Actividad de aprendizaje autónomo 1: Mecanismo Manivela-Corredera

Dinámica Computacional de Sistemas Multicuerpo

Camilo Andres Vera Ruiz

caverar@unal.edu.co

A continuación se plantea el desarrollo de la formulación computacional para el cálculo de posición, velocidad y aceleración de todos los eslabones de un mecanismo de manivela-corredera compuesto por 4 eslabones, una bancada, una manivela con un movimiento de rotación constante, un acoplador, y una corredera; así como una comparación de los resultados del entorno de simulación dinámica del software Autodesk Inventor Professional 2024.

Se parte del diagrama cinemático de la figura 1, donde se busca determinar la variación del ángulo θ_3 a partir de una velocidad constante en el ángulo θ_2. Para ello se proporcionan las dimensiones de las longitudes a, b, c, un valor constante para ω_2 y un ángulo inicial de la manivela θ_{2_0}. En la tabla 1 se presentan los valores de cada parámetro.

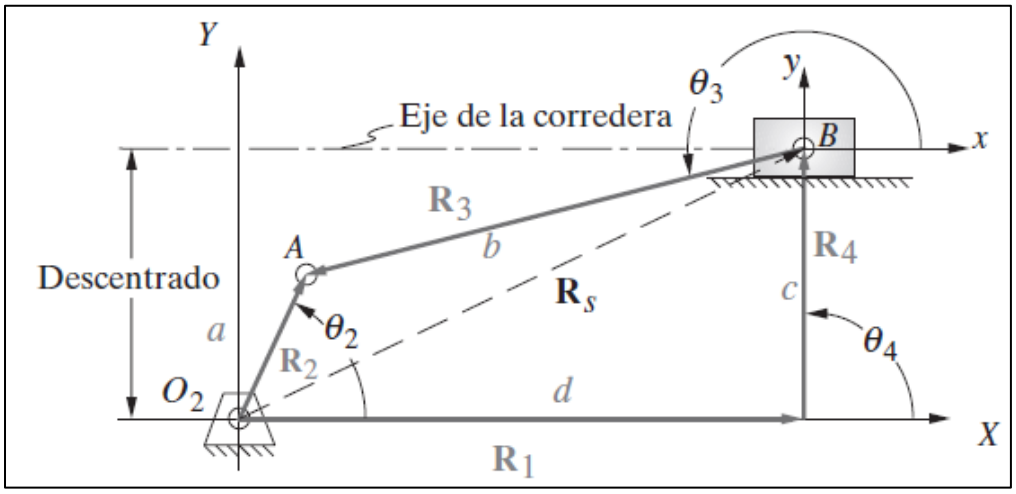

Figura 1: Diagrama cinemático del mecanismo, Tomado del enunciado del problema

Tabla 1: Parámetros del Mecanismo

Parámetro	Valor
a	$20mm$
b	$65mm$
c	$25mm$
ω_2	$2\pi \frac{rad}{s}$
θ_{2_0}	$0\,rad$

A partir de allí se plantea el desarrollo de las ecuaciones de las restricciones de cada junta, y las ecuaciones de gobierno del mecanismo, siguiendo la notación del libro de Shabana [1] y teniendo en cuenta que el ángulo θ_3 presentado en el diagrama, no se corresponde exactamente con el ángulo θ^3 de la formulación computacional, el cual corresponde al ángulo de inclinación del eslabón 3 con respecto al centro de coordenadas global ubicado en el punto O_2 de la figura 1, aclarando que ambos difieren en exactamente 180°, siendo el primero mayor que el segundo.

- Junta Fija: Ecuaciones de restricción de la bancada al sistema de coordenadas global.

$$R_x^1 = 0$$
$$R_y^1 = 0$$
$$\theta^1 = 0$$

- Junta Rotacional (O_2): Dado que el eslabón 2 se encuentra fijo a la bancada en un de sus extremos, la posición de su centro de masa únicamente esta determinada por el ángulo de inclinación.

$$R_x^2 = \frac{a}{2}\cos(\theta^2)$$
$$R_y^2 = \frac{a}{2}\sin(\theta^2)$$

- Junta Rotacional (A): En esta junta se conectan los eslabones 2 y 3, y la posición del centro de masa del eslabón 3 está determinada por su propia inclinación, así como por la inclinación y la posición del centro de masa del eslabón 2.

$$R_x^3 = R_x^2 + \frac{a}{2}\cos(\theta^2) + \frac{b}{2}\cos(\theta^3)$$
$$R_y^3 = R_y^2 + \frac{a}{2}\sin(\theta^2) + \frac{b}{2}\sin(\theta^3)$$

  En la ecuación anterior es fundamental la distinción entre θ_3 y θ^3 debido a que la formulación computacional solo es válida para θ^3 que es el ángulo medido en el centro de masa del eslabón, dado que durante el desarrollo de la formulación computacional, se identificó que no se obtenían los resultados esperados si seleccionaba θ_3 en vez de θ^3.

- Junta Rotacional (B): Esta junta conecta al eslabón 3 que rota, con el eslabón 4 que no rota, por lo cual la posición del centro de masa del eslabón 4, solo esta determinada por la posición del centro de masa del eslabón 3 y su inclinación

$$R_x^4 = R_x^3 + \frac{b}{2}\cos(\theta^3)$$
$$R_y^4 = R_y^3 + \frac{b}{2}\sin(\theta^3)$$

- Junta Prismática (C): Siguiendo las ecuaciones de restricción para una junta prismática definidas en el libro de Shabana [1], la primera expresión define un ángulo relativo entre el eslabón 4 y la bancada, de un valor constante igual a cero, mientras que la segunda expresión define al vector perpendicular al movimiento de la junta, y dado que el vector de movimiento de la junta es coincidente con el vector R_x^4, basta con definir R_y^4 con un valor constante para que el movimiento de la junta solo se de en el vector R_x^4.

$$\theta^4 - \theta^1 = 0$$
$$R_y^4 = c$$

- Ecuación de gobierno para R_2: Finalmente para la ecuación de gobierno, solo se requiere una ecuación de posición angular para velocidad constante, sobre el eslabón 2.

$$\theta^2 = \omega^2(t) + \theta_0^2 = \dot{\theta}^2(t) + \theta_0^2$$

De las expresiones anteriores, se tienen 12 ecuaciones, lo cual es suficiente para construir el vector de restricciones $C(q,t)$ así como el vector de incógnitas q.

$$q = \begin{bmatrix} R_x^1 \\ R_y^1 \\ \theta^1 \\ R_x^2 \\ R_y^2 \\ \theta^2 \\ R_x^3 \\ R_y^3 \\ \theta^3 \\ R_x^4 \\ R_y^4 \\ \theta^4 \end{bmatrix}, \qquad C(q,t) = \begin{bmatrix} R_x^1 \\ R_y^1 \\ \theta^1 \\ R_x^2 - \frac{a}{2}\cos(\theta^2) \\ R_y^2 - \frac{a}{2}\sin(\theta^2) \\ -R_x^3 + R_x^2 + \frac{a}{2}\cos(\theta^2) + \frac{b}{2}\cos(\theta^3) \\ -R_y^3 + R_y^2 + \frac{a}{2}\sin(\theta^2) + \frac{b}{2}\sin(\theta^3) \\ R_x^4 - R_x^3 - \frac{b}{2}\cos(\theta^3) \\ R_y^4 - R_y^3 - \frac{b}{2}\sin(\theta^3) \\ \theta^4 - \theta^1 \\ R_y^4 - c \\ \theta^2 - \dot{\theta}^2(t) - \theta_0^2 \end{bmatrix}$$

A partir de allí se procede a realizar la implementación en código, en este caso se utilizó el lenguaje de programación Python, junto con las librerías SymPy [2] para el cálculo simbólico, y NumPy [3] para el cálculo numérico. Mediante SymPy se calculan las expresiones analíticas para la jacobiana C_q, el vector C_t para el cálculo de velocidad, y el vector Q_d para el cálculo de aceleraciones.

$$C_q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \frac{a\sin(\theta^2)}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{a\cos(\theta^2)}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{a\sin(\theta^2)}{2} & -1 & 0 & -\frac{b\sin(\theta^3)}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{a\cos(\theta^2)}{2} & 0 & -1 & \frac{b\cos(\theta^3)}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & \frac{b\sin(\theta^3)}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -\frac{b\cos(\theta^3)}{2} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, C_t = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\omega^2 \end{bmatrix},$$

$$Q_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\frac{\dot{(\theta^2)}^2 a\cos(\theta^2)}{2} \\ -\frac{\dot{(\theta^2)}^2 a\sin(\theta^2)}{2} \\ \frac{\dot{(\theta^2)}^2 a\cos(\theta^2)}{2} + \frac{\dot{(\theta^3)}^2 b\cos(\theta^3)}{2} \\ \frac{\dot{(\theta^2)}^2 a\sin(\theta^2)}{2} + \frac{\dot{(\theta^3)}^2 b\sin(\theta^3)}{2} \\ -\frac{\dot{(\theta^3)}^2 b\cos(\theta^3)}{2} \\ -\frac{\dot{(\theta^3)}^2 b\sin(\theta^3)}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

C_q fue calculado mediante la función "jacobian" sobre el vector de restricciones C, derivando con respecto al vector q. C_t se calculó mediante la función "diff" que calcula la derivada escalar, en este caso con respecto a t. Q_d es el resultado de la siguiente expresión, donde las derivadas parciales con respecto a q son operaciones de cálculo de matrices jacobianas y las derivadas parciales con respecto a t son operaciones de derivadas con respecto a un escalar.

$$Q_d = -\left(\left(\frac{\delta}{\delta q}\left(C_q \cdot \frac{\delta}{\delta t} q\right)\right) \cdot \frac{\delta}{\delta t} q\right) - \left(2 \cdot \frac{\delta}{\delta t}(C_q) \cdot \frac{\delta}{\delta t} q\right) - \frac{\delta^2}{\delta t^2} C$$

Con estos vectores y matrices construidos, se remplazan los parámetros invariantes en el tiempo, tales como las longitudes a, b, c, la velocidad angular del eslabón 2 ($\dot{\theta^2}$) y la posición inicial de θ^2 con el fin de minimizar el tiempo requerido en código para remplazar estos parámetros en cada iteración.

A partir de allí se plantea el código para el algoritmo de Newton-Raphson, con los parámetros que se muestran en la tabla 2:

Tabla 2: Parámetros del Algoritmo

Parámetro	Descripción	Valor
max_iter	Número máximo de iteraciones por posición	10
max_error	Error máximo (norma de C)	$1 \cdot 10^{-8}$
$steps$	Numero de posiciones intermedias del mecanismo	200
$time$	Tiempo total de la simulación	$1s$

Mediante un ciclo for y un ciclo while anidados, se realiza la iteración sobre las 200 configuraciones del mecanismo, y para cada configuración inicialmente se ejecuta el algoritmo de Newton-Raphson para el cálculo de la posición, mediante la ecuación $C_q(q,t)\ \Delta_q = -C(t)$, iterando hasta alcanzar el error máximo admisible "max_$error$" para la norma del vector C, a continuación se calcula la velocidad y la aceleración de forma directa, mediante las expresiones $C_q(q,t) \cdot \dot{q} = -C_t(t)$ y $C_q(q,t) \cdot \ddot{q} = Q_d(q,\dot{q})$. De allí se obtienen 3 vectores de resultados $q, \dot{q}, \ddot{q}$ para cada instante del tiempo, donde se encuentran los valores de posición, velocidad y aceleración de cada eslabón, con las magnitudes angulares en radianes. Todas las operaciones matriciales, se ejecutan convirtiendo las matrices simbólicas definidas anteriormente, en arreglos numéricos de la biblioteca NumPy, remplazando los valores de los parámetros $t, q, \dot{q}$ en cada iteración.

Para comparar los resultados, se realiza una simulación dinámica en Autodesk Inventor y se exportan los resultados a un archivo .csv que puede ser fácilmente cargado por Python, para graficar los resultados mediante la librería Matplotlib, teniendo en cuenta que los resultados de los ángulos en Inventor están en grados, mientras que los resultados del método computacional en Python están en radianes, por lo cual se realiza la pertinente conversión a grados.

En la figura 2 se muestran los resultados, tanto para el método computacional implementado en Python, marcados en color azul, y los resultados de la simulación dinámica en Inventor, mercados en amarillo, se grafican los valores de posición, velocidad, y aceleración para el ángulo de inclinación del eslabón 3 (θ^3) y la posición en el eje x del eslabón 4 (R_x^4), adicionalmente en la tercera columna se grafica el error absoluto de los resultados de Python con respecto a los resultados de Inventor en valores porcentuales.

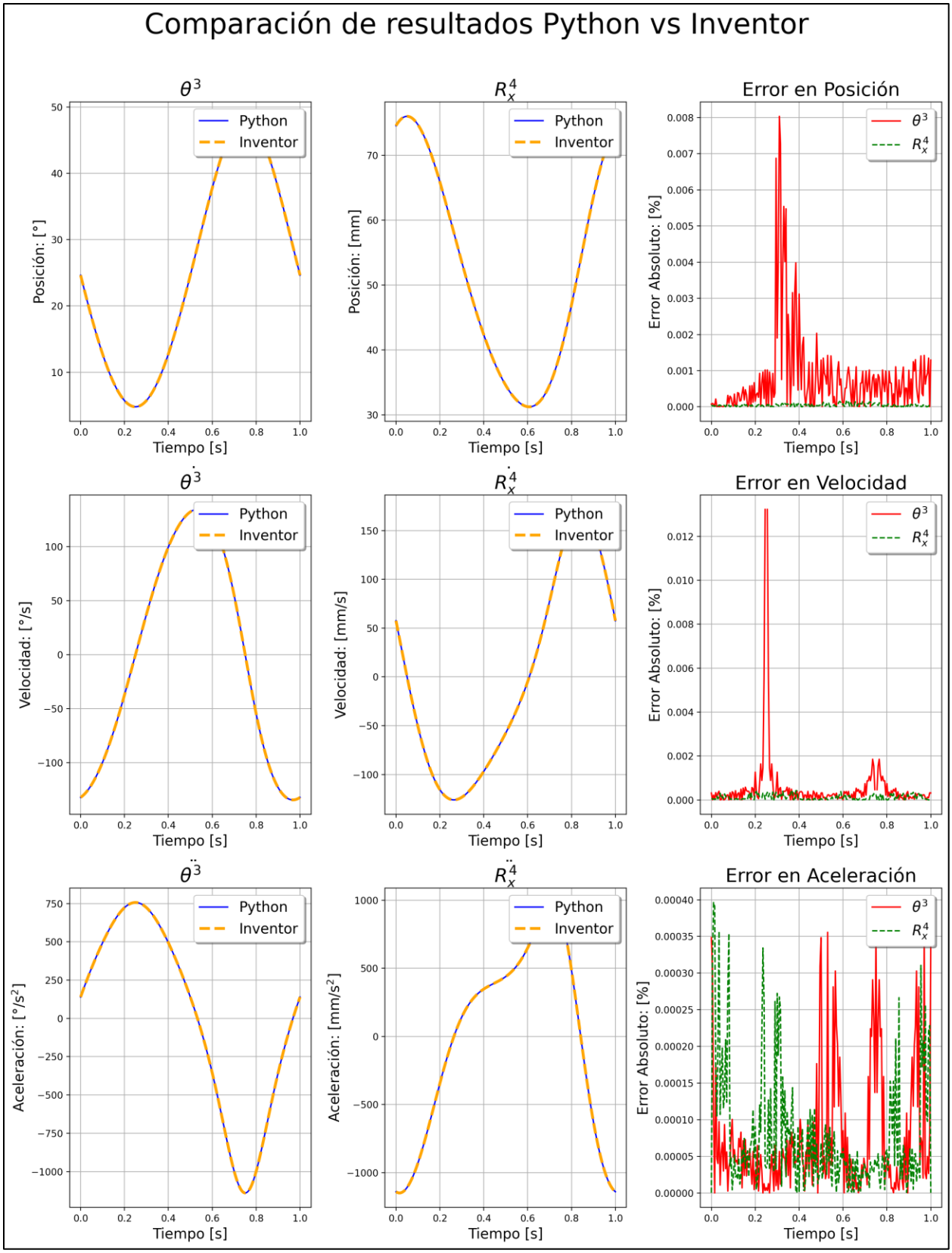

Figura 2: Resultados de la simulación y comparación con Inventor.

Conclusiones:

En la figura 2 se puede observar que los resultados de la simulación fueron satisfactorios, con una coincidencia casi perfecta entre los resultados de Inventor y los resultados del algoritmo en Python, se evidencia un error absoluto que no supera el 0.014% y que se puede atribuir a la cantidad de decimales tomados de los resultados de inventor a la hora de exportar al archivo .csv; al proceso de conversión de radianes a grados; y al error proveniente del uso del tipo de dato flotante, que no es capaz de representar todos los números que abarca, en los 64bits que tiene en el caso de Python y en la cantidad de bits desconocida que utiliza Inventor para representar los valores de q.

Anexos:

El código esta disponible como adjunto en .zip a este documento, allí se incluye un archivo. ipynb con el código, y un .csv con los resultados de Inventor. De igual forma se encuentra disponible en GitHub únicamente para su visualización, y en un documento público de Google Colab donde se puede ejecutar sin ningún problema, siempre que se aceda con una cuenta de Google con dominio @unal.edu.co.

Referencias

[1] A. A. Shabana, "Computational Dynamics," 3rd ed. John Wiley & Sons, 2009.
[2] Meurer, A., et al. "SymPy: symbolic computing in Python," in PeerJ Computer Science, vol. 3, pp. e103, 2017.
[3] Charles R. Harris, et al. "Array programming with NumPy," in Nature, vol. 585, no. 7825, pp. 357–362, 2020.