UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA MECÁNICA Y MECATRÓNICA

MÉTODOS NUMÉRICOS

GRUPO 2

2021-1

Aplicación del método de Newton Raphson para describir la cinemática de un mecanismo

Autores:
Andrés Holguín R.
Julián A. Caipa P.
Santiago Marín B.

Docente:
Ing. Oswaldo
Rojas Camacho

Índice

1. Objetivos

- Resolver un problema aplicado a la ingeniería implementando los conceptos adquiridos en el curso de métodos numéricos.
- Realizar el proceso de resolución de mecanismos mediante el uso recurrente de métodos numéricos como procedimiento de solución del problema con el fin de optimizar la precisión de la solución y de los recursos computacionales asociados al mismo.
- Implementar el método de Newton Raphson para lograr determinar los perfiles cinemáticos del mecanismo a partir de un movimiento ejercido por el eje de un motor.

2. Introducción y planteamiento del problema

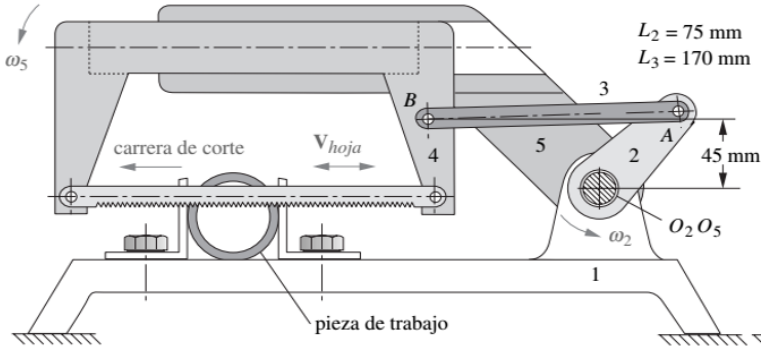


Figura 1: Mecanismo de trabajo tipo sierra eléctrica.

Se tiene un mecanismo de tipo sierra eléctrica, utilizado para cortar barras de metal. Este trabaja mediante transmisión de movimiento radial a lineal a través de la conexión de sus diferentes eslabones. Visto en la figura 1, el movimiento radial empieza con el eje en el punto 1, y termina en movimiento lineal en 4, permitiendo el corte de la pieza. Se va a determinar el análisis cinemático completo del mecanismo a partir del análisis de su cadena cinemática, logrando describir la posición, velocidad y aceleración de sus componentes para cualquier condición (1). Esto es necesario para lograr realizar un análisis cinético, de resistencia y a fatiga, sin embargo, estos conceptos no se abarcan en este informe. De este modo, se describen los vectores y ángulos del sistema:

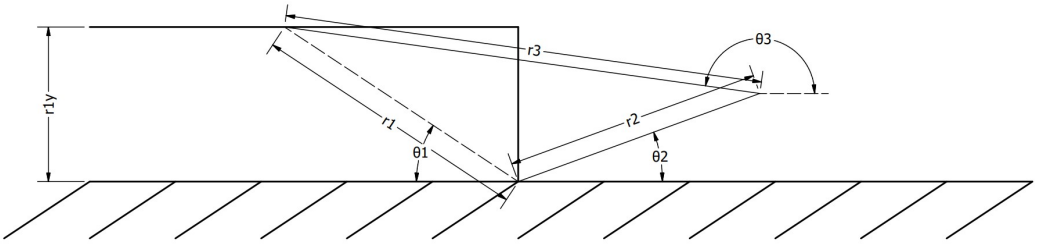


Figura 2: Diagrama del sistema.

Donde:

- $\vec{r_1}$: Eslabón formado por $\vec{r_2} + \vec{r_3}$.
- V_r: Velocidad lineal de $\vec{r_1}$.
- A_r: Aceleración lineal de $\vec{r_1}$.
- θ_1: Ángulo de $\vec{r_1}$ respecto a la horizontal.
- ω_1: Velocidad angular de θ_1.
- α_1: Aceleración angular de θ_1.
- θ_3: Ángulo de $\vec{r_3}$ respecto a la horizontal.
- ω_3: Velocidad angular de θ_3.
- α_3: Aceleración angular de θ_3.

3. Marco teórico

3.1. Método de Newton Raphson

3.1.1. Método de Newton Raphson en una variable real

El método de Newton Raphson es un método numérico abierto algorítmico que permite el calculo de raíces, inicialmente planteado para una función real, pero extrapolado, para una función en el dominio de los complejos, o un sistema de ecuaciones lineales o no lineales. Este método tiene diversas aplicaciones en el campo de la computación y en general, en cualquier campo de aplicación matemática de problemas de solución de raíces, máximos y mínimos, optimización, ceros de funciones notables, entre otros (2).

Inicialmente el método tiene la definición más sencilla en una función real f y una aproximación inicial a la raíz x_0, llamado valor semilla, y que está considerablemente cerca a la raíz que se espera encontrar, para posteriormente aproximar la función utilizando su recta tangente en dicho punto utilizando cálculo diferencial, encontrando el intercepto de dicha línea con el eje x. Dicho intercepto será una mejor aproximación de la raíz que el supuesto inicial, por lo cual es posible iterar este procedimiento hasta encontrar el valor que aproxime de mejor forma a la raíz con la tolerancia de error definida para la aproximación. Formalmente el método se puede definir de la siguiente forma (2):

Sea f de dominio y rango real una función diferenciable en el intervalo (a, b), y una aproximación a la raíz de dicha función x_n, se puede encontrar una mejor aproximación x_{n+1} que se puede deducir desde la recta tangente a la curva dada como:

$$y = f'(x_n)(x - x_n) + f(x_n)$$

Evaluando para $(x,\ y) = (x_{n+1},\ 0)$:

$$0 = f'(x_n)(x_{n+1} - x_n) + f(x_n)$$

Despejando para x_{n+1}:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

A partir de esta expresión se realiza el proceso iterativo de determinar x_{n+1}, sin embargo, se tienen ciertas condiciones de convergencia a tener en cuenta:

- El valor semilla x_0 debe estar relativamente cerca a la raíz.
- El valor semilla x_0 no puede ser estacionario, esto quiere decir que su derivada no puede ser cero, ya que al evaluar en el algoritmo, se genera una división por cero.
- La derivada debe ser continua en el intervalo donde se encuentra la raíz.

Tratándose de un método abierto, el error absoluto y relativo asociado al NR se define como:

$$\varepsilon_{abs} = |x_{n+1} - x_n| \qquad \varepsilon_{rel} = \frac{|x_{n+1} - x_n|}{|x_{n+1}|} \tag{1}$$

3.1.2. Extrapolación del método a varias variables

En el caso de sistemas de ecuaciones, el método emplea el uso de derivadas parciales para hacer el mismo algoritmo iterativo de aproximación mediante tangentes. La aproximación se realiza variable

a variable de forma simultanea, por lo cual es necesario todas las derivadas parciales del sistema de ecuaciones (3).

Dicho esto, se tiene un sistema de ecuaciones lineal o no lineal compatible determinado, descrito por el vector columna de funciones $F(x)$, diferenciable en $F : \mathbb{R}^k \to \mathbb{R}^k$, donde x es el vector que contiene las variables del sistema, x_n es el vector columna con los valores actuales de la iteración de las variables, y x_{n+1} el vector columna que contiene los valores de la siguiente iteración. Para las derivadas, se aplica el concepto del Jacobiano del sistema, matriz que aplica un operador diferencial que permite derivar cada función respecto a cada variable y se obtienen los resultados en dicha matriz definida como:

$$J_F(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_k} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_k(x)}{\partial x_1} & \frac{\partial f_k(x)}{\partial x_2} & \cdots & \frac{\partial f_k(x)}{\partial x_k} \end{pmatrix}$$

Ahora bien, aplicado al método, es necesario dividir por las derivadas en el algoritmo iterativo, por lo que se implementa la matriz inversa del Jacobiano, es decir, $J_F(x)^{-1}$. Finalmente, la ecuación iterativa para el método de Newton Raphson se da como:

$$x_{n+1} = x_n - J_F(x_n)^{-1}\, F(x_n) \tag{2}$$

Para el caso de varias variables, se puede determinar el error absoluto y relativo asociado a cada variable del vector x_n y determinar la norma del vector compuesto por los errores determinados.

4. Perfil cinemático del motor

Como condiciones iniciales del sistema, se va a generar el perfil cinemático del eje del motor. Dicho esto, se establece que el eje va a partir del reposo y, durante el periodo de tiempo en que el eje del motor complete su primera revolución, va a tener una aceleración máxima de $5\ rad\ s^{-2}$. Para minimizar el jerk del motor, este periodo de tiempo va a separarse en tres periodos de misma longitud:

1. Parte de aceleración en 0, en aumento lineal hasta $5\ rad\ s^{-2}$.
2. Se mantiene en $5\ rad\ s^{-2}$.
3. Disminuye de $5\ rad\ s^{-2}$ hasta 0.

Después de haber ejercido la primera revolución, se va a mantener una velocidad angular constante $\omega_2 \approx 6{,}45\ rad\ s^{-1} \approx 62\ rpm$, la cual es la resultante de esta aceleración. De este modo, se puede definir las ecuaciones de aceleración angular, y mediante procesos de integración, su velocidad y posición angular. A continuación se muestran los resultados:

$$\alpha_2(t) = \begin{cases} \frac{15\,t}{T} & \text{si } 0 \le t < T/3 \\ 5 & \text{si } T/3 \le t\ 2T/3 \\ 15 - \frac{15\,t}{T} & \text{si } 2T/3 \le t < T \\ 0 & \text{si } T \le t \end{cases} \tag{3}$$

$$\omega_2(t) = \begin{cases} \frac{15\,t^2}{2\,T} & \text{si } 0 \le t < T/3 \\ 5\,t - \frac{5\,T}{6} & \text{si } T/3 \le t < 2T/3 \\ 15\,t - \frac{25\,T}{6} - \frac{15\,t^2}{2\,T} & \text{si } 2T/3 \le t \\ \frac{10\,T}{3} & \text{si } T \le t \end{cases} \tag{4}$$

$$\theta_2(t) = \begin{cases} \frac{5\,t^3}{2\,T} & \text{si } 0 \le t < T/3 \\ \frac{5\,T^2}{54} - \frac{5\,t\,(T-3\,t)}{6} & \text{si } T/3 \le t < 2T/3 \\ \frac{5\,T^2}{6} - \frac{25\,T\,t}{6} + \frac{15\,t^2}{2} - \frac{5\,t^3}{2\,T} & \text{si } 2T/3 \le t < T \\ \frac{10\,T\,t}{3} - \frac{5\,T^2}{3} & \text{si } T \le t \end{cases} \tag{5}$$

Con base a los resultados obtenidos en (5), es posible determinar el tiempo T en que el eje realiza la primera revolución. Dado que $\theta_2(T) = 2\pi$, se soluciona para esta condición, determinando que $T = 1{,}942\ s$. A partir de este resultado se grafican los valores de tiempo el perfil cinemático del eje del motor para un tiempo $2T$, con el fin de evidenciar el motor a velocidad constante.

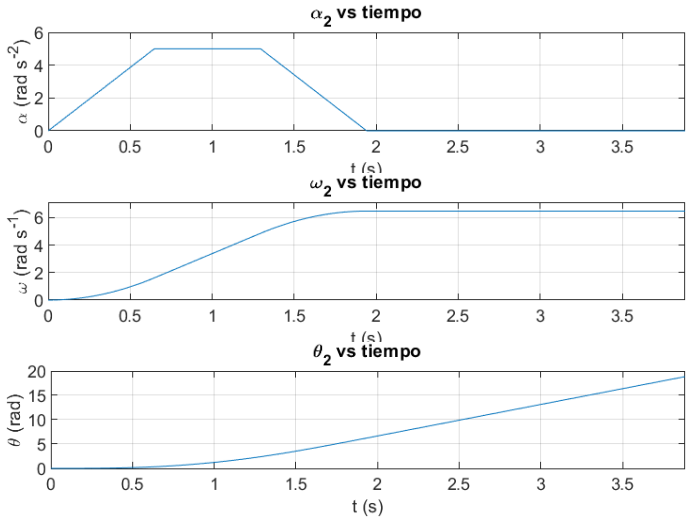


Figura 3: Perfil cinemático del eje del motor.

5. Resolución del problema

5.1. Cadena cinemática

Para poder solucionar el problema, lo primero es definir la cadena cinemática definida por los vectores del problema a partir de la figura 2. A continuación se evidencia la expresión correspondiente:

$$\vec{r_1} = \vec{r_2} + \vec{r_3} \tag{6}$$

A partir de (6) se pueden definir las ecuaciones de posición respecto a las variables deseadas del sistema. Sumado a esto, igualando las ecuaciones a 0 y mediante procesos de derivación de las mismas, se pueden determinar los perfiles cinemáticos de las variables.

5.1.1. Posición

Con las condiciones dadas del problema, se pueden definir dos ecuaciones dimensionales respecto a los ejes coordenados x y y e igualarlas a 0. Adicionalmente, se debe plantear la ecuación de ligadura entre r_1 y θ_1. Teniendo en cuenta que r_1, θ_1 θ_2 y θ_3 varían en el tiempo, se tiene que:

$$Fx(t) = 0 = -r_1(t)Cos(\theta_1(t)) + r_2Cos(\theta_2(t)) + r_3Cos(\theta_2(t)) \tag{7}$$
$$Fy(t) = 0 = r_1Sen(\theta_1(t)) - r_2Sen(\theta_2(t)) - r_3Sen(\theta_2(t)) \tag{8}$$
$$Fc(t) = 0 = r_1(t)Sen(\theta_1(t)) - r_{1y} \tag{9}$$

5.1.2. Velocidad

Para las ecuaciones de velocidad basta con derivar las ecuaciones de posición respecto al tiempo.

$$\frac{\partial Fx(t)}{\partial t} = 0 = \omega_1\, r_1 \sin(\theta_1) - V_r \cos(\theta_1) - \omega_2\, r_2 \sin(\theta_2) - \omega_3\, r_3 \sin(\theta_3) \tag{10}$$
$$\frac{\partial Fy(t)}{\partial t} = 0 = V_r \sin(\theta_1) + \omega_1\, r_1 \cos(\theta_1) - \omega_2\, r_2 \cos(\theta_2) - \omega_3\, r_3 \cos(\theta_3) \tag{11}$$
$$\frac{\partial Fc(t)}{\partial t} = 0 = V_r \sin(\theta_1) + \omega_1\, r_1 \cos(\theta_1) \tag{12}$$

5.1.3. Aceleración

Para determinar la aceleración, se derivan las funciones de velocidad.

$$\frac{\partial^2 Fx(t)}{\partial t^2} = 0 = r_1 \cos(\theta_1)\ {\omega_1}^2 + 2\,V_r \sin(\theta_1)\ \omega_1 - r_2 \cos(\theta_2)\ {\omega_2}^2 - r_3 \cos(\theta_3)\ {\omega_3}^2 \\ -a_r \cos(\theta_1) + \alpha_1\, r_1 \sin(\theta_1) - \alpha_2\, r_2 \sin(\theta_2) - \alpha_3\, r_3 \sin(\theta_3) \tag{13}$$

$$\frac{\partial^2 Fy(t)}{\partial t^2} = 0 = -r_1 \sin(\theta_1)\ {\omega_1}^2 + 2\,V_r \cos(\theta_1)\ \omega_1 + r_2 \sin(\theta_2)\ {\omega_2}^2 + r_3 \sin(\theta_3)\ {\omega_3}^2 \\ +a_r \sin(\theta_1) + \alpha_1\, r_1 \cos(\theta_1) - \alpha_2\, r_2 \cos(\theta_2) - \alpha_3\, r_3 \cos(\theta_3) \tag{14}$$

$$\frac{\partial^2 Fc(t)}{\partial t^2} = 0 = -r_1 \sin(\theta_1)\,\omega_1^2 \ +\ 2\,V_r \cos(\theta_1)\ \omega_1 \ +\ a_r \sin(\theta_1) \ +\ \alpha_1\, r_1 \cos(\theta_1) \tag{15}$$

5.2. Implementación del método de Newton Raphson

Con las ecuaciones del sistema definidas, se puede implementar el método de Newton Raphson para resolver las variables deseadas, con una cota de error $\varepsilon < 10^{-6}$. Inicialmente se trabajará el método de forma consecutiva, es decir, se resolverán las variables de posición, y se utilizarán sus valores como entradas para la resolución de la velocidad, y de la misma forma con los resultados de velocidad para la aceleración. Luego se realizará una segunda forma de aplicar el método, resolviendo las nueve variables de forma simultanea.

A partir de unas condiciones iniciales del motor en reposo: $\theta_2 = 0$, $\omega_2 = 0$ y $\alpha_2 = 0$, con valores de las constantes del sistema r_{1y}, r_2 y r_3 de $45\ mm,\ 75\ mm$ y $170\ mm$ respectivamente, se va a tomar un valor semilla:

$$x_0 = \begin{pmatrix} r_1 \\ \theta_1 \\ \theta_3 \\ v_r \\ \omega_1 \\ \omega_3 \\ a_r \\ \alpha_1 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} -0{,}1 \\ -0{,}5 \\ 3 \\ 0{,}5 \\ 0 \\ -0{,}5 \\ 0{,}5 \\ 0 \\ -0{,}5 \end{pmatrix} \begin{bmatrix} m \\ rad \\ rad \\ m\ s^{-1} \\ rad\ s^{-1} \\ rad\ s^{-1} \\ m\ s^{-2} \\ rad\ s^{-2} \\ rad\ s^{-2} \end{bmatrix} \tag{16}$$

Se toman estos valores medianamente alejados de la solución para que se evidencie la convergencia mediante el método.

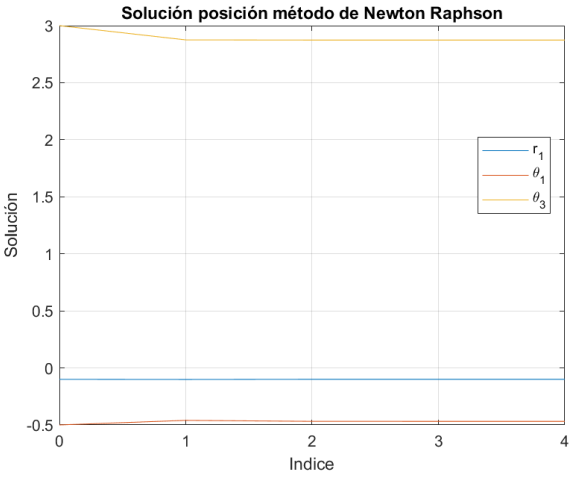
5.2.1. NR Consecutivo

Aplicando el método de Newton Raphson definido por (2), se implementan las expresiones de (7), (8) y (9) como el vector $F(x_n)$. Definiendo el jacobiano para $F(x_n)$ como:

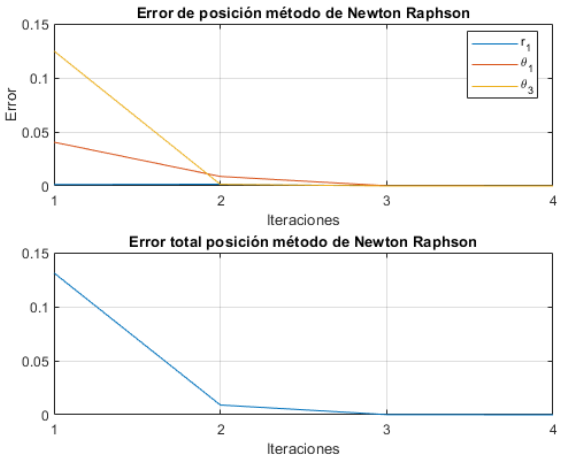
$$J_{pos} = \begin{pmatrix} \sin(\theta_1) & r_1 \cos(\theta_1) & 0 \\ 0 & r_{1\mathrm{y}} \left(\cot(\theta_1)^2 + 1\right) & -r_3 \sin(\theta_3) \\ 0 & 0 & -r_3 \cos(\theta_3) \end{pmatrix} \tag{17}$$

Y su Jacobiano inverso que se encuentra en el Anexo C.1.

Con todo y lo anterior, se realiza el proceso iterativo para la solución del sistema, obteniendo las siguientes gráficas de la convergencia de las variables de posición r_1, θ_1 y θ_3, con sus respectivos errores por variable, y el error total:



(a) Convergencia de las variables de posición.



(b) Errores de las variables de posición.

Figura 4: Resultados del método de NR para la posición.

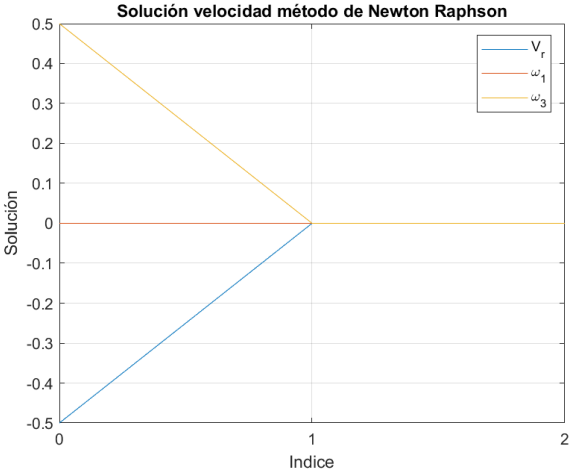
Después de 4 iteraciones, se tiene que:

$$r_1 = -0{,}099672\ m \qquad \theta_1 = -0{,}468422\ rad \qquad \theta_3 = 2{,}873694\ rad$$

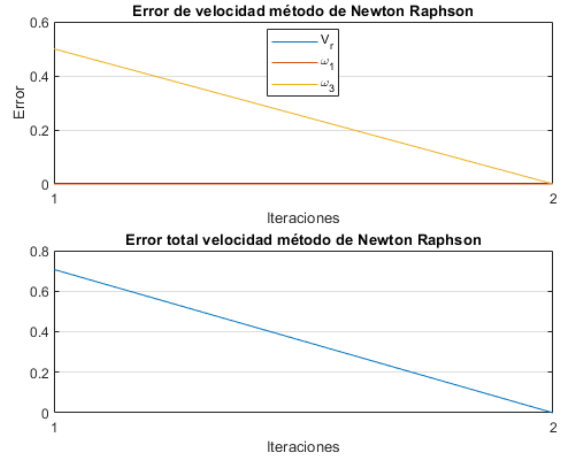
Para resolver las velocidades, se utilizan las ecuaciones (10), (11) y (12) para formar $F(x_n)$, y se define su matriz Jacobiana (su Jacobiano inverso se encuentra en el Anexo C.2):

$$J_{vel} = \begin{pmatrix} -\cos(\theta_1) & r_1 \sin(\theta_1) & -r_3 \sin(\theta_3) \\ \sin(\theta_1) & r_1 \cos(\theta_1) & -r_3 \cos(\theta_3) \\ \sin(\theta_1) & r_1 \cos(\theta_1) & 0 \end{pmatrix} \tag{18}$$

Al implementar el método descrito en (2) y los valores semilla dados, se obtienen los resultados para V_r, ω_1 y ω_3:



(a) Convergencia de las variables de velocidad.



(b) Errores de las variables de velocidad.

Figura 5: Resultados del método de NR para la velocidad.

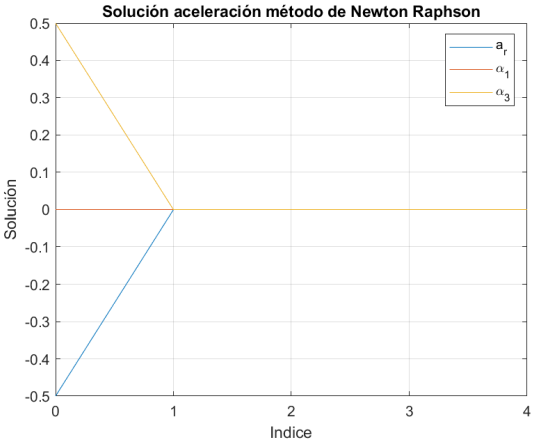
Después de 2 iteraciones, se tiene que:

$$V_1 = 0\ m\ s^{-1} \qquad \omega_1 = 0\ rad\ s^{-1} \qquad \omega_3 = 0\ rad\ s^{-1}$$

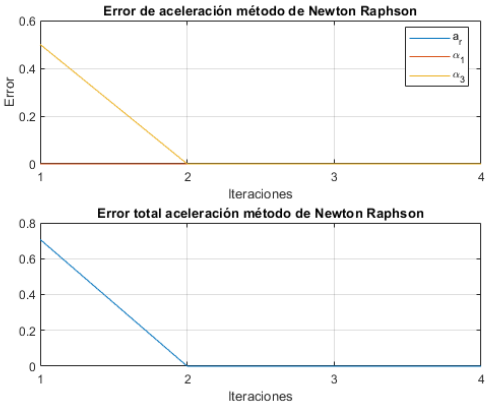
Finalmente, para los valores de aceleración, utilizando (13), (14) y (15) para formar $F(x_n)$, se define su Jacobiano (y su Jacobiano que se encuentra en el Anexo C.3) como:

$$J_{ac} = \begin{pmatrix} \sin(\theta_1) & r_1\cos(\theta_1) & 0 \\ -\cos(\theta_1) & r_1\sin(\theta_1) & -r_3\sin(\theta_3) \\ \sin(\theta_1) & r_1\cos(\theta_1) & -r_3\cos(\theta_3) \end{pmatrix} \tag{19}$$

Al aplicar el método de NR con los valores semilla, se obtienen los siguientes resultados para A_r, α_1 y α_3:



(a) Convergencia de las variables de aceleración.



(b) Errores de las variables de aceleración.

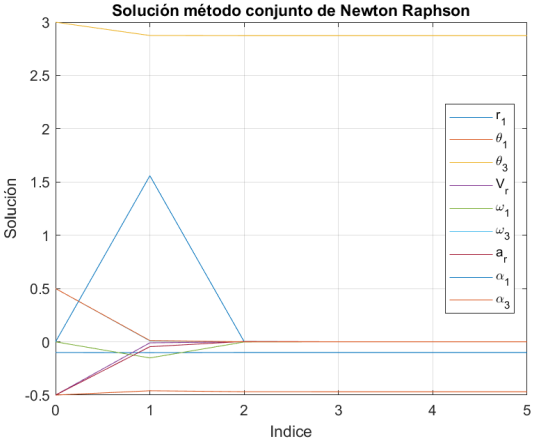
Figura 6: Resultados del método de NR para la aceleración.

Después de 4 iteraciones, se tiene que:

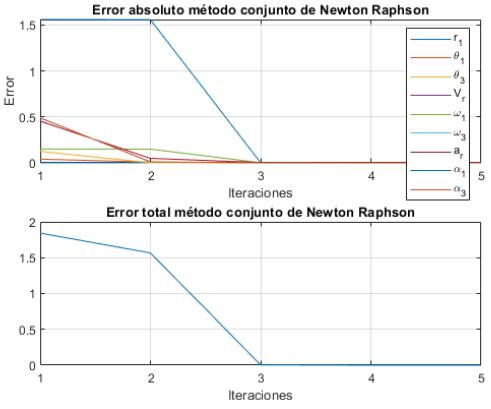
$$A_r = 0\ m\ s^{-2} \qquad \alpha_1 = 0\ rad\ s^{-2} \qquad \alpha_3 = 0\ rad\ s^{-2}$$

5.2.2. NR Simultaneo

Ahora bien, estudiando el método de NR, fue posible determinar que el sistema se puede resolver llevando a cabo una implementación simultanea de las nueve ecuaciones que describen la cinemática general del sistema. A pesar de que las soluciones de velocidad y aceleración dependan de valores de posición, y de posición y velocidad respectivamente, a partir del valor semilla y de la convergencia de la posición, se converge simultáneamente a la solución de velocidad, y del mismo modo para la aceleración. Debido a que el sistema se resuelve en simultaneo, $F(x_n)$ se define con las nueve ecuaciones de cinemática, su Jacobiano de tamaño 9×9 se encuentra en el anexo D, debido a la extensión del inverso del Jacobiano, no se anexa al documento. Con todo y lo anterior, se obtienen los siguientes resultados:



(a) Convergencia de las variables en simultaneo.



(b) Errores de las variables en simultaneo.

Figura 7: Resultados del método de NR en simultaneo.

Después de cinco iteraciones, se tiene que los valores finales son:

$$\begin{aligned} r_1 &= -0{,}099672\ m & \theta_1 &= -0{,}468422\ rad & \theta_3 &= 2{,}873694\ rad \\ V_r &= 0\ m\ s^{-1} & \omega_1 &= 0\ rad\ s^{-1} & \omega_3 &= 0\ rad\ s^{-1} \\ A_r &= 0\ m\ s^{-2} & \alpha_1 &= 0\ rad\ s^{-2} & \alpha_3 &= 0\ rad\ s^{-2} \end{aligned}$$

Comparando estos resultados con los encontrados de manera consecutiva, se evidencia que ambos métodos convergen a la misma solución.

5.3. Solución analítica

A pesar de que el problema se va a solucionar implementando el método de Newton Raphson, es importante tener la solución analítica al problema. Esto se va a realizar porque es necesario comparar los resultados obtenidos por el método de NR con los valores exactos del sistema, para determinar el error real obtenido en la solución. Además, si se fueran a comparar los resultados del NR directamente con los resultados de simulación se pueden tener errores de medición, debido a las limitaciones de los software como *Inventor*.

Dicho esto, a partir de las ecuaciones encontradas en la sección 5.1, es posible determinar las soluciones analíticas para las nueve variables del sistema. Estas ecuaciones se determinan mediante procedimientos matemáticos que no se van a profundizar en el informe.

$$\theta_3 = \pi - \operatorname{asin}\left(\frac{r_{1y} - r_2 \sin(\theta_2)}{r_3}\right) \tag{20}$$

$$\theta_1 = \operatorname{acot}\left(\frac{r_2 \cos(\theta_2) + r_3 \cos(\theta_3)}{r_{1y}}\right) \tag{21}$$

$$r_1 = -\frac{\omega_2\, r_2 \cos(\theta_2)}{r_3 \cos(\theta_3)} \tag{22}$$

$$\omega_3 = \operatorname{acot}\left(\frac{r_2 \cos(\theta_2) + r_3 \cos(\theta_3)}{r_{1y}}\right) \tag{23}$$

$$\omega_1 = -\frac{V_r \tan(\theta_1)}{r_1} \tag{24}$$

$$V_r = -\cos(\theta_1)\ (\omega_2\, r_2 \sin(\theta_2) + \omega_3\, r_3 \sin(\theta_3)) \tag{25}$$

$$\alpha_3 = \frac{r_2 \sin(\theta_2)\ {\omega_2}^2 + r_3 \sin(\theta_3)\ {\omega_3}^2 - \alpha_2\, r_2 \cos(\theta_2)}{r_3 \cos(\theta_3)} \tag{26}$$

$$\alpha_1 = {\omega_1}^2 \tan(\theta_1) - \frac{2\, V_r\, \omega_1 + a_r \tan(\theta_1)}{r_1} \tag{27}$$

$$\begin{aligned} A_r = r_1\, {\omega_1}^2 - r_2 \cos(\theta_1)\ \cos(\theta_2)\ {\omega_2}^2 - r_3 \cos(\theta_1)\ \cos(\theta_3)\ {\omega_3}^2 \\ -\alpha_2\, r_2 \cos(\theta_1)\ \sin(\theta_2) - \alpha_3\, r_3 \cos(\theta_1)\ \sin(\theta_3) \end{aligned} \tag{28}$$

6. Resultados

Al demostrar la convergencia del método de Newton Raphson para el punto inicial del movimiento del mecanismo, es posible definir todo el movimiento rotacional del mecanismo bajo los perfiles cinemáticos del motor descritos en la sección 4. Esto se realizará mediante la implementación del NR de forma simultanea, de forma consecutiva, de manera analítica, y mediante el software de diseño asistido por computadora *Autodesk Inventor*. La solución se realiza para 1000 puntos distribuidos en un periodo de $2T \approx 3{,}88\ s$.

Acerca del método del NR, los valores semilla del punto P_i serán la solución determinada por el método para el punto P_{i-1}, lo cual garantiza que el valor semilla sea cercano a la solución. Para el

punto inicial en $t = 0$ cuando el motor está en reposo, el valor semilla será los resultados obtenidos en la sección 5.2.1 y 5.2.2, los cuales son equivalentes.

6.1. NR Simultaneo

Al aplicar el método de Newton Raphson de forma simultanea se obtienen los siguientes perfiles de posición, velocidad y aceleración para las variables a encontrar. También se muestra el error asociado al método.

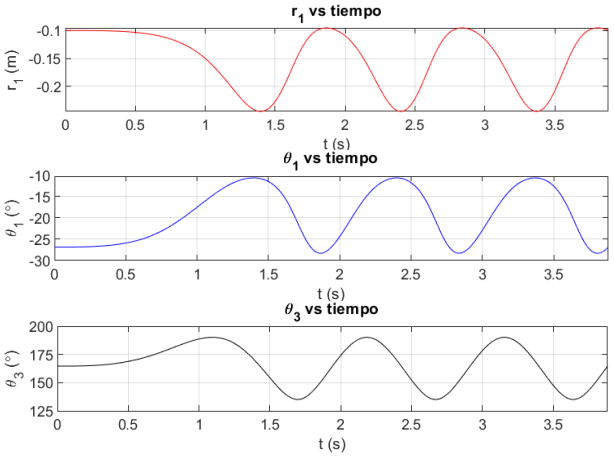


Figura 8: Posiciones para NR simultaneo.

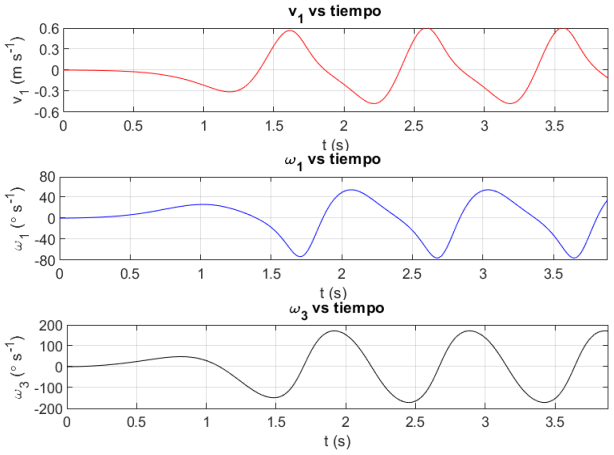


Figura 9: Velocidades para NR simultaneo.

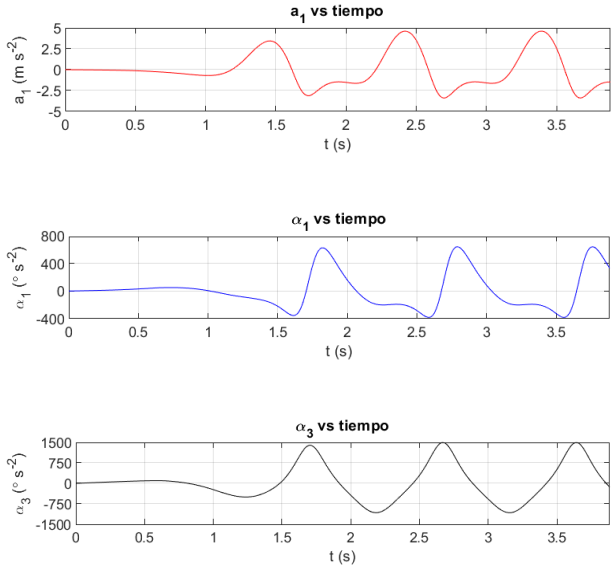


Figura 10: Aceleraciones para NR simultaneo.

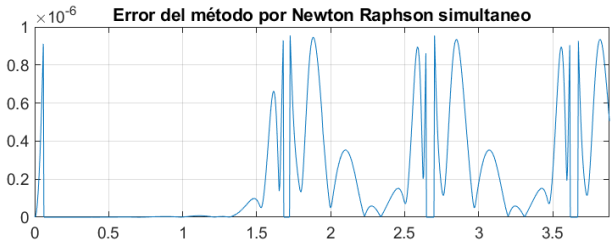


Figura 11: Error asociado al método de NR simultáneo.

6.2. NR Consecutivo

El mismo procedimiento realizado para el método de Newton Raphson de forma simultanea se realiza para el método de forma consecutiva. Debido a la extensión del documento, y a la similitud que se tienen con los resultados anteriores, las gráficas de resultados se encuentran en el anexo A.

6.3. Simulación

Para poder corroborar los resultados obtenidos mediante el uso del método de Newton Raphson, se decide hacer un modelado 3D del mecanismo en el software *Autodesk Inventor*. El mecanismo simulado resume el movimiento de la sierra eléctrica real utilizando eslabones y juntas que definidos a partir de las condiciones iniciales del problema.

Figura 12: Modelo en Inventor del Mecanismo 3D

Como se puede evidenciar, a la bancada se conecta la manivela, el cual es el eslabón donde se genera el movimiento por el perfil cinemático del motor. A la manivela se le conecta el acoplador, el cual es el eslabón de mayor extensión, también conectado a la corredera, la cual se une a la sierra y se pueden desplazar sobre la guía de la bancada.

Los resultados de simulación se generan a partir del entorno de simulación dinámica del software. En este entorno, se impone el perfil de velocidad del motor al eje de la manivela, donde es el mismo perfil utilizado para encontrar los resultados por NR. Con esto, se genera la simulación del modelado, donde se pueden extraer los valores cinemáticos de todas las variables implicadas en el sistema. Cabe resaltar que, al igual que con el método de NR, se resuelve para 1000 puntos en un periodo de $3{,}883\ s$. Las gráficas de resultados se encuentran en el anexo B.

7. Análisis de resultados

7.1. Comparación de aplicación de NR

Se realizaron dos variaciones del método de Newton Raphson, uno donde se resolvían las nueve variables en simultaneo, y otra donde se resolvían de manera consecutiva, por lo que es conveniente señalar las diferencias entre estos dos casos. Lo primero es comparar los errores asociados al método, comparando la figura 11 con la figura 24 del anexo A, El error del método simultáneo suele ser mayor al método consecutivo, sin embargo, todos los valores están acotados por la tolerancia establecida ε. Ahora, se muestra el error absoluto entre estos dos casos:

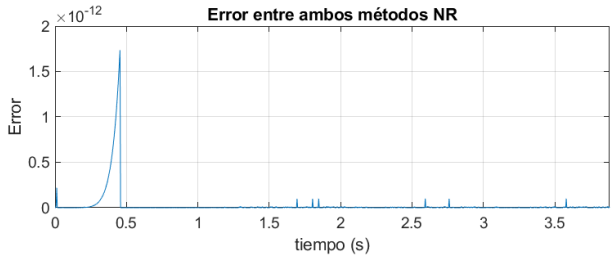
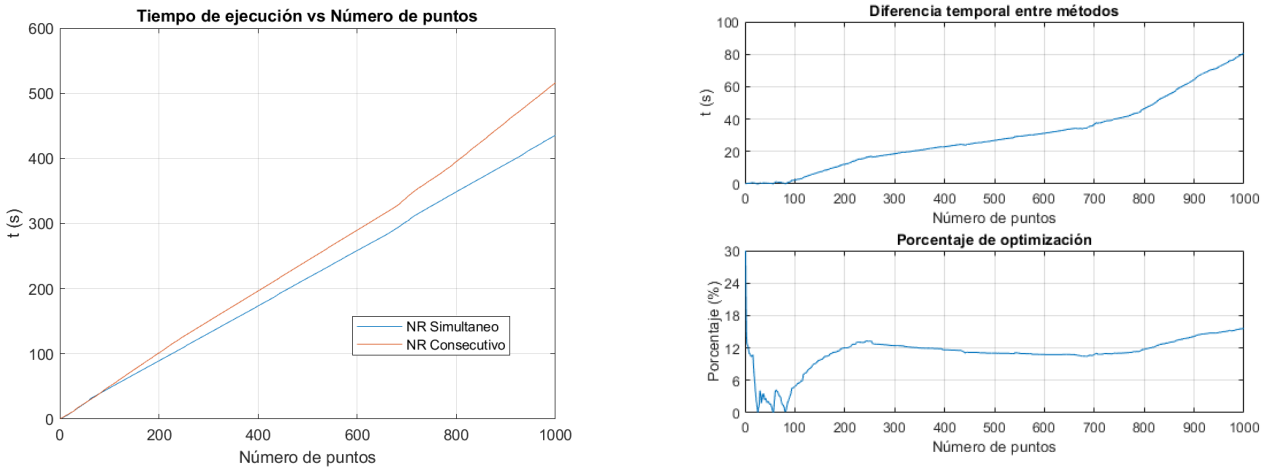


Figura 13: Error absoluto entre las versiones del método de Newton Raphson

Como se puede evidenciar, un error absoluto en el orden de 10^{-12} mientras se tiene una cota de error del método de $\varepsilon < 10^{-6}$ indica que los resultados obtenidos entre los métodos fueron bastante precisos a pesar de resolverse de maneras distintas.

Ahora bien, para definir cual de estos dos se toma como la solución final del sistema, se va a mostrar el tiempo de ejecución para cada caso. De este modo, se puede establecer cual caso es más eficiente en calcular los resultados.



(a) Tiempo de ejecución para los métodos.

(b) Diferencia en los tiempos de ejecución.

Figura 14: Resultados de tiempo de ejecución del sistema.

Como se puede evidenciar, resolver el sistema mediante el método de Newton Raphson de manera simultánea es más óptimo que realizarlo de manera consecutiva. A pesar de que los tiempos de ejecución tienen un comportamiento lineal, la pendiente del método de forma consecutiva es más elevada. Dicho esto, considerando que los resultados obtenidos por ambos métodos son casi iguales, y que se tiene una reducción del tiempo de ejecución de aproximadamente $12\,\%$ para el método en simultáneo, este se considera como la mejor solución, teniendo en cuenta que ese porcentaje es considerablemente alto cuando se trabaja con un número de datos elevados. Así pues, se van a considerar los resultados obtenidos por el método de NR simultáneo como los resultados finales del problema.

7.2. Error absoluto del método de Newton Raphson

A partir de las ecuaciones determinadas de la solución analítica en la sección 5.3 se pueden determinar las soluciones exactas del problema, donde se van a utilizar para determinar el error absoluto que se generó al solucionar el sistema. Al realizar este procedimiento, se determinan los siguientes errores absolutos a las soluciones encontradas.

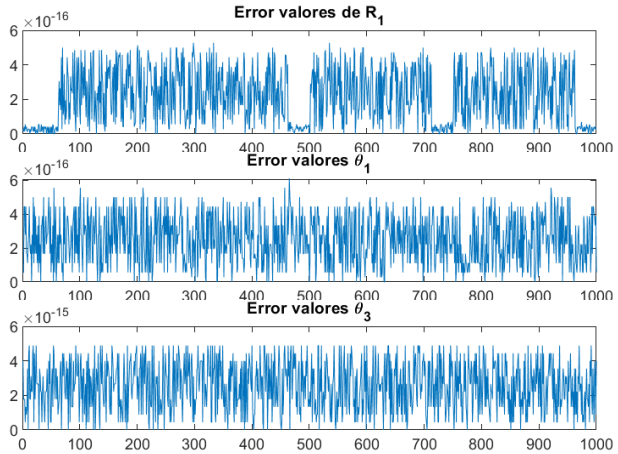


Figura 15: Error absoluto para resultados de posición.

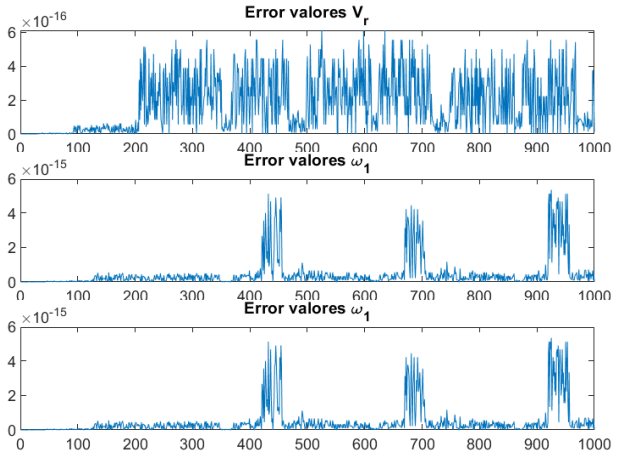


Figura 16: Error absoluto para resultados de velocidad.

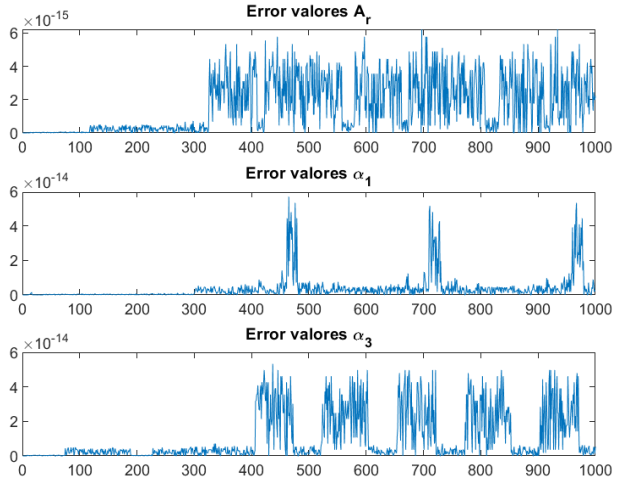


Figura 17: Error absoluto para resultados de aceleración.

Como se puede evidenciar en las gráficas, el error absoluto asociado a las posiciones y velocidades del sistema se encuentra en el orden de 10^{-15}, mientras que para las aceleraciones se encuentra en el orden de 10^{-14}. Ahora bien, teniendo en cuenta que el error de NR se acotó como $\varepsilon < 10^{-6}$, y que el error absoluto de todos los resultados siempre es menor a 10^{-13}, se establece que los valores obtenidos mediante el método de Newton Raphson simultaneo fueron altamente exactos y que resolver el sistema mediante esta versión del método numérico es una manera bastante sencilla y efectiva de solución.

7.3. Comparación de resultados NR-Simulación *Inventor*

Para finalizar, se van a comparar los resultados obtenidos por el método de Newton Raphson con los valores de simulación que se pueden evidenciar en el anexo B. De este modo, los resultados de *Inventor* se obtienen a partir de una interfaz de simulación dinámica altamente utilizada para resolver este tipo de sistemas en las aplicaciones de ingeniería, debido a su capacidad de mostrar de manera virtual el comportamiento de los mecanismos ensamblados en la vida real. Hecha esta observación, se superponen los resultados obtenidos para el sistema con los valores obtenidos en *Inventor*, obteniendo las siguientes gráficas:

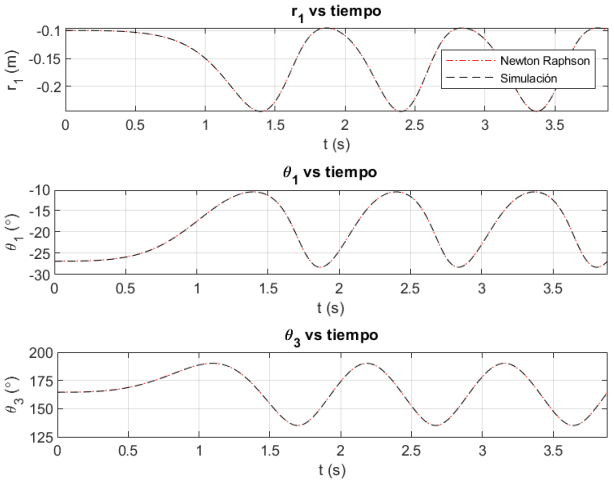


Figura 18: Comparación NR-Inventor para el perfil de posición.

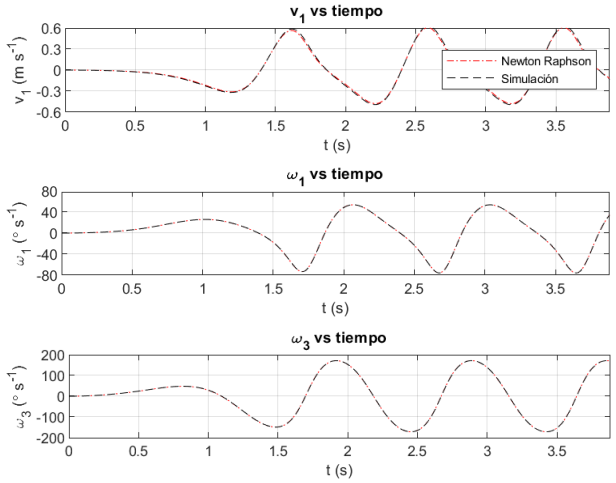


Figura 19: Comparación NR-Inventor para el perfil de velocidad.

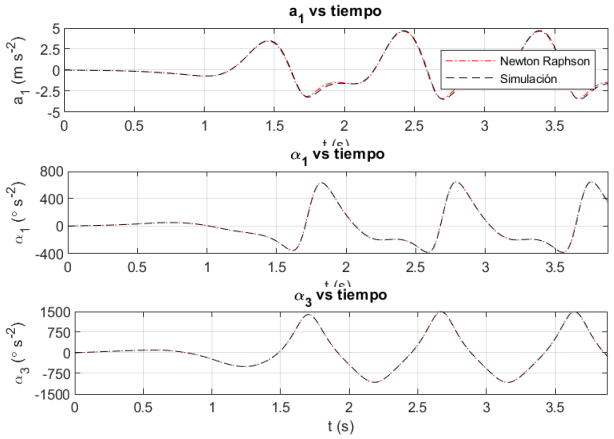


Figura 20: Comparación NR-Inventor para el perfil de aceleración.

Como es posible apreciar mediante el análisis de las gráficas, los resultados obtenidos en la simulación en *Inventor* concuerdan con los valores obtenidos mediante el método de Newton Raphson simultaneo. Esto quiere decir que mediante la implementación de sistemas de solución por simulaciones CAD se obtienen resultados semejantes a los que se determinan resolviendo el mismo sistema utilizando el método numérico de Newton Raphson para sistemas de ecuaciones no lineales, con lo que se puede establecer la veracidad de solucionan del mecanismo a partir del método.

8. Conclusiones

- Se puede implementar el método de Newton Raphson para resolver los sistemas de ecuaciones que describen problemas aplicados a la ingeniería, en este caso, los perfiles cinemáticos asociados a un mecanismo.
- El uso recurrente de métodos numéricos se puede implementar como procedimiento de solución a la cinemática de mecanismos para cualquier instante t o intervalo $[t_0 \quad t_1]$, esto con el fin de optimizar la precisión de la solución y de los recursos computacionales asociados al mismo.
- Se obtienen los mismos resultados finales resolviendo mediante Newton Raphson el sistema de posición, velocidad y aceleración simultáneamente, a resolver cada uno de estos de manera individual. Además, resolver el mecanismo de manera simultánea reduce los recursos computacionales asociados a la solución, disminuyendo el tiempo total para solucionar el sistema.
- Se tiene una alta exactitud de solución al problema al implementar el método de Newton Raphson, obteniendo un error absoluto con la solución analítica menor a 10^{-13} para cualquier caso del sistema. Sumado a esto, al contrastar los resultados con las simulaciones del mecanismo en $Autodesk\ \ Inventor$, se logra establecer la veracidad de la solución para toda variable e instante del movimiento.

9. Referencias

[1] Myzka,H,David.(2012).Máquinas y Mecanismos.4.Ed.

[2] Chapra, S. C., & Canale, R. P. Numerical methods for engineers. McGraw-Hill Higher Education, Boston, 2006

[3] Mathews, J.H. & Fink, K.D. Métodos numéricos con Matlab. Prentice Hall, Madrid, 2000

A. Gráficas de resultados para el método de Newton Raphson consecutivo

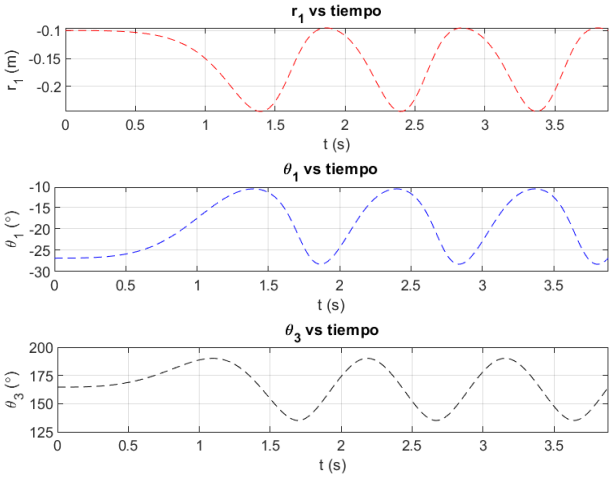


Figura 21: Posiciones para NR consecutivo.

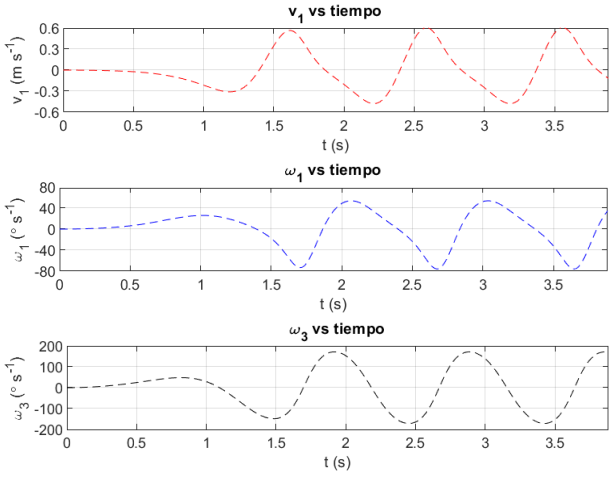


Figura 22: Velocidades para NR consecutivo.

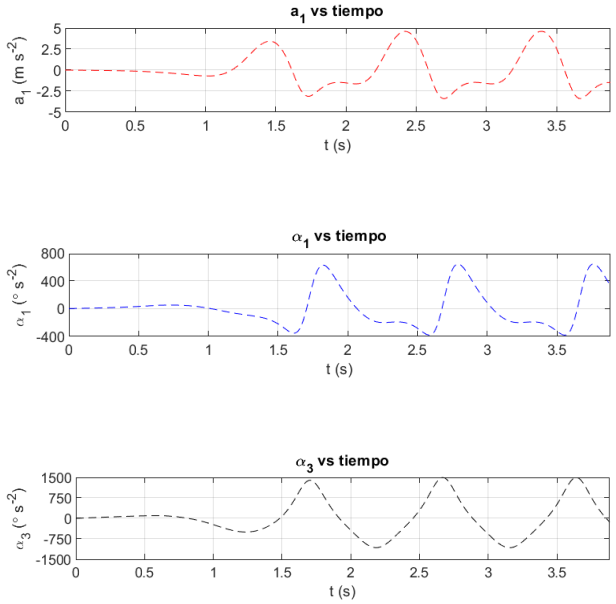


Figura 23: Aceleraciones para NR consecutivo.

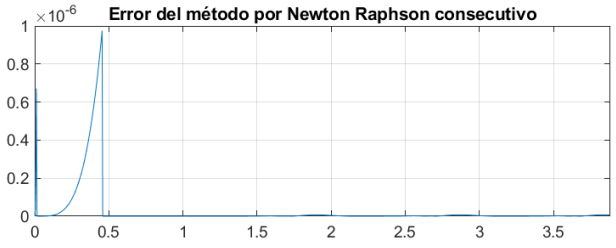


Figura 24: Error asociado al método de NR consecutivo.

B. Resultados de simulación de inventor

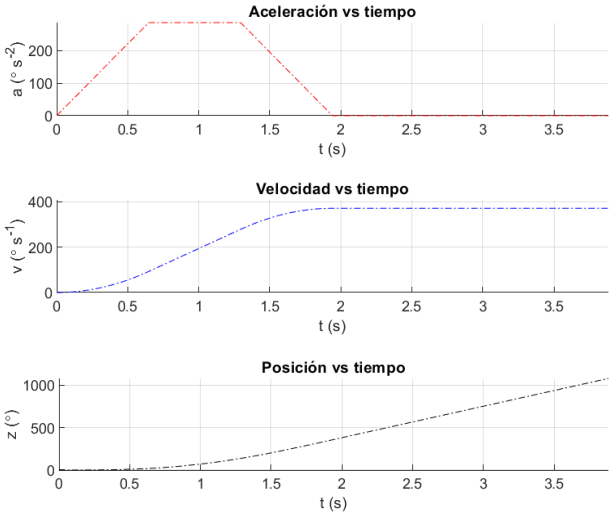


Figura 25: Perfil cinemático del eje del motor en *Autodesk Inventor*.

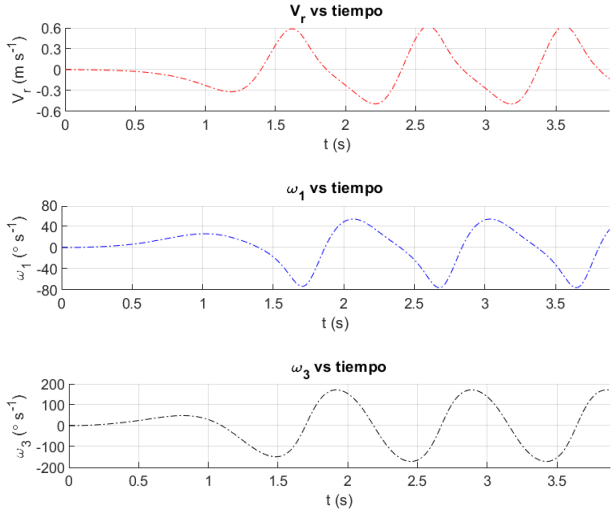


Figura 27: Velocidades obtenidas en *Autodesk Inventor*

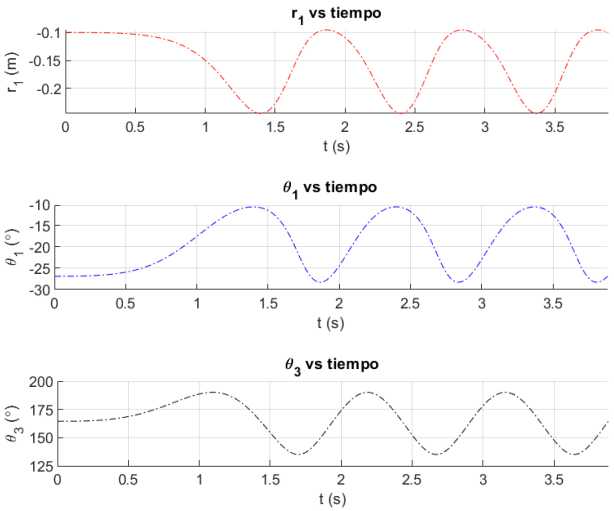


Figura 26: Posiciones obtenidas en *Autodesk Inventor*

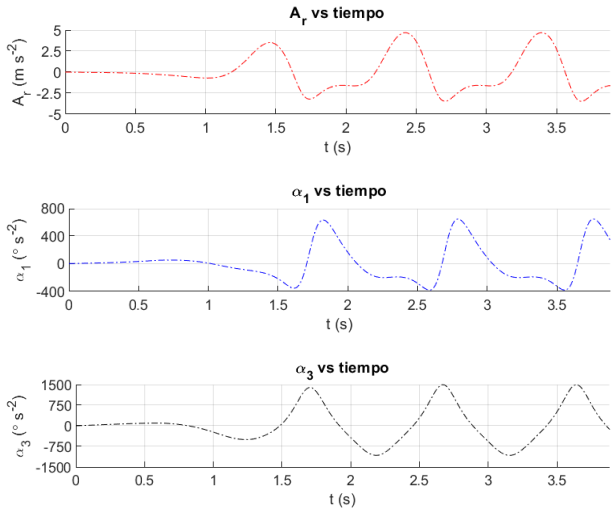


Figura 28: Aceleraciones obtenidas en *Autodesk Inventor*.

C. Jacobianos Inversos

C.1. Posición:

$$J_{pos}^{-1} = \begin{pmatrix} \frac{1}{\sin(\theta_1)} & -\frac{r_1\cos(\theta_1)}{r_{1y}\sin(\theta_1)\,\sigma_1} & \frac{r_1\cos(\theta_1)\sin(\theta_3)}{r_{1y}\cos(\theta_3)\sin(\theta_1)\,\sigma_1} \\ 0 & \frac{1}{r_{1y}\,\sigma_1} & -\frac{\sin(\theta_3)}{r_{1y}\cos(\theta_3)\,\sigma_1} \\ 0 & 0 & -\frac{1}{r_3\cos(\theta_3)} \end{pmatrix}$$

Donde:

$$\sigma_1 = \cot(\theta_1)^2 + 1$$

C.2. Velocidad:

$$J_{vel}^{-1} = \begin{pmatrix} -\frac{\cos(\theta_1)}{\cos(\theta_1)^2+\sin(\theta_1)^2} & \frac{\sigma_4}{\sigma_2} & -\frac{\sigma_4-\cos(\theta_3)\sin(\theta_1)}{\sigma_2} \\ \frac{\sin(\theta_1)}{r_1\cos(\theta_1)^2+r_1\sin(\theta_1)^2} & -\frac{\sigma_3}{\sigma_1} & \frac{\cos(\theta_1)\cos(\theta_3)+\sigma_3}{\sigma_1} \\ 0 & -\sigma_5 & \sigma_5 \end{pmatrix}$$

Donde:

$$\sigma_1 = r_1\cos(\theta_3)\,\cos(\theta_1)^2 + r_1\cos(\theta_3)\,\sin(\theta_1)^2$$

$$\sigma_2 = \cos(\theta_3)\,\cos(\theta_1)^2 + \cos(\theta_3)\,\sin(\theta_1)^2$$

$$\sigma_3 = \sin(\theta_1)\,\sin(\theta_3)$$

$$\sigma_4 = \cos(\theta_1)\,\sin(\theta_3)$$

$$\sigma_5 = \frac{1}{r_3\cos(\theta_3)}$$

C.3. Aceleración:

$$J_{ac}^{-1} = \begin{pmatrix} -\frac{\sigma_4 - \cos(\theta_3)\sin(\theta_1)}{\sigma_2} & -\frac{\cos(\theta_1)}{\cos(\theta_1)^2+\sin(\theta_1)^2} & \frac{\sigma_4}{\sigma_2} \\ \frac{\cos(\theta_1)\cos(\theta_3)+\sigma_3}{\sigma_1} & \frac{\sin(\theta_1)}{r_1\cos(\theta_1)^2+r_1\sin(\theta_1)^2} & -\frac{\sigma_3}{\sigma_1} \\ \sigma_5 & 0 & -\sigma_5 \end{pmatrix}$$

Donde:

$$\sigma_1 = r_1 \cos(\theta_3)\ \cos(\theta_1)^2 + r_1 \cos(\theta_3)\ \sin(\theta_1)^2$$

$$\sigma_2 = \cos(\theta_3)\ \cos(\theta_1)^2 + \cos(\theta_3)\ \sin(\theta_1)^2$$

$$\sigma_3 = \sin(\theta_1)\ \sin(\theta_3)$$

$$\sigma_4 = \cos(\theta_1)\ \sin(\theta_3)$$

$$\sigma_5 = \frac{1}{r_3\cos(\theta_3)}$$

D. Jacobiano del sistema simultaneo:

Col. 1-3:

$$\begin{pmatrix} \sin(\theta_1) & r_1\cos(\theta_1) & 0 \\ 0 & r_{1y}\left(\cot(\theta_1)^2+1\right) & \sigma_4 \\ 0 & 0 & \sigma_5 \\ \omega_1\sin(\theta_1) & V_r\sin(\theta_1)+\omega_1 r_1\cos(\theta_1) & -\omega_3 r_3\cos(\theta_3) \\ \omega_1\cos(\theta_1) & \sigma_6 & \omega_3 r_3\sin(\theta_3) \\ \omega_1\cos(\theta_1) & \sigma_6 & 0 \\ \sigma_3 & \sigma_1 & 0 \\ \cos(\theta_1)\,{\omega_1}^2+\alpha_1\sin(\theta_1) & -r_1\sin(\theta_1)\,{\omega_1}^2+2\,V_r\cos(\theta_1)\,\omega_1+a_r\sin(\theta_1)+\alpha_1 r_1\cos(\theta_1) & {\omega_3}^2 r_3\sin(\theta_3)-\alpha_3 r_3\cos(\theta_3) \\ \sigma_3 & \sigma_1 & r_3\cos(\theta_3)\,{\omega_3}^2+\alpha_3 r_3\sin(\theta_3) \end{pmatrix}$$

Col. 4-6:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\cos(\theta_1) & r_1\sin(\theta_1) & \sigma_4 \\ \sin(\theta_1) & r_1\cos(\theta_1) & \sigma_5 \\ \sin(\theta_1) & r_1\cos(\theta_1) & 0 \\ \sigma_7 & \sigma_2 & 0 \\ 2\,\omega_1\sin(\theta_1) & 2\,V_r\sin(\theta_1)+2\,\omega_1 r_1\cos(\theta_1) & -2\,\omega_3 r_3\cos(\theta_3) \\ \sigma_7 & \sigma_2 & 2\,\omega_3 r_3\sin(\theta_3) \end{pmatrix}$$

Col. 7-9:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \sigma_7 & \cos(\theta_1) & 0 \\ -\cos(\theta_1) & r_1\sin(\theta_1) & \sigma_4 \\ \sin(\theta_1) & r_1\cos(\theta_1) & \sigma_5 \end{pmatrix}$$

Donde:

$$\sigma_1 = -r_1 \cos(\theta_1)\ {\omega_1}^2 - 2\,V_r \sin(\theta_1)\ \omega_1 + a_r \cos(\theta_1) - \alpha_1\, r_1 \sin(\theta_1)$$

$$\sigma_2 = 2\,V_r \cos(\theta_1) - 2\,\omega_1\, r_1 \sin(\theta_1)$$

$$\sigma_3 = \alpha_1 \cos(\theta_1) - {\omega_1}^2 \sin(\theta_1)$$

$$\sigma_4 = -r_3 \sin(\theta_3)$$

$$\sigma_5 = -r_3 \cos(\theta_3)$$

$$\sigma_6 = V_r \cos(\theta_1) - \omega_1\, r_1 \sin(\theta_1)$$

$$\sigma_7 = 2\,\omega_1 \cos(\theta_1)$$

E. Códigos de Matlab del desarrollo

A continuación se expone el listado de códigos generados para la solución del proyecto y la función que estos realizan:

1. **Perfil cinemático:** Enfocado en la realización del perfil cinemático del motor, el cual se utiliza como datos de entrada al sistema.
2. **Solución por Newton Raphson consecutivo:** Solución para un único punto para el método de Newton Raphson solucionando la posición, velocidad y aceleración de manera consecutiva.
3. **Solución por Newton Raphson simultáneo:** Solución para un único punto para el método de Newton Raphson solucionando la posición, velocidad y aceleración de manera simultánea.
4. **Solución por método de Newton Raphson:** Solución a todo el perfil cinemático del motor para el mecanismo. Se implementan ambas versiones del NR y se comparan entre si.
5. **Solución analítica:** Se realiza la solución analítica para todo el perfil cinemático del motor para el mecanismo.
6. **Resultados NR simultáneo vs solución analítica:** Comparar los resultados obtenidos por el método de NR con la solución analítica.
7. **Resultados NR simultáneo vs simulación:** Comparar los resultados obtenidos por el método de NR con los resultados de simulación.

Cabe resaltar que la mayoría de archivos utilizan datos externos en formato .txt, generados por los mismos u otro archivo de Matlab, por lo que es importante verificar la ruta de acceso a estos archivos.

E.1. Perfil cinemático

```
clear
clc
syms a1(t) a2(t) a3(t) a4(t) a5(t)  T A
syms v1(t) v2(t) v3(t) v4(t) v5(t)
syms x1(t) x2(t) x3(t) x4(t) x5(t)
assume(T>0)
assume(A>0)

A=5;

# Aceleracion
a1(t)=3*A/T*t;
a2(t)=A;
a3(t)=-3*A/T*t;
Ca3=A-a3(2*T/3);
a3(t)=a3(t)+Ca3;
a4(t)=0;
fprintf('Funcion de aceleracion:')
aceleracion(t)=piecewise((0≤t<T/3),a1,(T/3≤t<2*T/3),a2,(2*T/3≤t<T),a3,(T≤t),a4)

# Velocidad
v1(t)=int(a1);
v2(t)=int(a2);
Cv2=v1(T/3)-v2(T/3);
v2(t)=v2(t)+Cv2;
v3(t)=int(a3);
Cv3=v2(2*T/3)-v3(2*T/3);
v3(t)=v3(t)+Cv3;
v4(t)=int(a4);
Cv4=v3(T)-v4(T);
v4(t)=v4(t)+Cv4;
fprintf('Funcion de velocidad:')
velocidad(t)=piecewise((0≤t<T/3),v1,(T/3≤t<2*T/3),v2,(2*T/3≤t<T),v3,(t≥T),v4)

# Desplazamiento
z1(t)=int(v1);
z2(t)=int(v2);
Cz2=z1(T/3)-z2(T/3);
z2(t)=z2(t)+Cz2;
z3(t)=int(v3);
Cz3=z2(2*T/3)-z3(2*T/3);
z3(t)=z3(t)+Cz3;
z4(t)=int(v4);
Cz4=z3(T)-z4(T);
z4(t)=z4(t)+Cz4;
fprintf('Funcion de desplazamiento:')
posicion(t)=piecewise((0≤t<T/3),z1,(T/3≤t<2*T/3),z2,(2*T/3≤t<T),z3,(t≥T),z4)

# Solucion de tMax
f=2*pi==posicion(T);
Tgiro=solve(f,T);
aceleracion=subs(aceleracion,T,Tgiro);
velocidad=subs(velocidad,T,Tgiro);
posicion=subs(posicion,T,Tgiro);
```

```
fprintf('Tiempo de primera vuelta  %.3f s',Tgiro)

```

Graficas

```
ts=linspace(0,2*Tgiro,1000);
Vmax=double(velocidad(Tgiro));

subplot(3,1,1)
plot(ts,aceleracion(ts))
title('Aceleracion angular vs tiempo')
grid on
xlabel('t (s)')
ylabel('\alpha (rad s^{-2})')
ylim([0 6])
xlim([0 double(2*Tgiro)])
yticks(0:2:6)

subplot(3,1,2)
plot(ts,velocidad(ts))
title('Velocidad angular vs tiempo')
grid on
xlabel('t (s)')
ylabel('\omega (rad s^{-1})')
ylim([0 Vmax*1.1])
xlim([0 double(2*Tgiro)])
yticks(0:2:8)

subplot(3,1,3)
plot(ts,posicion(ts))
title('Posicion angular vs tiempo')
grid on
xlabel('t (s)')
ylabel('\theta (rad)')
xlim([0 double(2*Tgiro)])
yticks(0:5:20)

TablaCinematica=table(double(ts)',double(aceleracion(ts))',double...
(velocidad(ts))',double(posicion(ts))');
TablaCinematica.Properties.VariableNames=["Tiempo","Aceleracion",...
"Velocidad","Posicion"]
writetable(TablaCinematica,'Perfil_cinematica.txt','delimiter',' ', ...
    'WriteVariableNames', true)
TablaTXT=table(double(ts)',double(velocidad(ts)*180/pi)');
writetable(TablaTXT,'Perfil_Velocidad.txt','delimiter',' ', ...
    'WriteVariableNames', false)
```

E.2. Solución por Newton Raphson Consecutivo

```
clear
syms t Theta_1(t) Theta_2(t) Theta_3(t) R_1(t)
syms r_1 r_1y r_2 r_3 x
syms V_r omega_1 omega_2 omega_3 theta_1 theta_2 theta_3
syms a_r alpha_1 alpha_2 alpha_3

# Ecuaciones
Vr_2=0.075
Vr_3=0.17
Vr_1y=0.045

Vtheta_2=0
Vomega_2=0
Valpha_2=0

Fc=sin(Theta_1)*R_1-r_1y;
Fc=simplify(Fc,'Steps',5)

fc=subs(Fc,[Theta_1, R_1],[theta_1, r_1]);
SolR_1=solve(fc,r_1);

Fx=-R_1*cos(Theta_1)+r_2*cos(Theta_2)+r_3*cos(Theta_3);
Fx=simplify(Fx,'Steps',5)

fx=formula(subs(Fx,[Theta_1, Theta_2, Theta_3, R_1],[theta_1, theta_2, ...
    theta_3, SolR_1]))
fx=simplify(fx,'Steps',5)
SolTheta_1=solve(fx,theta_1);


Fy=R_1*sin(Theta_1)-r_2*sin(Theta_2)-r_3*sin(Theta_3);
Fy=simplify(Fy,'Steps',5)

fy=formula(subs(Fy,[Theta_1, Theta_2, Theta_3, R_1],[theta_1, theta_2, ...
    theta_3, SolR_1]))

SolTheta_3=solve(fy,theta_3);
SolTheta_3=SolTheta_3(1)

DFcDt=diff(Fc);
DFxDt=diff(Fx);
DFyDt=diff(Fy);

F1=formula(subs(DFxDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]))
F2=formula(subs(DFyDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]))
F3=formula(subs(DFcDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]))


```

```
D2FcDt2=diff(DFcDt);
D2FxDt2=diff(DFxDt);
D2FyDt2=diff(DFyDt);

F4=formula(subs(D2FcDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
    alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]))
F5=formula(subs(D2FxDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
    alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]))
F6=formula(subs(D2FyDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
    alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]))

Pi=[-0.1,-0.5,3,-0.5,0,0.5,-0.5,0,0.5]
 %Pi=[-0.0997,-0.4684,2.8737,0,0,0,0,0,0];
```

Posición

```
fP(r_1,theta_1,theta_3)=[fc;fx;fy];
fP=subs(fP,[theta_2,r_1y,r_2,r_3],[Vtheta_2,Vr_1y,Vr_2,Vr_3]);
fP=simplify(fP,'Steps',5);

valTheta_3=subs(SolTheta_3,[theta_2,r_2,r_3,r_1y],[Vtheta_2,Vr_2,Vr_3,Vr_1y]);
ValTheta_1=subs(SolTheta_1,[theta_2,r_2,r_3,r_1y,theta_3],[Vtheta_2,Vr_2, ...
    Vr_3,Vr_1y,valTheta_3]);
valR_1=subs(SolR_1,[r_1y,theta_1],[Vr_1y,ValTheta_1]);

JacP=jacobian(fP, [r_1,theta_1,theta_3]);
JacInvP=JacP^-1;


pP=[Pi(1),Pi(2),Pi(3)]';
ValoresPNR(1,:)=pP';
iteraciones=15;
for i=1:iteraciones
    pP=double(pP-JacInvP(pP(1),pP(2),pP(3))*fP(pP(1),pP(2),pP(3)));
    ValoresPNR(i+1,:)=double(pP)';
    ErrorNRP(i,:)=abs(ValoresPNR(i,:)-ValoresPNR(i+1,:));
    NormErrorNRP(i)=norm(ErrorNRP(i,:));
     if NormErrorNRP(i)<0.000001
        break;
    end
end
SolPosNR=double(pP);


```

Velocidad

```
fV(V_r,omega_1,omega_3)=[F1;F2;F3];
fV=subs(fV,[theta_2,r_1y,r_2,r_3,r_1,theta_1,theta_3,omega_2],[Vtheta_2, ...
    Vr_1y,Vr_2,Vr_3,SolPosNR(1),SolPosNR(2),SolPosNR(3),Vomega_2]);
```

```
fV=simplify(fV,'Steps',5);

JacV=jacobian(fV, [V_r,omega_1,omega_3]);
JacInvV=JacV^-1;

pV=[Pi(4),Pi(5),Pi(6)]';
ValoresVNR(1,:)=pV';

iteraciones=15;
for i=1:iteraciones
    pV=double(pV-JacInvV(pV(1),pV(2),pV(3))*fV(pV(1),pV(2),pV(3)));
    ValoresVNR(i+1,:)=double(pV)';
    ErrorNRV(i,:)=abs(ValoresVNR(i,:)-ValoresVNR(i+1,:));
    NormErrorNRV(i)=norm(ErrorNRV(i,:));
     if NormErrorNRV(i)<0.000001
        break;
    end
end

SolVelNR=double(pV);
```

Aceleración

```
fA(a_r,alpha_1,alpha_3)=[F4;F5;F6];
fA=subs(fA,[theta_2,r_1y,r_2,r_3,r_1,theta_1,theta_3,omega_2,V_r,omega_1, ...
    omega_3,alpha_2],[Vtheta_2,Vr_1y,Vr_2,Vr_3,SolPosNR(1),SolPosNR(2), ...
    SolPosNR(3),Vomega_2,SolVelNR(1),SolVelNR(2),SolVelNR(3),Valpha_2]);
fA=simplify(fA,'Steps',5);
JacA=jacobian(fA, [a_r,alpha_1,alpha_3]);
JacInvA=JacA^-1;

PiA=[Pi(7),Pi(8),Pi(9)]';
ValoresANR(1,:)=PiA';

iteraciones=15;
for i=1:iteraciones
    PiA=double(PiA-JacInvA(PiA(1),PiA(2),PiA(3))*fA(PiA(1),PiA(2),PiA(3)));
    ValoresANR(i+1,:)=double(PiA)';
    ErrorNRA(i,:)=abs(ValoresANR(i,:)-ValoresANR(i+1,:));
    NormErrorNRA(i)=norm(ErrorNRA(i,:));
     if NormErrorNRP(i)<0.000001
        break;
    end
end
SolAceNR=double(PiA);
```

Gráficas

```
figure()
plot(0:(length(ValoresPNR(:,1))-1),ValoresPNR)
title('Solucion posicion metodo de Newton Raphson')
legend("r_1","\theta_1","\theta_3","Location","best")
ylabel('Solucion')
xlabel('Indice')
xticks(0:length(ValoresPNR(:,1)))
grid on

figure()
```

```
subplot(2,1,1)
plot(ErrorNRP)
title('Error de posicion metodo de Newton Raphson')
legend("r_1","\theta_1","\theta_3","Location","best")
ylabel('Error')
xlabel('Iteraciones')
xticks(1:(length(ValoresPNR(:,1))-1))
grid on
subplot(2,1,2)
plot(NormErrorNRP)
title('Error total posicion metodo de Newton Raphson')
xlabel('Iteraciones')
xticks(1:(length(ValoresPNR(:,1))-1))
grid on


figure()
plot(0:(length(ValoresVNR(:,1))-1),ValoresVNR)
title('Solucion velocidad metodo de Newton Raphson')
legend("V_r","\omega_1","\omega_3","Location","best")
ylabel('Solucion')
xlabel('Indice')
xticks(0:length(ValoresVNR(:,1)))
grid on


figure()
subplot(2,1,1)
plot(ErrorNRV)
title('Error de velocidad metodo de Newton Raphson')
legend("V_r","\omega_1","\omega_3","Location","best")
ylabel('Error')
xlabel('Iteraciones')
xticks(1:(length(ValoresVNR(:,1))-1))
grid on
subplot(2,1,2)
plot(NormErrorNRV)
title('Error total velocidad metodo de Newton Raphson')
xlabel('Iteraciones')
xticks(1:(length(ValoresVNR(:,1))-1))
grid on

figure()
plot(0:(length(ValoresANR(:,1))-1),ValoresANR)
title('Solucion aceleracion metodo de Newton Raphson')
legend("a_r","\alpha_1","\alpha_3","Location","best")
ylabel('Solucion')
xlabel('Indice')
xticks(0:length(ValoresANR(:,1)))
grid on

figure()
subplot(2,1,1)
plot(ErrorNRA)
title('Error de aceleracion metodo de Newton Raphson')
legend("a_r","\alpha_1","\alpha_3","Location","best")
ylabel('Error')
xlabel('Iteraciones')
```

```
xticks(1:(length(ValoresANR(:,1))-1))
grid on
subplot(2,1,2)
plot(NormErrorNRA)
title('Error total aceleracion metodo de Newton Raphson')
xlabel('Iteraciones')
xticks(1:(length(ValoresANR(:,1))-1))
grid on

format long
Solucion=[SolPosNR;SolVelNR;SolAceNR]
```

E.3. Solución por Newton Raphson simultaneo

```
clear
clc
syms t Theta_1(t) Theta_2(t) Theta_3(t) R_1(t)


syms r_1 r_1y r_2 r_3 x

syms V_r omega_1 omega_2 omega_3 theta_1 theta_2 theta_3

syms a_r alpha_1 alpha_2 alpha_3

Vr_2=0.075
Vr_3=0.17
Vr_1y=0.045

Vtheta_2=0
Vomega_2=0
Valpha_2=0

Fc=sin(Theta_1)*R_1-r_1y;
Fc=simplify(Fc,'Steps',5)
fc=subs(Fc,[Theta_1, R_1],[theta_1, r_1])
SolR_1=solve(fc,r_1)
Fx=-R_1*cos(Theta_1)+r_2*cos(Theta_2)+r_3*cos(Theta_3);
Fx=simplify(Fx,'Steps',5)
fx=formula(subs(Fx,[Theta_1, Theta_2, Theta_3, R_1],[theta_1, theta_2, ...
    theta_3, SolR_1]))
fx=simplify(fx,'Steps',5)
SolTheta_1=solve(fx,theta_1);
Fy=R_1*sin(Theta_1)-r_2*sin(Theta_2)-r_3*sin(Theta_3);
Fy=simplify(Fy,'Steps',5);
fy=formula(subs(Fy,[Theta_1, Theta_2, Theta_3, R_1],[theta_1, theta_2, ...
    theta_3, SolR_1]))
SolTheta_3=solve(fy,theta_3);
SolTheta_3=SolTheta_3(1);

DFcDt=diff(Fc);
DFxDt=diff(Fx);
DFyDt=diff(Fy);

F1=formula(subs(DFxDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]))
F2=formula(subs(DFyDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]))
F3=formula(subs(DFcDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]))


D2FcDt2=diff(DFcDt);
D2FxDt2=diff(DFxDt);
D2FyDt2=diff(DFyDt);

```

```
F4=formula(subs(D2FcDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1),
iff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1,theta_2, ...
theta_3,r_1]))
F5=formula(subs(D2FxDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
    alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1,theta_2, ...
    theta_3,r_1]))
F6=formula(subs(D2FyDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
    alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1,theta_2, ...
    theta_3,r_1]))


f(r_1,theta_1,theta_3,V_r,omega_1,omega_3,a_r,alpha_1,alpha_3)=[fc;fx;fy;
    F1;F2;F3;F4;F5;F6]
f=subs(f,[theta_2,r_1y,r_2,r_3,omega_2,alpha_2],[Vtheta_2,Vr_1y,Vr_2, ...
    Vr_3,Vomega_2,Valpha_2]);
f=simplify(f,'Steps',5);

Jac=jacobian(f, ...
   [r_1,theta_1,theta_3,V_r,omega_1,omega_3,a_r,alpha_1,alpha_3])
JacInv=Jac^-1




p1=[-0.1,-0.5,3,-0.5,0,0.5,-0.5,0,0.5]'
 %p1=[-0.0997,-0.4684,2.8737,0,0,0,0,0,0]';

ValoresNR(1,:)=p1';

iteraciones=15;
for i=1:iteraciones
    p1=double(p1-JacInv(p1(1),p1(2),p1(3),p1(4),p1(5),p1(6),p1(7),p1(8), ...
        p1(9))*f(p1(1),p1(2),p1(3),p1(4),p1(5),p1(6),p1(7),p1(8),p1(9)))
    ValoresNR(i+1,:)=double(p1)';
    ErrorNR(i,:)=abs(ValoresNR(i,:)-ValoresNR(i+1,:));
    NormErrorNR(i)=norm(ErrorNR(i,:));
     if NormErrorNR(i)<0.000001
        break;
    end
end
figure()
plot(0:(length(ValoresNR(:,1))-1),ValoresNR)
title('Solucion metodo conjunto de Newton Raphson')
legend("r_1","\theta_1","\theta_3","V_r","\omega_1","\omega_3","a_r", ...
    "\alpha_1","\alpha_3","Location","best")

ylabel('Solucion')
xlabel('Indice')
xticks(0:length(ValoresNR))
grid on
figure()
subplot(2,1,1)
```

```
plot(ErrorNR)
title('Error absoluto metodo conjunto de Newton Raphson')
legend("r_1","\theta_1","\theta_3","V_r","\omega_1","\omega_3","a_r", ...
    "\alpha_1","\alpha_3","Location","east")
ylabel('Error')
xlabel('Iteraciones')
xticks(1:(length(ValoresNR)-1))
grid on
subplot(2,1,2)
plot(NormErrorNR)
title('Error total metodo conjunto de Newton Raphson')
xlabel('Iteraciones')
xticks(1:(length(ValoresNR)-1))
grid on



format long
SolNR=double(p1)
ErrorNRFinal=double(NormErrorNR(end))
```

E.4. Solución por método de Newton Raphson

```
clear
clc
Desarrollo teórico
syms Theta_1(t) Theta_2(t) Theta_3(t) R_1(t) t
syms r_1 r_1y r_2 r_3
syms theta_1 theta_2 theta_3
syms V_r omega_1 omega_2 omega_3
syms a_r alpha_1 alpha_2 alpha_3

Vr_2=0.075
Vr_3=0.17
Vr_1y=0.045

Fc=sin(Theta_1)*R_1-r_1y;
fc=subs(Fc,[Theta_1, R_1],[theta_1, r_1]);

Fx=-R_1*cos(Theta_1)+r_2*cos(Theta_2)+r_3*cos(Theta_3);
Fx=simplify(Fx,'Steps',5);
fx=formula(subs(Fx,[Theta_1, Theta_2, Theta_3, R_1],[theta_1, theta_2, ...
    theta_3, r_1]));

Fy=R_1*sin(Theta_1)-r_2*sin(Theta_2)-r_3*sin(Theta_3);
Fy=simplify(Fy,'Steps',5);
fy=formula(subs(Fy,[Theta_1, Theta_2, Theta_3, R_1],[theta_1, theta_2, ...
    theta_3, r_1]));

DFcDt=diff(Fc);
DFxDt=diff(Fx);
DFyDt=diff(Fy);
F1=formula(subs(DFxDt,[diff(R_1),diff(Theta_1),diff(Theta_2),diff ...
    (Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]));
F2=formula(subs(DFyDt,[diff(R_1),diff(Theta_1),diff(Theta_2),diff ...
    (Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]));
F3=formula(subs(DFcDt,[diff(R_1),diff(Theta_1),diff(Theta_2),diff ...
    (Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]));

D2FcDt2=diff(DFcDt);
D2FxDt2=diff(DFxDt);
D2FyDt2=diff(DFyDt);
F4=formula(subs(D2FcDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1], ...
    [a_r,alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]));
F5=formula(subs(D2FxDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1], ...
    [a_r,alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]));
F6=formula(subs(D2FyDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
```

```
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1], ...
    [a_r,alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]));

Tabla_Cinematica=readtable("Perfil_cinematica.txt","Delimiter"," ");
```

Solución NR Simultaneo

```
f(r_1,theta_1,theta_3,V_r,omega_1,omega_3,a_r,alpha_1,alpha_3)=
[fc;fx;fy;F1;F2;F3;F4;F5;F6];
f=subs(f,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
f=simplify(f,'Steps',5);
JacInv=jacobian(f, [r_1,theta_1,theta_3,V_r,omega_1,omega_3,a_r, ...
    alpha_1,alpha_3])^-1;


p1=[-0.0997,-0.4684,2.8737,0,0,0,0,0,0];

for i=1:length(Tabla_Cinematica.Tiempo)
    tic
    [p1,error_NR(i)]=NR_Sistema(f,JacInv,p1,Tabla_Cinematica.Posicion(i), ...
        Tabla_Cinematica.Velocidad(i),Tabla_Cinematica.Aceleracion(i));
    ValoresSolucion_NR(i,:)=p1;
    tiepoEjeNR(i)=toc;
end

for i=1:length(tiepoEjeNR)
    tiempoTotalNr(i)=sum(tiepoEjeNR(1:i));
end

Tabla_Cinematica.R_1=ValoresSolucion_NR(:,1);
Tabla_Cinematica.theta_1=ValoresSolucion_NR(:,2);
Tabla_Cinematica.theta_3=ValoresSolucion_NR(:,3);
Tabla_Cinematica.V_1=ValoresSolucion_NR(:,4);
Tabla_Cinematica.omega_1=ValoresSolucion_NR(:,5);
Tabla_Cinematica.omega_3=ValoresSolucion_NR(:,6);
Tabla_Cinematica.a_1=ValoresSolucion_NR(:,7);
Tabla_Cinematica.alpha_1=ValoresSolucion_NR(:,8);
Tabla_Cinematica.alpha_3=ValoresSolucion_NR(:,9);
writetable(Tabla_Cinematica,'Resultados.txt','delimiter',' ', ...
    'WriteVariableNames', true)
```

Solución NR Consecutivo

```
p1=[-0.0997,-0.4684,2.8737,0,0,0,0,0,0];
fP(r_1,theta_1,theta_3)=[fc;fx;fy];
fP=subs(fP,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
fP=simplify(fP,'Steps',5);
JacP=jacobian(fP, [r_1,theta_1,theta_3]);
JacInvP=JacP^-1;
fV(V_r,omega_1,omega_3)=[F1;F2;F3];
fV=subs(fV,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
fV=simplify(fV,'Steps',5)
JacV=jacobian(fV, [V_r,omega_1,omega_3]);
JacInvV=JacV^-1;
fA(a_r,alpha_1,alpha_3)=[F4;F5;F6];
fA=subs(fA,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
fA=simplify(fA,'Steps',5);
JacA=jacobian(fA, [a_r,alpha_1,alpha_3]);
```

```
JacInvA=JacA^-1;

for i=1:length(Tabla_Cinematica.Tiempo)
    tic
    [p1,Error_NR_EX(i)]= ...
       NR_Sistema_extenso(fP,JacInvP,fV,JacInvV,fA,JacInvA,p1, ...
        Tabla_Cinematica.Posicion(i),Tabla_Cinematica.Velocidad(i), ...
        Tabla_Cinematica.Aceleracion(i));
    ValoresSolucion_NR_EX(i,:)=p1;
    tiempoEje_EX(i)=toc;
end

tabla_Cinematica_NR_Ext=array2table([Tabla_Cinematica.Tiempo, ...
    Tabla_Cinematica.Aceleracion,Tabla_Cinematica.Velocidad, ...
    Tabla_Cinematica.Posicion,ValoresSolucion_NR_EX]);
tabla_Cinematica_NR_Ext.Properties.VariableNames=
Tabla_Cinematica.Properties.VariableNames()
writetable(tabla_Cinematica_NR_Ext,'Resultados_NR_Extendido.txt', ...
    'delimiter',' ', 'WriteVariableNames', true)

for i=1:length(tiempoEje_EX)
    tiempoTotalNrEX(i)=sum(tiempoEje_EX(1:i));
end
Tiempo_NR=array2table([tiempoTotalNr',tiempoTotalNrEX',tiepoEjeNR', ...
    tiempoEje_EX']);
Tiempo_NR.Properties.VariableNames=["Tiempo_total_NR","Tiempo_total_NR_EX",
    "tiempo_puntos_NR","tiempo_puntos_NR_EX"]
writetable(Tiempo_NR,'Tiempor_NR.txt','delimiter',' ', ...
   'WriteVariableNames', ...
    true)

Error_NR=array2table([error_NR',Error_NR_EX']);
Error_NR.Properties.VariableNames=["Error_NR","Error_NR_EX"]
writetable(Error_NR,'Error_NR.txt','delimiter',' ', 'WriteVariableNames', ...
    true)
```

Gráficas de resultados

Tabla de resultados

```
TablaError=readtable("Error_NR.txt");
Tabla_Cinematica=readtable("Resultados.txt","Delimiter",' ');
tabla_Cinematica_NR_Ext=readtable('Resultados_NR_Extendido.txt', ...
    'delimiter',' ');
tiempo=Tabla_Cinematica.Tiempo;
```

NR simultaneo

```
fprintf("Graficas de resultados del metodo de NR simultaneo")
figure()
subplot(3,1,1)
plot(tiempo,Tabla_Cinematica.R_1,"r")
title("r_1 vs tiempo")
xlabel("t (s)")
ylabel("r_1 (m)")
grid on
xlim([0 tiempo(end)])
```

```
subplot(3,1,2)
plot(tiempo,Tabla_Cinematica.theta_1*180/pi,"b")
title("\theta_1 vs tiempo")
xlabel("t (s)")
ylabel("\theta_1 (\circ)")
grid on
xlim([0 tiempo(end)])
yticks(-30:5:-10)

subplot(3,1,3)
plot(tiempo,Tabla_Cinematica.theta_3*180/pi,"k")
title("\theta_3 vs tiempo")
xlabel("t (s)")
ylabel("\theta_3 (\circ)")
grid on
xlim([0 tiempo(end)])
ylim([125 200])
yticks(125:25:200)

figure()
subplot(3,1,1)
plot(tiempo,Tabla_Cinematica.V_1,"r")
title("v_1 vs tiempo")
xlabel("t (s)")
ylabel("v_1 (m s^{-1})")
grid on
xlim([0 tiempo(end)])
ylim([-0.6 0.6])
yticks(-0.6:0.3:0.6)

subplot(3,1,2)
plot(tiempo,Tabla_Cinematica.omega_1*180/pi,"b")
title("\omega_1 vs tiempo")
xlabel("t (s)")
ylabel("\omega_1 (\circ s^{-1})")
grid on
xlim([0 tiempo(end)])
ylim([-80 80])
yticks(-80:40:80)


subplot(3,1,3)
plot(tiempo,Tabla_Cinematica.omega_3*180/pi,"k")
title("\omega_3 vs tiempo")
xlabel("t (s)")
ylabel("\omega_3 (\circ s^{-1})")
grid on
xlim([0 tiempo(end)])
yticks([-200:100:200])

figure()
subplot(3,1,1)
plot(tiempo,Tabla_Cinematica.a_1,"r")
title("a_1 vs tiempo")
xlabel("t (s)")
ylabel("a_1 (m s^{-2})")
grid on
xlim([0 tiempo(end)])
```

```
yticks(-5:2.5:5)

subplot(3,1,2)
plot(tiempo,Tabla_Cinematica.alpha_1*180/pi,"b")
title("\alpha_1 vs tiempo")
xlabel("t (s)")
ylabel("\alpha_1 (\circ s^{-2})")
grid on
xlim([0 tiempo(end)])
ylim([-400 800])
yticks([-400:400:800])

subplot(3,1,3)
plot(tiempo,Tabla_Cinematica.alpha_3*180/pi,"k")
title("\alpha_3 vs tiempo")
xlabel("t (s)")
ylabel("\alpha_3 (\circ s^{-2})")
grid on
xlim([0 tiempo(end)])
ylim([-1500 1500])
yticks(-1500:750:1500)


figure()
subplot(2,1,1)
plot(tiempo,TablaError.Error_NR)
title("Error del metodo por Newton Raphson simultaneo")
xlim([0 tiempo(end)])
grid on
```

NR consecutivo

```
fprintf("Graficas de resultados del metodo de NR consecutivo")
figure()
subplot(3,1,1)
plot(tiempo,tabla_Cinematica_NR_Ext.R_1,"r--")
title("r_1 vs tiempo")
xlabel("t (s)")
ylabel("r_1 (m)")
grid on
xlim([0 tiempo(end)])

subplot(3,1,2)
plot(tiempo,tabla_Cinematica_NR_Ext.theta_1*180/pi,"b--")
title("\theta_1 vs tiempo")
xlabel("t (s)")
ylabel("\theta_1 (\circ)")
grid on
xlim([0 tiempo(end)])
yticks(-30:5:-10)

subplot(3,1,3)
plot(tiempo,tabla_Cinematica_NR_Ext.theta_3*180/pi,"k--")
title("\theta_3 vs tiempo")
xlabel("t (s)")
ylabel("\theta_3 (\circ)")
grid on
xlim([0 tiempo(end)])
```

```
ylim([125 200])
yticks(125:25:200)

figure()
subplot(3,1,1)
plot(tiempo,tabla_Cinematica_NR_Ext.V_1,"r--")
title("v_1 vs tiempo")
xlabel("t (s)")
ylabel("v_1 (m s^{-1})")
grid on
xlim([0 tiempo(end)])
ylim([-0.6 0.6])
yticks(-0.6:0.3:0.6)

subplot(3,1,2)
plot(tiempo,tabla_Cinematica_NR_Ext.omega_1*180/pi,"b--")
title("\omega_1 vs tiempo")
xlabel("t (s)")
ylabel("\omega_1 (\circ s^{-1})")
grid on
xlim([0 tiempo(end)])
ylim([-80 80])
yticks(-80:40:80)


subplot(3,1,3)
plot(tiempo,tabla_Cinematica_NR_Ext.omega_3*180/pi,"k--")
title("\omega_3 vs tiempo")
xlabel("t (s)")
ylabel("\omega_3 (\circ s^{-1})")
grid on
xlim([0 tiempo(end)])
yticks([-200:100:200])

figure()
subplot(3,1,1)
plot(tiempo,tabla_Cinematica_NR_Ext.a_1,"r--")
title("a_1 vs tiempo")
xlabel("t (s)")
ylabel("a_1 (m s^{-2})")
grid on
xlim([0 tiempo(end)])
yticks(-5:2.5:5)

subplot(3,1,2)
plot(tiempo,tabla_Cinematica_NR_Ext.alpha_1*180/pi,"b--")
title("\alpha_1 vs tiempo")
xlabel("t (s)")
ylabel("\alpha_1 (\circ s^{-2})")
grid on
xlim([0 tiempo(end)])
ylim([-400 800])
yticks([-400:400:800])

subplot(3,1,3)
plot(tiempo,tabla_Cinematica_NR_Ext.alpha_3*180/pi,"k--")
title("\alpha_3 vs tiempo")
xlabel("t (s)")
```

```
ylabel("\alpha_3 (\circ s^{-2})")
grid on
xlim([0 tiempo(end)])
ylim([-1500 1500])
yticks(-1500:750:1500)

figure()
subplot(2,1,1)
plot(tiempo,TablaError.Error_NR_EX)
title("Error del metodo por Newton Raphson consecutivo")
xlim([0 tiempo(end)])
grid on


```

Comparación entre ambas versiones de Newton Raphson

Error Resultados

```
ValoresSolucion_NR=table2array(Tabla_Cinematica);
ValoresSolucion_NR=ValoresSolucion_NR(:,(2:end));
ValoresSolucion_NR_EX=table2array(tabla_Cinematica_NR_Ext);
ValoresSolucion_NR_EX=ValoresSolucion_NR_EX(:,(2:end));

for i=1:length(ValoresSolucion_NR)
    Error_M(i)=norm(abs(ValoresSolucion_NR(i,:)-ValoresSolucion_NR_EX(i,:)));
end
figure()
subplot(2,1,1)
plot(Tabla_Cinematica.Tiempo,Error_M)
writematrix(Error_M,"Error_entre_NR")
xlim([0 max(Tabla_Cinematica.Tiempo)])
title("Error entre ambos metodos NR")
xlabel("tiempo (s)")
ylabel("Error")
grid on

```

Tiempo de ejecución

```
tablaTiempo=readtable("Tiempor_NR.txt");
tiempoTotalNr=tablaTiempo.Tiempo_total_NR;
tiempoTotalNrEX=tablaTiempo.Tiempo_total_NR_EX;
figure()
plot(tiempoTotalNr)
hold on
plot(tiempoTotalNrEX)
hold off
title("Tiempo de ejecucion vs Numero de puntos")
legend("NR Simultaneo","NR Consecutivo","Location","best")
grid on
ylabel("t (s)")
xlabel("Numero de puntos")
Diferencia=abs(tiempoTotalNr-tiempoTotalNrEX);
DiferenciaRelativa=Diferencia./tiempoTotalNrEX*100;
figure()
subplot(2,1,1)
plot(Diferencia)
title("Diferencia temporal entre metodos")
xlabel("Numero de puntos")
ylabel("t (s)")
```

```
yticks(0:20:100)
grid on
subplot(2,1,2)
plot(DiferenciaRelativa)
title("Porcentaje de optimizacion")
ylabel("Porcentaje (%)")
xlabel("Numero de puntos")
ylim([0 30])
yticks(0:6:30)
grid on

function [Sol,Error]= NR_Sistema(F,FJacInv,Pi,Ftheta_2,Fomega_2,Falpha_2)
syms omega_2 theta_2 alpha_2
f=subs(F,[theta_2,omega_2,alpha_2],[Ftheta_2,Fomega_2,Falpha_2]);
JacInv=subs(FJacInv,[theta_2,omega_2,alpha_2],[Ftheta_2,Fomega_2,Falpha_2]);
p=Pi';
ValoresNR(1,:)=p';
iteraciones=50;
for i=1:iteraciones
    p=double(p-JacInv(p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8),p(9))*f(p(1), ...
        ...
        p(2),p(3),p(4),p(5),p(6),p(7),p(8),p(9)));
    ValoresNR(i+1,:)=double(p)';
    ErrorNR(i,:)=abs(ValoresNR(i,:)-ValoresNR(i+1,:));
    NormErrorNR(i)=norm(ErrorNR(i,:));
     if NormErrorNR(i)<0.000001
        break;
    end
end
    Error=NormErrorNR(end);
    Sol=p';
end


function [Sol,Error]= ...
   NR_Sistema_extenso(fP,JacInvP,fV,JacInvV,fA,JacInvA,Pi,Vtheta_2, ...
    Vomega_2,Valpha_2)
syms  r_1y r_1 r_2 r_3 theta_3 theta_2 theta_1 V_r omega_1 omega_2 ...
   omega_3 a_r alpha_1 alpha_2 alpha_3
fP=subs(fP,theta_2,Vtheta_2);
PiP=[Pi(1),Pi(2),Pi(3)]';
ValoresNR(1,:)=PiP';
iteraciones=50;
for i=1:iteraciones
    PiP=double(PiP-JacInvP(PiP(1),PiP(2),PiP(3))*fP(PiP(1), ...
        PiP(2),PiP(3)));
    ValoresNR(i+1,:)=double(PiP)';
    ErrorNR_P(i,:)=abs(ValoresNR(i,:)-ValoresNR(i+1,:));
    NormErrorNR_P(i)=norm(ErrorNR_P(i,:));
     if NormErrorNR_P(i)<0.000001
        break;
    end
end
SolPosNR=PiP;
ErrorP=NormErrorNR_P(end);
fV=subs(fV,[theta_2,r_1,theta_1,theta_3,omega_2],[Vtheta_2,SolPosNR(1), ...
    SolPosNR(2),SolPosNR(3),Vomega_2]);
JacInvV=subs(JacInvV,[theta_2,r_1,theta_1,theta_3,omega_2],[Vtheta_2, ...
```

```
    SolPosNR(1),SolPosNR(2),SolPosNR(3),Vomega_2]);
PiV=[Pi(4),Pi(5),Pi(6)]';
ValoresVNR(1,:)=PiV';
iteraciones=50;
for i=1:iteraciones
    PiV=double(PiV-JacInvV(PiV(1),PiV(2),PiV(3))*fV(PiV(1),PiV(2),PiV(3)));
    ValoresVNR(i+1,:)=double(PiV)';
    ErrorNR_V(i,:)=abs(ValoresVNR(i,:)-ValoresVNR(i+1,:));
    NormErrorNR_V(i)=norm(ErrorNR_V(i,:));
     if NormErrorNR_V(i)<0.000001
        break;
    end
end
SolVelNR=double(PiV);
ErrorV=NormErrorNR_V(end);
fA=subs(fA,[theta_2,r_1,theta_1,theta_3,omega_2,V_r,omega_1,omega_3, ...
    alpha_2],[Vtheta_2,SolPosNR(1),SolPosNR(2),SolPosNR(3),Vomega_2, ...
    SolVelNR(1),SolVelNR(2),SolVelNR(3),Valpha_2]);
JacInvA=subs(JacInvA,[theta_2,r_1,theta_1,theta_3,omega_2,V_r,omega_1, ...
    omega_3,alpha_2],[Vtheta_2,SolPosNR(1),SolPosNR(2),SolPosNR(3), ...
    Vomega_2,SolVelNR(1),SolVelNR(2),SolVelNR(3),Valpha_2]);
PiA=[Pi(7),Pi(8),Pi(9)]';
ValoresVNR(1,:)=PiA';
iteraciones=50;
for i=1:iteraciones
    PiA=double(PiA-JacInvA(PiA(1),PiA(2),PiA(3))*fA(PiA(1),PiA(2),PiA(3)));
    ValoresVNR(i+1,:)=double(PiA)';
    ErrorNR_A(i,:)=abs(ValoresVNR(i,:)-ValoresVNR(i+1,:));
    NormErrorNR_A(i)=norm(ErrorNR_A(i,:));
     if NormErrorNR_A(i)<0.000001
        break;
    end
end
SolAceNR=double(PiA);
ErrorA=NormErrorNR_A(end);
Sol=[SolPosNR',SolVelNR',SolAceNR'];
Error=norm([ErrorP,ErrorV,ErrorA]);
end
```

E.5. Solución analítica

```
clear
clc

syms Theta_1(t) Theta_2(t) Theta_3(t) R_1(t) t
syms r_1 r_1y r_2 r_3
syms theta_1 theta_2 theta_3
syms V_r omega_1 omega_2 omega_3
syms a_r alpha_1 alpha_2 alpha_3

Vr_2=0.075
Vr_3=0.17
Vr_1y=0.045
Vtheta_2=0
Vomega_2=0
Valpha_2=0

Fc=sin(Theta_1)*R_1-r_1y;
fc=formula(subs(Fc,[Theta_1, R_1],[theta_1, r_1]));

Fx=-R_1*cos(Theta_1)+r_2*cos(Theta_2)+r_3*cos(Theta_3);
Fx=simplify(Fx,'Steps',5);
fx=formula(subs(Fx,[Theta_1, Theta_2, Theta_3, R_1],[theta_1, theta_2, ...
    theta_3, r_1]));

Fy=R_1*sin(Theta_1)-r_2*sin(Theta_2)-r_3*sin(Theta_3);
Fy=simplify(Fy,'Steps',5);
fy=formula(subs(Fy,[Theta_1, Theta_2, Theta_3, R_1],[theta_1, theta_2, ...
    theta_3, r_1]));

DFcDt=diff(Fc);
DFxDt=diff(Fx);
DFyDt=diff(Fy);
F1=formula(subs(DFxDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]));
F2=formula(subs(DFyDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]));
F3=formula(subs(DFcDt,[diff(R_1),diff(Theta_1),diff(Theta_2), ...
    diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[V_r,omega_1,omega_2, ...
    omega_3,theta_1,theta_2,theta_3,r_1]));

D2FcDt2=diff(DFcDt);
D2FxDt2=diff(DFxDt);
D2FyDt2=diff(DFyDt);
F4=formula(subs(D2FcDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
    alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]));
F5=formula(subs(D2FxDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
    alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]));
```

```
F6=formula(subs(D2FyDt2,[diff(diff(R_1)),diff(diff(Theta_1)), ...
    diff(diff(Theta_2)),diff(diff(Theta_3)),diff(R_1),diff(Theta_1), ...
    diff(Theta_2),diff(Theta_3),Theta_1,Theta_2,Theta_3,R_1],[a_r, ...
    alpha_1,alpha_2,alpha_3,V_r,omega_1,omega_2,omega_3,theta_1, ...
    theta_2,theta_3,r_1]));


SolR_1=solve(fc,r_1)
SolTheta_1=solve(simplify(subs(fx,r_1,SolR_1),"Steps",15),theta_1)
SolTheta_3=solve(fy-fc,theta_3);
SolTheta_3=SolTheta_3(2)


SolOmega_3=solve(F3-F2,omega_3)

SolOmega_1=solve(F3,omega_1);
SolOmega_1=simplify(SolOmega_1,"Steps",10)

SolV_r=solve(subs(F1,omega_1,SolOmega_1),V_r);
SolV_r=simplify(SolV_r,"Steps",10)

SolAlpha_3=solve(F6-F4,alpha_3);
SolAlpha_3=simplify(SolAlpha_3,"Steps",20)

SolAlpha_1=solve(F4,alpha_1);
SolAlpha_1=simplify(SolAlpha_1,"Steps",20)

SolA_r=solve(subs(F5,alpha_1,SolAlpha_1),a_r);
SolA_r=simplify(SolA_r,"Steps",20)


SolR_1=subs(SolR_1,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
SolTheta_1=subs(SolTheta_1,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
SolTheta_3=subs(SolTheta_3,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
SolV_r=subs(SolV_r,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
SolOmega_1=subs(SolOmega_1,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
SolOmega_3=subs(SolOmega_3,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
SolA_r=subs(SolA_r,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
SolAlpha_1=subs(SolAlpha_1,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);
SolAlpha_3=subs(SolAlpha_3,[r_1y,r_2,r_3],[Vr_1y,Vr_2,Vr_3]);



Tabla_Cinematica=readtable("Perfil_cinematica.txt","Delimiter"," ");
for i=1:length(Tabla_Cinematica.Tiempo)
    Sol=Fanalitica(SolR_1,SolTheta_1,SolTheta_3,SolV_r,SolOmega_1, ...
        SolOmega_3,SolA_r,SolAlpha_1,SolAlpha_3, ...
        Tabla_Cinematica.Posicion(i),Tabla_Cinematica.Velocidad(i), ...
        Tabla_Cinematica.Aceleracion(i));
    ValoresSolucion_NR(i,:)=Sol;
end

Tabla_Cinematica.R_1=ValoresSolucion_NR(:,1);
Tabla_Cinematica.theta_1=ValoresSolucion_NR(:,2);
Tabla_Cinematica.theta_3=ValoresSolucion_NR(:,3);
Tabla_Cinematica.V_1=ValoresSolucion_NR(:,4);
Tabla_Cinematica.omega_1=ValoresSolucion_NR(:,5);
Tabla_Cinematica.omega_3=ValoresSolucion_NR(:,6);
```

```
112 Tabla_Cinematica.a_1=ValoresSolucion_NR(:,7);
113 Tabla_Cinematica.alpha_1=ValoresSolucion_NR(:,8);
114 Tabla_Cinematica.alpha_3=ValoresSolucion_NR(:,9);
115 writetable(Tabla_Cinematica,'ResultadosAnaliticos.txt','delimiter',' ', ...
        'WriteVariableNames', true)
116
117
118 function ...
        solAnalitica=Fanalitica(SolR_1,SolTheta_1,SolTheta_3,SolV_r,SolOmega_1, ...
        ...
119     SolOmega_3,SolA_r,SolAlpha_1,SolAlpha_3,Vtheta_2,Vomega_2,Valpha_2)
120 syms r_1 theta_1 theta_2 theta_3 V_r omega_1 omega_2 omega_3 a_r alpha_1 ...
        alpha_2 alpha_3
121 Vtheta_3=double(subs(SolTheta_3,theta_2,Vtheta_2));
122 Vtheta_1=double(subs(SolTheta_1,[theta_2,theta_3],[Vtheta_2,Vtheta_3]));
123 Vr_1=double(subs(SolR_1,theta_1,Vtheta_1));
124 Vomega_3=double(subs(SolOmega_3,[theta_2,theta_3,omega_2],[Vtheta_2, ...
125     Vtheta_3,Vomega_2]));
126 Vv_r=double(subs(SolV_r,[theta_1,theta_2,theta_3,omega_2,omega_3], ...
127     [Vtheta_1,Vtheta_2,Vtheta_3,Vomega_2,Vomega_3]));
128 Vomega_1=double(subs(SolOmega_1,[r_1,theta_1,V_r],[Vr_1,Vtheta_1,Vv_r]));
129 Valpha_3=double(subs(SolAlpha_3,[theta_2,theta_3,omega_2, ...
130     omega_3,alpha_2],[Vtheta_2,Vtheta_3,Vomega_2,Vomega_3,Valpha_2]));
131 Va_r=double(subs(SolA_r,[r_1,theta_1,theta_2,theta_3,omega_1,omega_2, ...
132     omega_3,alpha_2,alpha_3],[Vr_1,Vtheta_1,Vtheta_2,Vtheta_3,Vomega_1, ...
133     Vomega_2,Vomega_3,Valpha_2,Valpha_3]));
134 Valpha_1=double(subs(SolAlpha_1,[r_1,theta_1,V_r,omega_1,a_r],[Vr_1, ...
135     Vtheta_1,Vv_r,Vomega_1,Va_r]));
136 solAnalitica=[Vr_1,Vtheta_1,Vtheta_3,Vv_r,Vomega_1,Vomega_3,Va_r,Valpha_1,
137     Valpha_3];
138 end
```

E.6. Resultados NR simultaneo vs solución analítica

```
clear
clc

Resultados_Analiticos=readtable("ResultadosAnaliticos.txt","Delimiter",' ');
Resultados_NR=readtable("Resultados.txt","Delimiter",' ');

ErrorR1=abs(Resultados_Analiticos.R_1-Resultados_NR.R_1);
ErrorTheta1=abs(Resultados_Analiticos.theta_1-Resultados_NR.theta_1);
ErrorTheta3=abs(Resultados_Analiticos.theta_3-Resultados_NR.theta_3);
ErrorV1=abs(Resultados_Analiticos.V_1-Resultados_NR.V_1);
ErrorW1=abs(Resultados_Analiticos.omega_1-Resultados_NR.omega_1);
ErrorW3=abs(Resultados_Analiticos.omega_3-Resultados_NR.omega_3);
ErrorA1=abs(Resultados_Analiticos.a_1-Resultados_NR.a_1);
ErrorAlpha1=abs(Resultados_Analiticos.alpha_1-Resultados_NR.alpha_1);
ErrorAlpha3=abs(Resultados_Analiticos.alpha_3-Resultados_NR.alpha_3);

figure()
subplot(3,1,1)
plot(ErrorR1)
title("Error valores de R_1")
ylim([0 10^-15])

subplot(3,1,2)
plot(ErrorTheta1)
title("Error valores \theta_1")
ylim([0 10^-15])

subplot(3,1,3)
plot(ErrorTheta3)
title("Error valores \theta_3")
ylim([0 10^-15])

figure()
subplot(3,1,1)
plot(ErrorV1)
title("Error valores V_r")
ylim([0 10^-15])

subplot(3,1,2)
plot(ErrorW1)
title("Error valores \omega_1")
ylim([0 10^-15])

subplot(3,1,3)
plot(ErrorW3)
title("Error valores \omega_3")
ylim([0 10^-15])

figure()
subplot(3,1,1)
plot(ErrorA1)
title("Error valores A_r")
ylim([0 10^-15])

subplot(3,1,2)
```

```
plot(ErrorAlpha1)
ylim([0 10^-15])
title("Error valores \alpha_1")

subplot(3,1,3)
plot(ErrorAlpha3)
title("Error valores \alpha_3")
ylim([0 10^-15])
```

E.7. Resultados NR simultaneo VS simulación

```
clear
clc
Tabla_Cinematica=readtable("Resultados.txt","Delimiter",' ')

t = xlsread("Resultados Inventor.xlsx");
TablaExcel=array2table(t);
TablaExcel.Properties.VariableNames=["Tiempo","Theta2","w2","A2","Theta3",
    "W3","A3","R1","Vr1","Ar1","Theta1","w1","A1"]

tiempo=Tabla_Cinematica.Tiempo;
ExTiempo=TablaExcel.Tiempo;

# Gráficas

# Perfil cinemático

Vmax=double(max(Tabla_Cinematica.Velocidad))*180/pi
figure()
subplot(3,1,1)
plot(tiempo,Tabla_Cinematica.Aceleracion*180/pi,"r-.")
hold on
plot(ExTiempo,TablaExcel.A2,"k--")
hold off
title('Aceleracion vs tiempo')
grid on
xlabel('t (s)')
ylabel('a (\circ s^{-2})')
xlim([0 ExTiempo(end)])

subplot(3,1,2)
plot(tiempo,Tabla_Cinematica.Velocidad*180/pi,"r-.")
hold on
plot(ExTiempo,TablaExcel.w2,"k--")
hold off
title('Velocidad vs tiempo')
grid on
xlabel('t (s)')
ylabel('v (\circ s^{-1})')
ylim([0 Vmax*1.1])
xlim([0 ExTiempo(end)])

subplot(3,1,3)
plot(tiempo,Tabla_Cinematica.Posicion*180/pi,"r-.")
hold on
plot(ExTiempo,TablaExcel.Theta2,"k--")
hold off
title('Posicion vs tiempo')
grid on
xlabel('t (s)')
ylabel('z (\circ)')
xlim([0 ExTiempo(end)])

# Gráficas de simulación y NR
figure()
```

```
55
56 subplot(3,1,1)
57 plot(tiempo,Tabla_Cinematica.R_1,"r-.")
58 hold on
59 plot(ExTiempo,-TablaExcel.R1,"k--")
60 hold off
61 title("r_1 vs tiempo")
62 xlabel("t (s)")
63 ylabel("r_1 (m)")
64 grid on
65 xlim([0 tiempo(end)])
66 legend("Newton Raphson","Simulacion","Location","east",'FontSize',7)
67
68 subplot(3,1,2)
69 plot(tiempo,Tabla_Cinematica.theta_1*180/pi,"r-.")
70 hold on
71 plot(ExTiempo,TablaExcel.Theta1,"k--")
72 hold off
73 title("\theta_1 vs tiempo")
74 xlabel("t (s)")
75 ylabel("\theta_1 (\circ)")
76 grid on
77 xlim([0 tiempo(end)])
78 yticks(-30:5:-10)
79
80 subplot(3,1,3)
81 plot(tiempo,Tabla_Cinematica.theta_3*180/pi,"r-.")
82 hold on
83 plot(ExTiempo,TablaExcel.Theta3+180,"k--")
84 hold off
85 title("\theta_3 vs tiempo")
86 xlabel("t (s)")
87 ylabel("\theta_3 (\circ)")
88 grid on
89 xlim([0 ExTiempo(end)])
90 ylim([125 200])
91 yticks(125:25:200)
92
93
94 figure()
95 subplot(3,1,1)
96 plot(tiempo,Tabla_Cinematica.V_1,"r-.")
97 hold on
98 plot(ExTiempo,TablaExcel.Vr1,"k--")
99 hold off
100 title("V_r vs tiempo")
101 xlabel("t (s)")
102 ylabel("V_r (m s^{-1})")
103 grid on
104 xlim([0 ExTiempo(end)])
105 ylim([-0.6 0.6])
106 yticks(-0.6:0.3:0.6)
107 legend("Newton Raphson","Simulacion","Location","east",'FontSize',7)
108
109 subplot(3,1,2)
110 plot(tiempo,Tabla_Cinematica.omega_1*180/pi,"r-.")
111 hold on
112 plot(ExTiempo,TablaExcel.w1,"k--")
```

```
hold off
title("\omega_1 vs tiempo")
xlabel("t (s)")
ylabel("\omega_1 (\circ s^{-1})")
grid on
xlim([0 ExTiempo(end)])
ylim([-80 80])
yticks(-80:40:80)


subplot(3,1,3)
plot(tiempo,Tabla_Cinematica.omega_3*180/pi,"r-.")
hold on
plot(ExTiempo,TablaExcel.W3,"k--")
hold off
title("\omega_3 vs tiempo")
xlabel("t (s)")
ylabel("\omega_3 (\circ s^{-1})")
grid on
xlim([0 ExTiempo(end)])
yticks([-200:100:200])

figure()
subplot(3,1,1)
plot(tiempo,Tabla_Cinematica.a_1,"r-.")
hold on
plot(ExTiempo,TablaExcel.Ar1,"k--")
hold off
title("A_r vs tiempo")
xlabel("t (s)")
ylabel("A_r (m s^{-2})")
grid on
xlim([0 ExTiempo(end)])
yticks(-5:2.5:5)
legend("Newton Raphson","Simulacion","Location","east",'FontSize',7)

subplot(3,1,2)
plot(tiempo,Tabla_Cinematica.alpha_1*180/pi,"r-.")
hold on
plot(ExTiempo,TablaExcel.A1,"k--")
hold off
title("\alpha_1 vs tiempo")
xlabel("t (s)")
ylabel("\alpha_1 (\circ s^{-2})")
grid on
xlim([0 ExTiempo(end)])
ylim([-400 800])
yticks([-400:400:800])

subplot(3,1,3)
plot(tiempo,Tabla_Cinematica.alpha_3*180/pi,"r-.")
hold on
plot(ExTiempo,TablaExcel.A3,"k--")
hold off
title("\alpha_3 vs tiempo")
xlabel("t (s)")
ylabel("\alpha_3 (\circ s^{-2})")
grid on
```

```
171 xlim([0 ExTiempo(end)])
172 ylim([-1500 1500])
173 yticks(-1500:750:1500)
```