \\
-L_{x}^{1} + L_{x}^{2} - \overline{x}_{R1}^{1}\cos\left(\theta^{1}\right) + \overline{x}_{R1}^{2}\cos\left(\theta^{2}\right) + \overline{y}_{R1}^{1}\sin\left(\theta^{1}\right) - \overline{y}_{R1}^{2}\sin\left(\theta^{2}\right) \\
-L_{y}^{1} + L_{y}^{2} - \overline{x}_{R1}^{1}\sin\left(\theta^{1}\right) + \overline{x}_{R1}^{2}\sin\left(\theta^{2}\right) - \overline{y}_{R1}^{1}\cos\left(\theta^{1}\right) + \overline{y}_{R1}^{2}\cos\left(\theta^{2}\right) \\
-L_{x}^{2} + L_{x}^{3} - \overline{x}_{R2}^{1}\cos\left(\theta^{1}\right) + \overline{x}_{R2}^{3}\cos\left(\theta^{3}\right) + \overline{y}_{R2}^{1}\sin\left(\theta^{1}\right) - \overline{y}_{R2}^{3}\sin\left(\theta^{3}\right) \\
-L_{y}^{2} + L_{y}^{3} - \overline{x}_{R2}^{1}\sin\left(\theta^{1}\right) + \overline{x}_{R2}^{3}\sin\left(\theta^{3}\right) - \overline{y}_{R2}^{1}\cos\left(\theta^{1}\right) + \overline{y}_{R2}^{3}\cos\left(\theta^{3}\right) \\
-L_{x}^{2} + L_{x}^{4} - \overline{x}_{R3}^{2}\cos\left(\theta^{2}\right) + \overline{x}_{R3}^{4}\cos\left(\theta^{4}\right) + \overline{y}_{R3}^{2}\sin\left(\theta^{2}\right) - \overline{y}_{R3}^{4}\sin\left(\theta^{4}\right) \\
-L_{y}^{2} + L_{y}^{4} - \overline{x}_{R3}^{2}\sin\left(\theta^{2}\right) + \overline{x}_{R3}^{4}\sin\left(\theta^{4}\right) - \overline{y}_{R3}^{2}\cos\left(\theta^{2}\right) + \overline{y}_{R3}^{4}\cos\left(\theta^{4}\right) \\
-L_{x}^{3} + L_{x}^{6} - \overline{x}_{R4}^{3}\cos\left(\theta^{3}\right) + \overline{x}_{R4}^{6}\cos\left(\theta^{6}\right) + \overline{y}_{R4}^{3}\sin\left(\theta^{3}\right) - \overline{y}_{R4}^{6}\sin\left(\theta^{6}\right) \\
-L_{y}^{3} + L_{y}^{6} - \overline{x}_{R4}^{3}\sin\left(\theta^{3}\right) + \overline{x}_{R4}^{6}\sin\left(\theta^{6}\right) - \overline{y}_{R4}^{3}\cos\left(\theta^{3}\right) + \overline{y}_{R4}^{6}\cos\left(\theta^{6}\right) \\
-L_{x}^{4} + L_{x}^{5} - \overline{x}_{R5}^{4}\cos\left(\theta^{4}\right) + \overline{x}_{R5}^{5}\cos\left(\theta^{5}\right) + \overline{y}_{R5}^{4}\sin\left(\theta^{4}\right) - \overline{y}_{R5}^{5}\sin\left(\theta^{5}\right) \\
-L_{y}^{4} + L_{y}^{5} - \overline{x}_{R5}^{4}\sin\left(\theta^{4}\right) + \overline{x}_{R5}^{5}\sin\left(\theta^{5}\right) - \overline{y}_{R5}^{4}\cos\left(\theta^{4}\right) + \overline{y}_{R5}^{5}\cos\left(\theta^{5}\right) \\
-L_{x}^{5} + L_{x}^{6} - \overline{x}_{R6}^{5}\cos\left(\theta^{5}\right) + \overline{x}_{R6}^{6}\cos\left(\theta^{6}\right) + \overline{y}_{R6}^{5}\sin\left(\theta^{5}\right) - \overline{y}_{R6}^{6}\sin\left(\theta^{6}\right) \\
-L_{y}^{5} + L_{y}^{6} - \overline{x}_{R6}^{5}\sin\left(\theta^{5}\right) + \overline{x}_{R6}^{6}\sin\left(\theta^{6}\right) - \overline{y}_{R6}^{5}\cos\left(\theta^{5}\right) + \overline{y}_{R6}^{6}\cos\left(\theta^{6}\right) \\
-L_{x}^{5} + L_{x}^{7} - \overline{x}_{R7}^{5}\cos\left(\theta^{5}\right) + \overline{x}_{R7}^{7}\cos\left(\theta^{7}\right) + \overline{y}_{R7}^{5}\sin\left(\theta^{5}\right) - \overline{y}_{R7}^{7}\sin\left(\theta^{7}\right) \\
-L_{y}^{5} + L_{y}^{7} - \overline{x}_{R7}^{5}\sin\left(\theta^{5}\right) + \overline{x}_{R7}^{7}\sin\left(\theta^{7}\right) - \overline{y}_{R7}^{5}\cos\left(\theta^{5}\right) + \overline{y}_{R7}^{7}\cos\left(\theta^{7}\right) \\
-L_{x}^{5} + L_{x}^{8} - \overline{x}_{R8}^{5}\cos\left(\theta^{5}\right) + \overline{x}_{R8}^{8}\cos\left(\theta^{8}\right) + \overline{y}_{R8}^{5}\sin\left(\theta^{5}\right) - \overline{y}_{R8}^{8}\sin\left(\theta^{8}\right) \\
-L_{y}^{5} + L_{y}^{8} - \overline{x}_{R8}^{5}\sin\left(\theta^{5}\right) + \overline{x}_{R8}^{8}\sin\left(\theta^{8}\right) - \overline{y}_{R8}^{5}\cos\left(\theta^{5}\right) + \overline{y}_{R8}^{8}\cos\left(\theta^{8}\right) \\
-L_{x}^{4} + L_{x}^{9} - \overline{x}_{R9}^{4}\cos\left(\theta^{4}\right) + \overline{x}_{R9}^{9}\cos\left(\theta^{9}\right) + \overline{y}_{R9}^{4}\sin\left(\theta^{4}\right) - \overline{y}_{R9}^{9}\sin\left(\theta^{9}\right) \\
-L_{y}^{4} + L_{y}^{9} - \overline{x}_{R9}^{4}\sin\left(\theta^{4}\right) + \overline{x}_{R9}^{9}\sin\left(\theta^{9}\right) - \overline{y}_{R9}^{4}\cos\left(\theta^{4}\right) + \overline{y}_{R9}^{9}\cos\left(\theta^{9}\right) \\
L_{x}^{10} - L_{x}^{6} - \overline{x}_{R10}^{6}\cos\left(\theta^{6}\right) + \overline{x}_{R10}^{10}\cos\left(\theta^{10}\right) + \overline{y}_{R10}^{6}\sin\left(\theta^{6}\right) - \overline{y}_{R10}^{10}\sin\left(\theta^{10}\right) \\
L_{y}^{10} - L_{y}^{6} - \overline{x}_{R10}^{6}\sin\left(\theta^{6}\right) + \overline{x}_{R10}^{10}\sin\left(\theta^{10}\right) - \overline{y}_{R10}^{6}\cos\left(\theta^{6}\right) + \overline{y}_{R10}^{10}\cos\left(\theta^{10}\right) \\
-L_{x}^{7} + L_{x}^{9} - \overline{x}_{R11}^{7}\cos\left(\theta^{7}\right) + \overline{x}_{R11}^{9}\cos\left(\theta^{9}\right) + \overline{y}_{R11}^{7}\sin\left(\theta^{7}\right) - \overline{y}_{R11}^{9}\sin\left(\theta^{9}\right) \\
-L_{y}^{7} + L_{y}^{9} - \overline{x}_{R11}^{7}\sin\left(\theta^{7}\right) + \overline{x}_{R11}^{9}\sin\left(\theta^{9}\right) - \overline{y}_{R11}^{7}\cos\left(\theta^{7}\right) + \overline{y}_{R11}^{9}\cos\left(\theta^{9}\right) \\
L_{x}^{10} - L_{x}^{8} - \overline{x}_{R12}^{8}\cos\left(\theta^{8}\right) + \overline{x}_{R12}^{10}\cos\left(\theta^{10}\right) + \overline{y}_{R12}^{8}\sin\left(\theta^{8}\right) - \overline{y}_{R12}^{10}\sin\left(\theta^{10}\right) \\
L_{y}^{10} - L_{y}^{8} - \overline{x}_{R12}^{8}\sin\left(\theta^{8}\right) + \overline{x}_{R12}^{10}\sin\left(\theta^{10}\right) - \overline{y}_{R12}^{8}\cos\left(\theta^{8}\right) + \overline{y}_{R12}^{10}\cos\left(\theta^{10}\right) \\
-\theta^{1} + \theta^{5} \\
L_{x}^{5} - \overline{x}_{P1}^{1} \\
\begin{cases} -L_{y}^{5} - \overline{y}_{P1}^{5} - 40t & \text{for } t < 0.5 \\ -L_{y}^{5} - \overline{y}_{P1}^{5} + 40t - 40 & \text{otherwise} \end{cases}
\end{bmatrix}$$

A partir de allí, es posible construir todas las matrices y vectores necesarios para el calculo completo de la cinemática, incluyendo posición, velocidad y aceleración, mediante las funciones matemáticas de SymPy. Estos vectores y matrices tales como la matriz jacobiana del vector de restricción, o la jacobiana del producto de la primer matriz jacobiana por el vector de velocidades, no se incluyen en el documento, debido a su considerable tamaño de

30 x 30 expresiones algebraicas, sin embargo pueden ser consultadas en el archivo de Jupyter disponible en el enlace de la sección 9 del documento.

Una vez que se construyen todos los vectores y matrices necesarios, se realiza un remplazo de los valores numéricos que no cambian en el tiempo, tales como las coordenadas relativas de las juntas con respecto a los eslabones; de esta manera únicamente se conservan las variables simbólicas necesarias durante la ejecución del algoritmo, dado que su valor cambia en función de la iteración.

Para ejecutar el algoritmo tal y como lo especifica el libro de Shabana, se selecciona un limite del error admisible (en este caso 1e-8) y un limite de iteraciones por cada instante de tiempo (50 iteraciones), además se ejecutan 100 iteraciones para un segundo de las misma manera en que se hiso para la simulación dinámica en Inventor. A partir de estos datos, se ejecuta el algoritmo de Newton Raphson para la solución de la posición, y se ejecuta la solución de ecuaciones lineales para la velocidad y la aceleración, esto se logra mediante la librería de NumPy, remplazando en cada iteración los valores de las variables simbólicas de las matrices originales, y luego convirtiéndolas en arreglos numéricos de NumPy que permiten la ejecución de operaciones sobre matrices a gran velocidad.

La ejecución del algoritmo tardo alrededor de 5 minutos para un equipo con un procesador de 6 núcleos, 12 hilos, 32 MB de cache y 32 GB de memoria RAM a 3200Mhz, utilizando alrededor de un 20% de los recursos disponibles a lo largo de la ejecución, en comparación a los 10-15 segundos que tarda la simulación dinámica en Inventor, lo cual refleja un costo computacional bastante alto, probablemente debido al uso de un lenguaje relativamente lento como puede ser Python, y al uso de matrices con variables simbólicas mediante la librería SymPy.

En las Figuras 14, 15 y 16 se muestra una comparación de los resultados de Inventor y Python para la posición, velocidad y aceleración para las coordenadas x, y del eslabón 9 que corresponde a la mordaza izquierda, y al ángulo del eslabón 2, ya que se considera que si estos valores son correctos, entonces la implementación resulto en valores correctos para las 90 variables calculadas, 30 de posición, 30 de velocidad y 30 de aceleración.

En las Figuras 14, 15, 16 se grafica en el costado izquierdo una superposición de los resultados de Inventor en color amarillo, y los resultados de Python en color azul, además se muestra en el costado derecho en color rojo, el error relativo entre ambos resultados. Es posible evidenciar que para las 9 variables mostradas se presenta un coincidencia casi perfecta entre ambos métodos de simulación, con errores relativos muy pequeños. De igual forma llama la atención el resultado de la velocidad en x para el eslabón 9, donde se evidencia un salto instantáneo de velocidad justo cuando se realiza el cambio de dirección en la ecuación de gobierno a los 0.5 segundos, lo cual técnicamente no es posible, en otras palabras los resultados de esta simulación deben tomarse como dos situaciones distintas simuladas en un único intervalo de tiempo; este problema podría solucionarse utilizando una ecuación de gobierno que imponga

un perfil de velocidad trapezoidal, el cual permite reflejar mejor el comportamiento real del mecanismo, integrando los posibles límites de velocidad y aceleración de un motor en el perfil de movimiento, siendo también conveniente para el calculo de la fuerza que impone el gobierno cinemático, ya que permite reflejar mejor los valores máximos para un posible cálculo de torque de un motor, a partir de una aceleración máxima requerida.

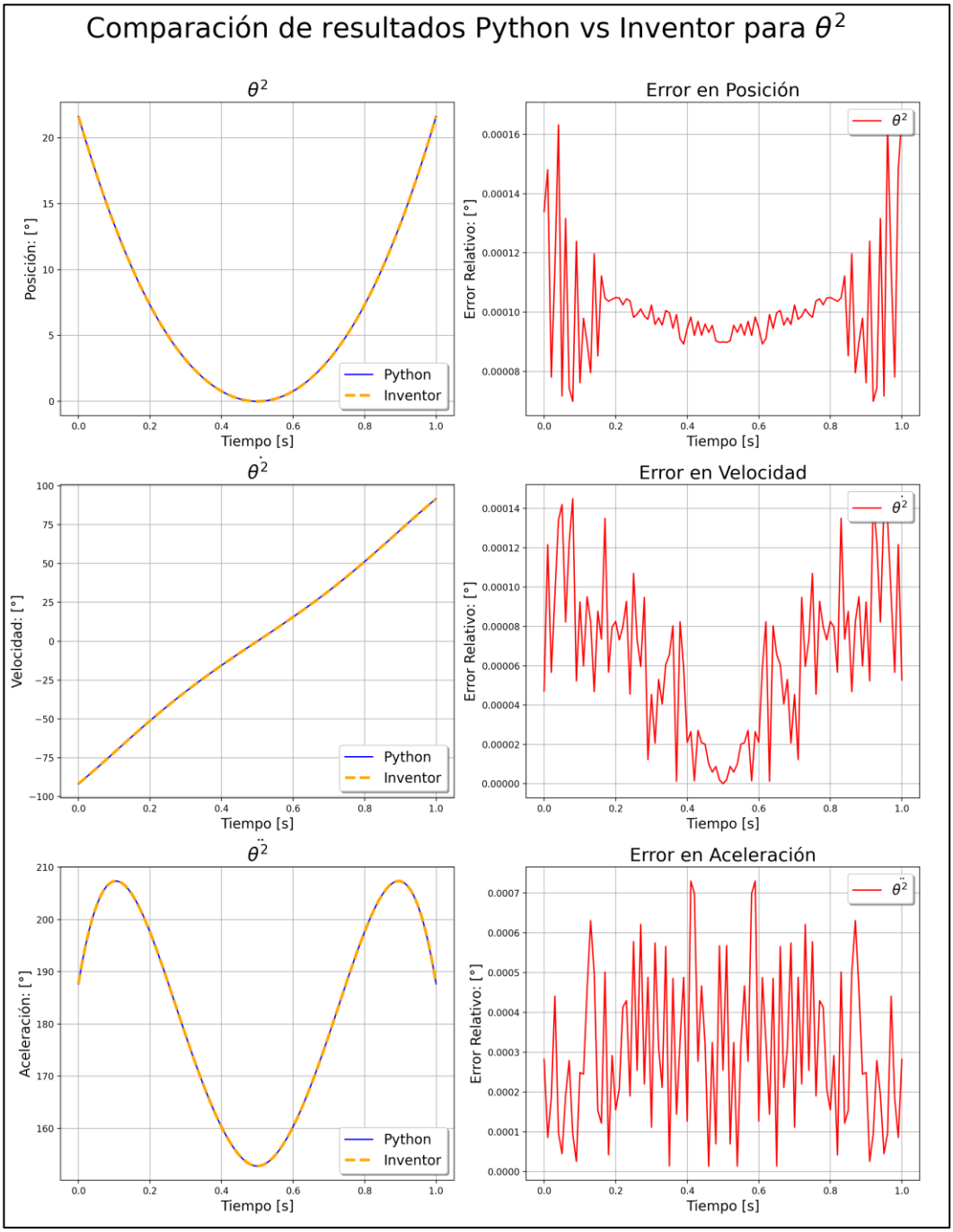

Figura 14: Angulo del Eslabón 2, Python vs Inventor.

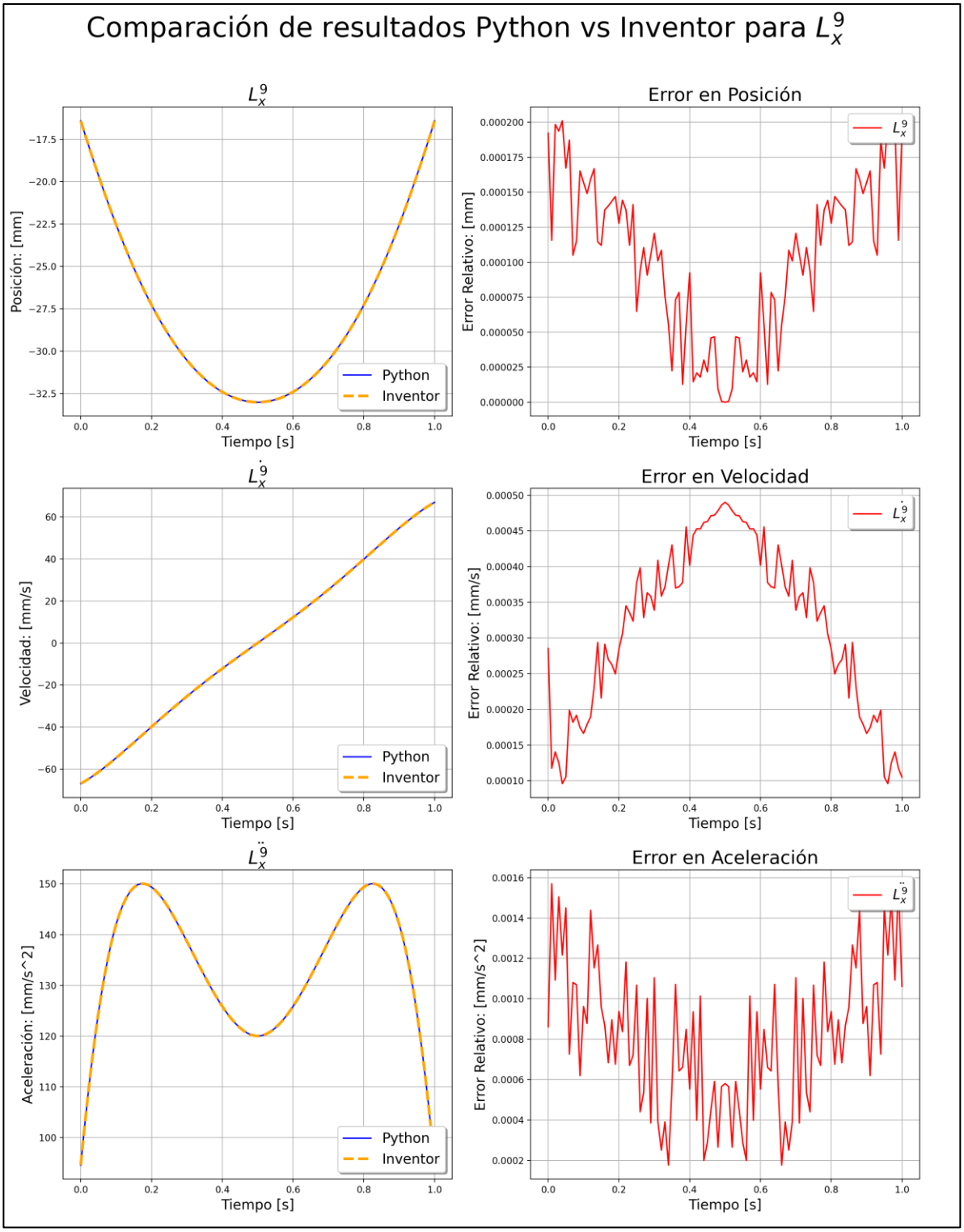

Figura 15: Posición en x del Eslabón 9 (Mordaza izquierda), Python vs Inventor.

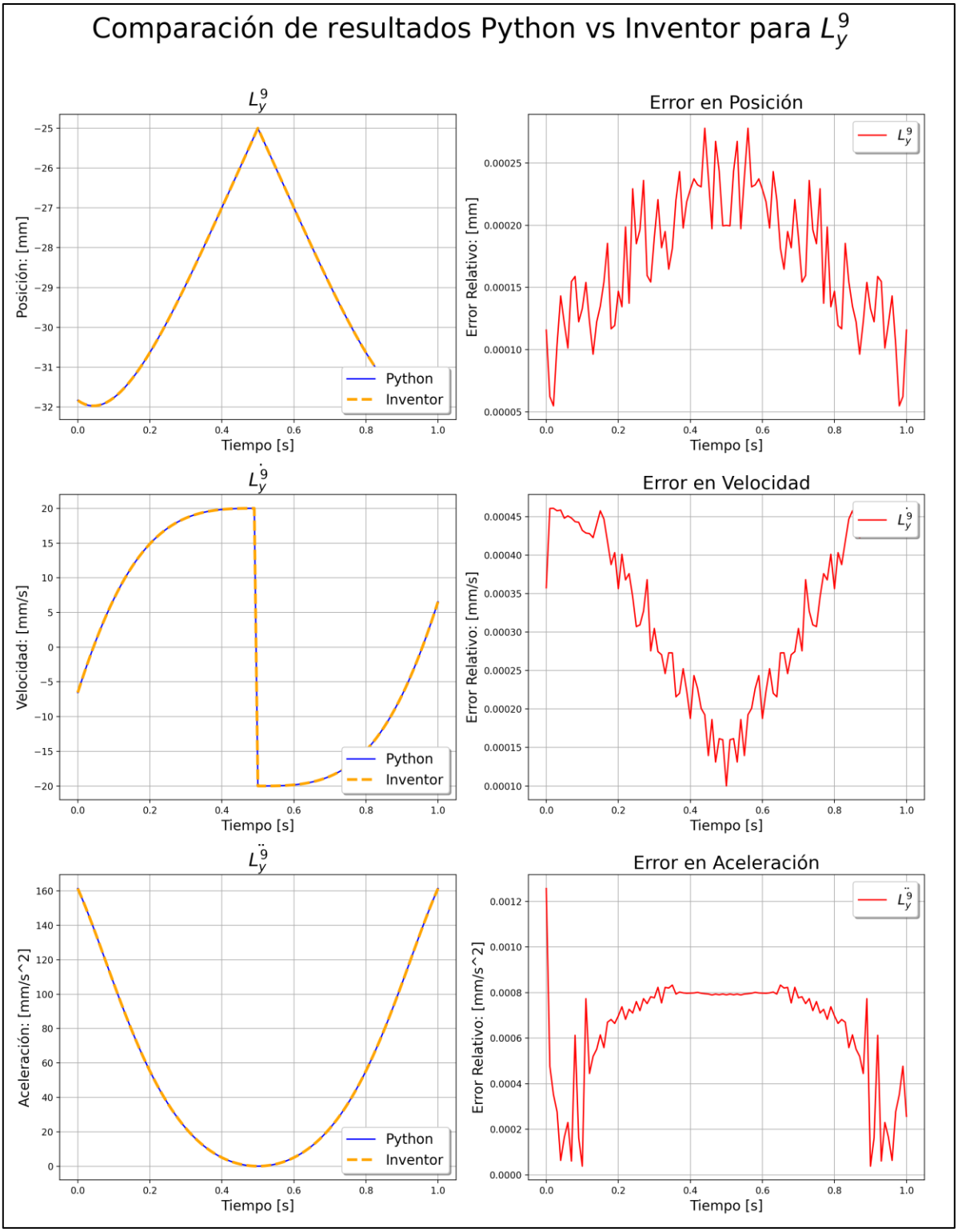

Figura 15: Posición en y del Eslabón 9 (Mordaza izquierda), Python vs Inventor.

9. Código

El código de la implementación computacional en Python, se encuentra disponible en un archivo de Jupyter en Google Colab mediante este enlace y en GitHub mediante este enlace.

10. Conclusiones

- El método computacional presenta excelentes resultados con error prácticamente nulo, sin embargo, para un uso práctico se requiere la construcción de una librería sobre un lenguaje de programación más rápido como C/C++ o Rust, dado que la simulación tardo bastante tiempo en ejecutarse, en comparación con el software de Autodesk Inventor. (5m vs 10s).
- La noción de rango de la matriz puede ser una herramienta muy útil para verificar el correcto planteamiento de las ecuaciones para una simulación cinemática computacional, pues permite verificar que todos los términos de la jacobiana sean linealmente independientes, requisito para poder utilizar el algoritmo de Newton-Raphson. Esta noción fue fundamental para solucionar problemas durante el proceso de desarrollo del código puesto que permitió identificar un error al encontrar un rango de 29, menor al rango de 30 esperado, lo que permitió reconocer un error en la definición de la ecuación de gobierno.
- Es posible mejorar los resultados de la simulación utilizando una ecuación de gobierno con un perfil trapezoidal para la velocidad, que refleje la limitación en la aceleración de un motor real, evitando cambios instantáneos de velocidad y obteniendo valores de torque más precisos.

11. Referencias

[1] M. Numan, "Robotic Gripper Save and Make Design" 2022, [En línea]. Disponible en: https://www.youtube.com/shorts/Go2qCykVICA

[2] A. A. Shabana, "Computational Dynamics," 3rd ed. John Wiley & Sons, 2009.

[3] Meurer, A., et al. "SymPy: symbolic computing in Python," in PeerJ Computer Science, vol. 3, pp. e103, 2017.

[4] Charles R. Harris, et al. "Array programming with NumPy," in Nature, vol. 585, no. 7825, pp. 357–362, 2020.