Práctica 3

Dinámica de Robots

3.1.-Introducción

La dinámica del robot relaciona el movimiento del robot y las fuerzas implicadas en el mismo. El modelo dinámico establece relaciones matemáticas entre las coordenadas articulares (o las coordenadas del extremo del robot), sus derivadas (velocidad y aceleración), las fuerzas y pares aplicados en las articulaciones (o en el extremo) y los parámetros del robot (masas de los eslabones, inercias, etc).

Siguiendo con la filosofía de este libro, se recomienda al lector que quiera profundizar sobre la dinámica de robots, la lectura de los textos [1],[2],[3], donde se estudian varias formulaciones clásicas como Lagrange-Euler o las ecuaciones generalizadas de D'Alembert. Hay que tener en cuenta que las ecuaciones de movimiento obtenidas con estas formulaciones son equivalentes en el sentido que describen la conducta dinámica del robot, sin embargo, cada una de ellas presenta características diferentes que la hacen más apropiada para ciertas tareas. Por ejemplo, la formulación de Lagrange-Euler presenta un modelo simple y elegante, dando como resultado una serie de ecuaciones diferenciales no lineales de 2º orden acopladas útiles para el estudio de estrategias de control en el espacio de estados de las variables articulares del robot, pero que se presentan ineficaces para aplicaciones en tiempo real dado el elevado tiempo de computación que requieren las operaciones con matrices de transformación homogénea.

Los modelos dinámicos que se estudian en esta práctica están basados en el algoritmo recursivo de Newton-Euler (N-E) desarrollado por Luh [1]. Aunque las formulaciones recursivas destruyen la estructura del modelo dinámico analítico y dan lugar a la falta de ecuaciones cerradas necesarias para el análisis del control, la dificultad de un análisis clásico es enorme debido a que se obtienen expresiones fuertemente no-lineales que constan de cargas inerciales, fuerzas de acoplo entre las articulaciones y efectos de las cargas de gravedad, con la dificultad añadida de que los pares/fuerzas dinámicos dependen de los parámetros físicos del manipulador, de la configuración instantánea de las articulaciones, de la velocidad, de la aceleración y de la

carga que soporta el robot. Aunque las ecuaciones del movimiento son equivalentes ya sean analíticas o recursivas, los diferentes planteamientos dependen de los objetivos que se quieran conseguir con ellos. En algunos casos es necesario solucionar el problema dinámico de un robot para lograr tiempos de calculo rápidos en la evaluación de los pares y fuerzas articulares para controlar el manipulador, y en otros casos son necesarios planteamientos para facilitar el análisis y la síntesis del control.

3.2.-Dinámica inversa. La formulación de Newton-Euler

El método de Newton-Euler permite obtener un conjunto de ecuaciones recursivas hacia delante de velocidad y aceleración lineal y angular las cuales están referidas a cada sistema de referencia articular. Las velocidades y aceleraciones de cada elemento se propagan hacia adelante desde el sistema de referencia de la base hasta el efector final. Las ecuaciones recursivas hacia atrás calculan los pares y fuerzas necesarios para cada articulación desde la mano (incluyendo en ella efectos de fuerzas externas), hasta el sistema de referencia de la base.

3.2.1. Sistemas de coordenadas en movimiento.

La formulación de N-E se basa en los sistemas de coordenadas en movimiento [1].

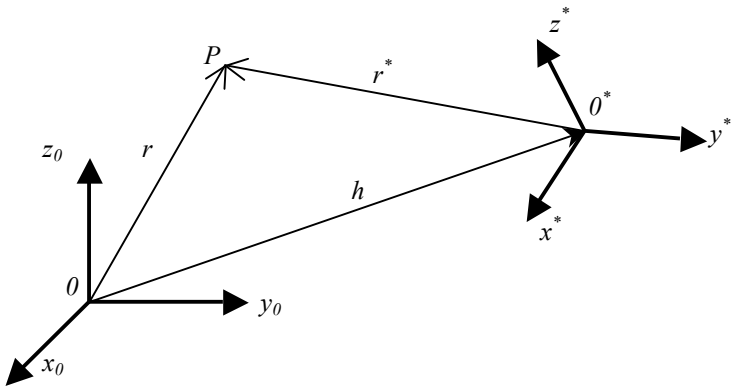


Figura 3.1. Sistemas de coordenadas en movimiento

Con respecto a la figura 3.1 se tiene que el sistema de coordenadas $\mathbf{0}^*$ se desplaza y gira en el espacio respecto del sistema de referencia de la base **0**, el vector que describe el origen del sistema en movimiento es ***h*** y el punto ***P*** se describe respecto del sistema $\mathbf{0}^*$ a través del vector $\mathbf{r}^*$, de acuerdo a esto, la descripción del punto ***P*** respecto del sistema de la base es:

$$r = r^* + h \tag{3.1}$$

$$\frac{dr}{dt} = \frac{dr^*}{dt} + \frac{dh}{dt} = v^* + v_h \tag{3.2}$$

donde v^* es la velocidad del punto ***P*** respecto del origen del sistema $\mathbf{0}^*$ en movimiento y v_h es la velocidad del origen del sistema $\mathbf{0}^*$ respecto de la base.

Si el punto ***P*** se desplaza y gira respecto del sistema $\mathbf{0}^*$ la ecuación (3.2) debe escribirse como:

$$v = \frac{dr^*}{dt} + \frac{dh}{dt} = \left(\frac{d^* r^*}{dt} + w \times r^* \right) + \frac{dh}{dt} \tag{3.3}$$

donde $\frac{d^* r^*}{dt}$ es la velocidad lineal del punto ***P*** respecto del origen $\mathbf{0}^*$ y $w \times r^*$ es la velocidad angular del punto ***P*** respecto del origen $\mathbf{0}^*$.[1]

De manera similar la aceleración general del sistema de puede describir como:

$$a = \frac{dv}{dt} = \frac{d^2 r^*}{dt^2} + \frac{d^2 h}{dt^2} = a^* + a_h \tag{3.4}$$

$$a = \frac{d^{*2} r^*}{dt^2} + 2w \times \frac{d^* r^*}{dt} + w \times (w \times r) + \frac{dw}{dt} \times r^* + \frac{d^2 h}{dt^2} \tag{3.5}$$

3.2.2. Cinemática de los eslabones del Robot.

A partir de las ecuaciones (3.1) a (3.5) de la sección anterior se desarrolla a continuación el planteamiento general para la cinemática de los eslabones del robot [1]

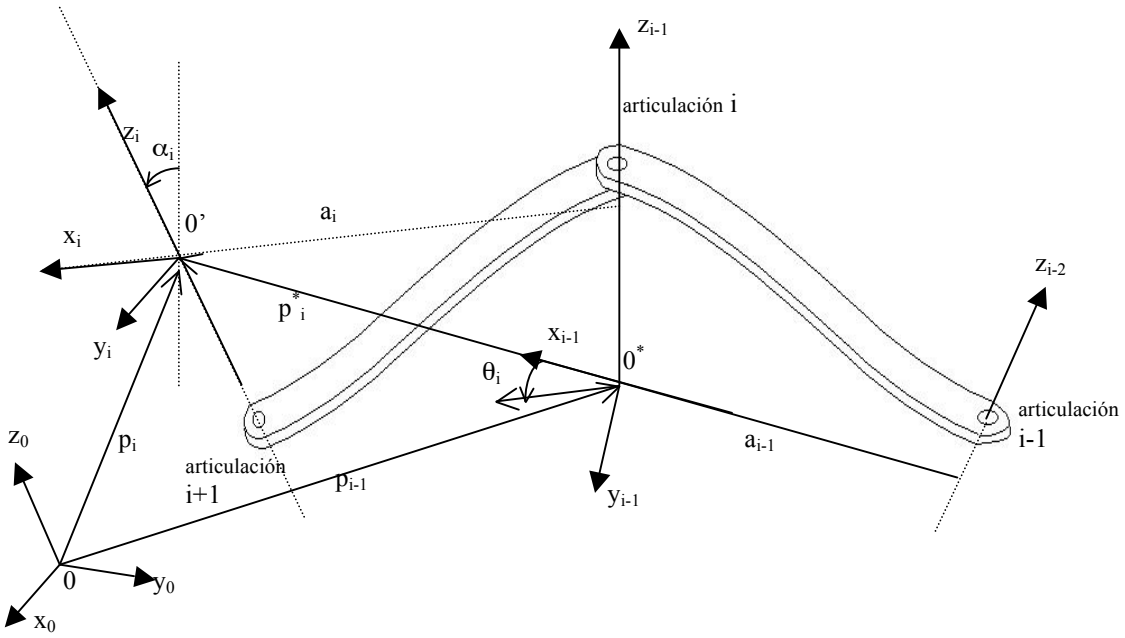


Figura 3.2. Relaciones vectoriales entre los sistemas de referencia $0,0^*$ y 0'

De acuerdo a la figura 3.2 las ecuaciones cinemáticas para los eslabones de un robot, se pueden escribir como:

$$v_i = \frac{d^* p_i^*}{dt} + w_{i-1} \times p_i^* + v_{i-1} \tag{3.6}$$

$$w_i = w_{i-1} + w_i^*$$

Debe notarse que la velocidad angular del sistema de referencia w_i es igual a la suma de la velocidad angular absoluta del sistema i-1 más la velocidad angular relativa w_i^* del eslabón referida a su propio sistema de coordenadas.

La aceleración lineal del sistema de coordenadas de la articulación i es:

$$\dot{v}_i = \frac{d^{*2} p_i^*}{dt^2} + \dot{w}_{i-1} \times p_i^* + 2w_{i-1} \times \frac{d^* p_i^*}{dt} + w_{i-1} \times \left(w_{i-1} \times p_i^*\right) + \dot{v}_{i-1} \tag{3.7}$$

$$\dot{w}_i = \dot{w}_{i-1} + \dot{w}_i^* \tag{3.8}$$

La aceleración angular del sistema de referencia i **(x_i, y_i, z_i)** respecto del sistema **(x_{i-1}, y_{i-1}, z_{i-1})** se consigue de manera similar a la ecuación (3.3)

$$\dot{w}_i^* = \frac{d^* w_i^*}{dt} + w_{i-1} \times w_i^* \tag{3.9}$$

por lo que la ecuación (3.8) queda como:

$$\dot{w}_i = \dot{w}_{i-1} + \frac{d^* w_i^*}{dt} + w_{i-1} \times w_i^* \tag{3.10}$$

En general para un robot los sistemas de coordenadas **(x_{i-1}, y_{i-1}, z_{i-1})** y **(x_i, y_i, z_i)** están unidos a los eslabones i-1 e i. La velocidad del eslabón i respecto del sistema de coordenadas i-1 es $\dot{q}_i$. Si el eslabón es prismático, la velocidad será una velocidad de traslación relativa respecto del sistema **(x_{i-1}, y_{i-1}, z_{i-1})** y si es rotacional le corresponderá una velocidad rotacional relativa del eslabón i respecto del sistema **(x_{i-1}, y_{i-1}, z_{i-1})**, por lo tanto:

$$w_i^* = \begin{cases} z_{i-1}\dot{q}_i & \text{si el eslabón i es rotacional} \\ 0 & \text{si el eslabón i es traslacional} \end{cases} \tag{3.11}$$

donde $\dot{q}_i$ es la magnitud de la velocidad angular del eslabón i con respecto al sistema de coordenadas **(x_{i-1}, y_{i-1}, z_{i-1})**. De manera similar:

$$\frac{d^* w^*}{dt} = \begin{cases} z_{i-1}\ddot{q}_i & \text{si el eslabón i es rotacional} \\ 0 & \text{si el eslabón i es traslacional} \end{cases} \tag{3.12}$$

Debe notarse que el vector z_{i-1} es igual a $(0, 0, 1)^T$.

Las velocidades y aceleraciones de los sistemas de coordenadas ligados a cada eslabón son absolutas y se calculan como:

$$w_i = \begin{cases} w_{i-1} + z_{i-1}\dot{q}_i & \text{si el eslabón i es rotacional} \\ w_{i-1} & \text{si el eslabón i es traslacional} \end{cases} \tag{3.13}$$

$$\dot{w}_i = \begin{cases} \dot{w}_{i-1} + z_{i-1}\ddot{q}_i + w_{i-1} \times \left(z_{i-1}\dot{q}_i\right) & \text{si el eslabón i es rotacional} \\ \dot{w}_{i-1} & \text{si el eslabón i es traslacional} \end{cases} \tag{3.14}$$

Las velocidades lineales de los sistemas de referencia de cada eslabón se calculan como:

$$\frac{d^* p_i}{dt} = \begin{cases} w_i \times p_i^* & \text{si el eslabón i es rotacional} \\ z_{i-1}\dot{q}_i & \text{si el eslabón i es traslacional} \end{cases} \tag{3.15}$$

$$\frac{d^{*2} p_i^*}{dt^2} = \begin{cases} \frac{d^* w_i^*}{dt} \times p_i^* + w_i^* \times \left(w_i^* \times p_i^*\right) & \text{si el eslabón i es rotacional} \\ z_{i-1}\ddot{q}_i & \text{si el eslabón i es traslacional} \end{cases} \tag{3.16}$$

por lo que la velocidad lineal absoluta del sistema de coordenadas ligado a cada eslabón se calcula como:

$$v_i = \begin{cases} w_i \times p_i^* + v_{i-1} & \text{si el eslabón i es rotacional} \\ z_{i-1}\dot{q}_i + w_i \times p_i^* + v_{i-1} & \text{si el eslabón i es traslacional} \end{cases} \tag{3.17}$$

La aceleración se calcula como:

$$\dot{v}_i = \begin{cases} \dot{w}_i \times p_i^* + w_i \times \left(w_i \times p_i^*\right) + \dot{v}_{i-1} & \text{si el eslabón i es rotacional} \\ z_{i-1}\ddot{q}_i + \dot{w}_i \times p_i^* + 2w_i \times \left(z_{i-1}\dot{q}_i\right) + w_i \times \left(w_i \times p_i^*\right) + \dot{v}_{i-1} & \text{si el eslabón i es traslacional} \end{cases} \tag{3.18}$$

3.2.3. Ecuaciones de movimiento recursivas.

A partir de las ecuaciones cinemáticas del apartado anterior y aplicando el principio de D'Alembert del equilibrio estático para todos los instantes de tiempo, se obtienen las ecuaciones recursivas de Newton-Euler.[1]

Si se utiliza la nomenclatura de la figura 3.2 sobre un eslabón cualquiera del robot, tal y como se muestra en la figura 3.3

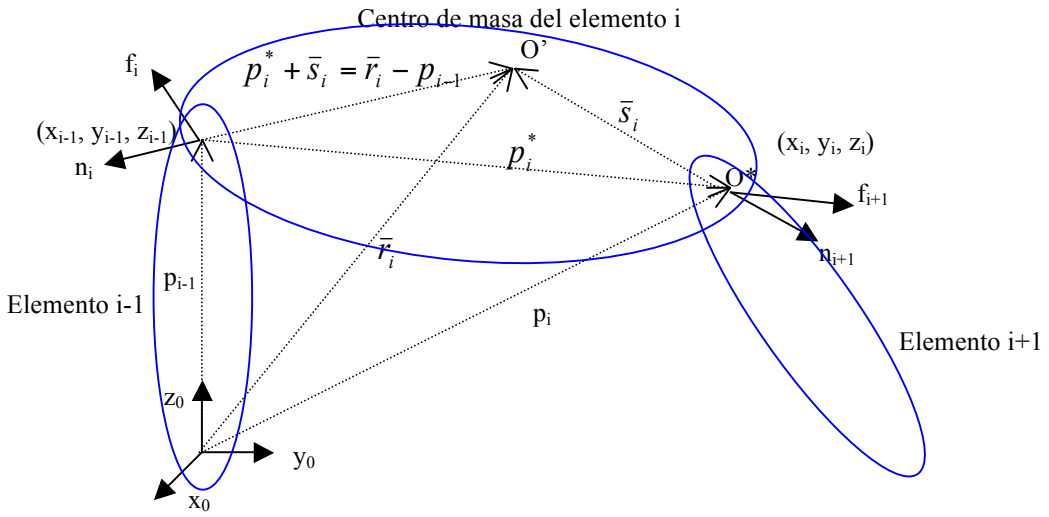


Figura 3.3.Fuerzas y momentos sobre el elemento i

Donde:

m_i	masa total del eslabón i,
$\bar{r}_i$	posición del centro de masa del elemento i desde el origen del sistema de referencia de la base,
$\bar{s}_i$	posición del centro de masa del elemento i desde el origen del sistema de coordenadas **(x_i, y_i, z_i),**
p_i^*	posición del origen de coordenadas i-ésimo con respecto al sistema de coordenadas (i-1)-ésimo,
$\bar{v}_i = \frac{d\bar{r}_i}{dt}$	velocidad lineal del centro de masa del elemento i,
$\bar{a}_i = \frac{d\bar{v}_i}{dt}$	aceleración lineal del centro de masa del elemento i,
F_i	fuerza total externa ejercida sobre el elemento i en el centro de masa,
N_i	momento total externo ejercido sobre el elemento i en el centro de masa,
I_i	matriz de inercia del elemento i respecto de su centro de masa con respecto al sistema de coordenadas **(x_0, y_0, z_0),**

f_i fuerza ejercida sobre el elemento i por el elemento (i-1) en el sistema de coordenadas **(x_{i-1}, y_{i-1}, z_{i-1})** para soportar al elemento i y a los elementos por encima de él,

n_i momento ejercido sobre el elemento i por el elemento (i-1) en el sistema de coordenadas **(x_{i-1}, y_{i-1}, z_{i-1}),**

⇒ NOTA: Es importante que se identifiquen estas variables sobre el dibujo del robot, para poder seguir los siguientes desarrollos.

Si se omiten los efectos del rozamiento viscoso en las articulaciones, y se aplica el principio de D'Alembert, se obtiene para cada eslabón:

$$F_i = \frac{d(m_i \bar{v}_i)}{dt} = m_i \bar{a}_i \tag{3.18}$$

$$N_i = \frac{d(I_i w_i)}{dt} = I_i \dot{w}_i + w_i \times (I_i w_i) \tag{3.19}$$

realizando el balance de pares y fuerzas en la figura 3.3:

$$F_i = f_i - f_{i+1} \tag{3.20}$$

$$N_i = n_i - n_{i+1} + (p_{i-1} - \bar{r}_i) \times f_i - (p_i - \bar{r}_i) \times f_{i+1} \tag{3.21}$$

$$= n_i - n_{i+1} + (p_{i-1} - \bar{r}_i) \times F_i - p_i^* \times f_{i+1} \tag{3.22}$$

que utilizando la relación geométrica:

$$\bar{r}_i - p_{i-1} = p_i^* + \bar{s}_i \tag{3.23}$$

se obtienen las ecuaciones recursivas:

$$f_i = F_i + f_{i+1} = m_i \bar{a}_i + f_{i+1} \tag{3.24}$$

$$n_i = n_{i+1} + p_i^* \times f_{i+1} + (p_i^* + \bar{s}_i) \times F_i + N_i \tag{3.25}$$

Se observa que estas ecuaciones son recursivas y permiten obtener las fuerzas y momentos en los elementos $i = 1,2,...,n$ para un robot de n elementos. f_{i+1} y n_{i+1} representan la fuerza y momento ejercidos por la mano del robot sobre un objeto externo.

Por lo tanto, el par/fuerza para cada articulación se expresa como:

$$\tau_i = \begin{cases} n_i^T z_{i-1} + b_i \dot{q}_i & \text{si el eslabón i es rotacional} \\ f_i^T z_{i-1} + b_i \dot{q}_i & \text{si el eslabón i es traslacional} \end{cases} \tag{3.26}$$

donde b_i es el coeficiente de rozamiento viscoso de la articulación.

3.2.4. Algoritmo computacional.

En resumen de los dos apartados anteriores, las ecuaciones de N-E consisten en un conjunto de ecuaciones recursivas [(3.13),(3.14), (3.17), (3.18)] hacia delante y [(3.18),(3.19), (3.20), (3.21),(3.26)] hacia atrás.

Para obtener un algoritmo computacional, se debe tener en cuenta que en las anteriores ecuaciones, las matrices de inercia I_i y los parámetros del robot, $\bar{r}_i$, $\bar{s}_i$, p_i^* están referenciados respecto del sistema de coordenadas de la base.

Luh y col. [1980] utilizaron las matrices de rotación 3x3 ${}^{i-1}R_i$, que ya han sido estudiadas en la práctica 2 pues corresponden a la submatriz superior izquierda de las matrices de transformación homogénea ${}^{i-1}A_i$, para expresar las variables $w_i, \dot{w}_i, v_i, \dot{v}_i, a_i, p_i^*, \bar{s}_i, F_i, N_i, f_i, n_i$ y τ_i que se refieren al sistema de coordenadas de la base como ${}^iR_0 w_i, {}^iR_0 \dot{w}_i, {}^iR_0 v_i, {}^iR_0 \dot{v}_i, {}^iR_0 a_i, {}^iR_0 p_i^*, {}^iR_0 \bar{s}_i, {}^iR_0 F_i, {}^iR_0 N_i, {}^iR_0 f_i, {}^iR_0 n_i$ y ${}^iR_0 \tau_i$, que están referenciados al propio sistema de coordenadas del elemento i.

De esta manera, las ecuaciones recursivas de N-E quedan expresadas en la siguiente tabla:

Ecuaciones hacia delante: i = 1,2,....,n

$$ {}^{i}R_0 w_i = \begin{cases} {}^{i}R_{i-1}({}^{i-1}R_0 w_{i-1} + z_0 \dot{q}_i) & \text{si el eslabón i es rotacional} \\ {}^{i}R_{i-1}({}^{i-1}R_0 w_{i-1}) & \text{si el eslabón i es traslacional} \end{cases} \quad (3.27) $$

$$ {}^{i}R_0 \dot{w}_i = \begin{cases} {}^{i}R_{i-1}[{}^{i-1}R_0 \dot{w}_{i-1} + z_0 \ddot{q}_i + ({}^{i-1}R_0 w_{i-1}) \times (z_0 \dot{q}_i)\ ] & \text{si el eslabón i es rotacional} \\ {}^{i}R_{i-1}({}^{i-1}R_0 \dot{w}_{i-1}) & \text{si el eslabón i es traslacional} \end{cases} \quad (3.28) $$

$$ {}^{i}R_0 \dot{v}_i = \begin{cases} ({}^{i}R_0 \dot{w}_i) \times ({}^{i}R_0 p_i^*) + ({}^{i}R_0 \dot{w}_i) \times \left[({}^{i}R_0 \dot{w}_i) \times ({}^{i}R_0 p_i^*) \right] + \\ + {}^{i}R_{i-1}\left({}^{i-1}R_0 \dot{v}_{i-1} \right) & \text{si el eslabón i es rotacional} \\ {}^{i}R_{i-1}(z_0 \ddot{q}_i + {}^{i-1}R_0 \dot{v}_{i-1}) + ({}^{i}R_0 \dot{w}_i) \times ({}^{i}R_0 p_i^*) + \\ + 2({}^{i}R_0 w_i) \times ({}^{i}R_{i-1} z_0 \dot{q}_i) + \\ + ({}^{i}R_0 w_i) \times \left[({}^{i}R_0 \dot{w}_i) \times ({}^{i}R_0 p_i^*) \right] & \text{si el eslabón i es traslacional} \end{cases} \quad (3.29) $$

$$ {}^{i}R_0 \bar{a}_i = \left({}^{i}R_0 \dot{w}_i \right) \times \left({}^{i}R_0 \bar{s}_i \right) + \left({}^{i}R_0 w_i \right) \times \left[\left({}^{i}R_0 w_i \right) \times \left({}^{i}R_0 \bar{s}_i \right) \right] + {}^{i}R_0 \dot{v}_i \quad (3.30) $$

Ecuaciones hacia atrás: i =n,n-1,....,1

$$ {}^{i}R_0 f_i = {}^{i}R_{i+1}({}^{i+1}R_0 f_{i+1}) + m_i \, {}^{i}R_0 \bar{a}_i \quad (3.31) $$

$$ \begin{aligned} {}^{i}R_0 n_i = {}^{i}R_{i+1}[{}^{i+1}R_0 n_{i+1} + ({}^{i+1}R_0 p_i^*) \times ({}^{i+1}R_0 f_{i+1})\ ] + ({}^{i}R_0 p_i^* + {}^{i}R_0 \bar{s}_i) \times ({}^{i}R_0 F_i) + \\ + \left({}^{i}R_0 I_i \, {}^{0}R_i \right)\left({}^{i}R_0 \dot{w}_i \right) + \left({}^{i}R_0 w_i \right) \times \left[\left({}^{i}R_0 I_i \, {}^{0}R_i \right)\left({}^{i}R_0 w_i \right) \right] \end{aligned} \quad (3.32) $$

$$ {}^{i}R_0 \tau_i = \begin{cases} ({}^{i}R_0 n_i)^T ({}^{i}R_{i-1} z_0) + b_i \dot{q}_i & \text{si el eslabón i es rotacional} \\ ({}^{i}R_0 f_i)^T ({}^{i}R_{i-1} z_0) + b_i \dot{q}_i & \text{si el eslabón i es traslacional} \end{cases} \quad (3.33) $$

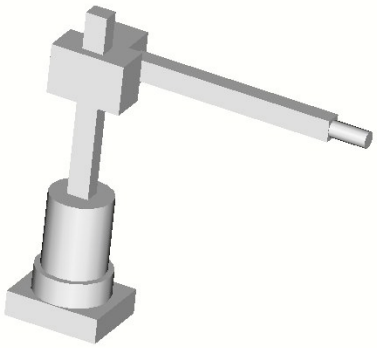
Tabla 3.1. Algoritmo recursivo de N-E

⇒ NOTA: Para la utilización de este algoritmo en la solución del problema dinámico inverso de un manipulador de cadena abierta se tienen que tener en cuenta las siguientes consideraciones:

- n = número de elementos de la cadena.
- $w_0 = \dot{w}_0 = v_0 = 0$, (la base del robot se considera en reposo)
- $\dot{v}_0 = g = (g_x, g_y, g_z)^T$ con $|g| = 9.8\,\text{m/s}^2$
- $z_0 = (0,0,1)^T$ lo que implica $\dot{v}_0 = g = (0,0,g)^T$ con $g = 9.8$

Ejemplo 3.1

A continuación se presenta la implementación del algoritmo de N-E para el robot prismático de 4 gdl.



Cada una de las ecuaciones (3.27) a (3.33) han sido implementadas por los autores como una función de ***MatLab®***, y se proporcionarán para la realización de la práctica.

La notación que se ha utilizado en las funciones necesarias para poder ejecutar el algoritmo se corresponde con la siguiente tabla:

Dh	Calcula la matriz de rotación ${}^{i-1}R_i$
Ri0pi	${}^{i}R_0 p_i^*$
Ri0si	${}^{i}R_0 \bar{s}_i$
Ri0wi	${}^{i}R_0 w_i$
Ri0wpi	${}^{i}R_0 \dot{w}_i$
Ri0vpi	${}^{i}R_0 \dot{v}_i$
Ri0ai	${}^{i}R_0 a_i$
Ri0fi	${}^{i}R_0 F_i$
Ri0ni	${}^{i}R_0 N_i$
Ri0fia	${}^{i}R_0 f_i$
Ri0nia	${}^{i}R_0 n_i$
T_r	${}^{i}R_0 \tau_i$ para una articulación rotacional
T_p	${}^{i}R_0 \tau_i$ para una articulación prismática

⇒ NOTA: Se recomienda que desde el entorno ***MatLab®***, se consulte la ayuda de todas las funciones anteriores, fijándose en los parámetros que se le pasan a la función en cada caso.

```
» help ri0pi

  RI0PI           Vector ri0pi.
      Y = RI0PI(A, D, ALFA) calcula el vector 3x1 que representa la
      localización de (xi,yi,zi) desde el origen de (xi-1,yi-1,zi-1)
      con respecto al sistema de coordenadas i-ésimo utilizando los
      parámetros de Denavit-Hartenberg A, D y ALFA.

»
```

Código en MatLab®. A continuación se presenta la función NEWTONEULER4, utilizada para resolver la dinámica inversa del robot prismático de 4 gdl. Realizando help NEWTONEULER4, la función nos informa sobre los parámetros necesarios para realizar los cálculos.

```
 » help newtoneuler4

NEWTONEULER4  Dinámica inversa utilizando el método de Newton-Euler.
       TAU = NEWTONEULER4(Q, QP, QPP, G, M5, IEXTER) calcula el vector
       4x1 de pares/fuerzas de entrada a las articulaciones. Q el
       vector 4x1 de coordenadas articulares. QP es el vector 4x1 que
       representa la velocidad de cada articulación. QPP es el vector
       4x1 que indicala aceleración de cada articulación. G es el valor
       de la gravedad (m/s^2).
       M5 es la masa de la carga externa(Kg) que transporta el brazo
       robot.
       IEXTER es la matriz 3x3 de inercia de la carga exterior(Kg-m^2).

       See also DH, RI0PI, RI0SI, RI0WI, RI0WIP, RI0VPI_R, RI0AI,
       RI0FI, RI0NI,RI0FIA, RI0NIA, T_R, F_P.

 »
```

Es importante que el lector repare en los parámetros que se pasan en cada caso a la función correspondiente.

En este ejemplo, para contabilizar los efectos de la carga exterior, se ha introducido como un eslabón nº 5. Para ello se ha necesitado definir una articulación entre el eslabón 4 y el 5, que puede definirse utilizando las librerías de enlace rotacional o prismático, dado que como condiciones de este eslabón imponen $q_5 = \dot{q}_5 = 0$, de manera que las ecuaciones (3.27),(3.28) y (3.29) coinciden para ambas librerías.

```
%  NEWTONEULER4  Dinámica inversa utilizando el método de Newton-
Euler.
%      TAU = NEWTONEULER4(Q, QP, QPP, G, M5, IEXTER) calcula el vector
%      4x1 de pares/fuerzas de entrada a las articulaciones. Q el vector
%      4x1 de coordenadas articulares. QP es el vector 4x1 que representa
%      la velocidad de cada articulación. QPP es el vector 4x1 que indica
%      la aceleración de cada articulación. G es el valor de la gravedad
(m/s^2).
%      M5 es la masa de la carga externa(Kg) que transporta el brazo robot.
%      IEXTER es la matriz 3x3 de inercia de la carga exterior(Kg-m^2).
%
%      See also DH, RI0PI, RI0SI, RI0WI, RI0WIP, RI0VPI_R, RI0AI, RI0FI, RI0NI,
%      RI0FIA, RI0NIA, T_R, F_P.

function tau = newtoneuler4(q,qp,qpp,g,m5,Iexter);

% ---------------------------------------------------------------
%  Parámetros Denavit-Hartenberg del robot
% ---------------------------------------------------------------
teta = [q(1)   0       0      q(4)];
d    = [0.4    q(2)    q(3)   0.2 ];
a    = [0     -0.1     0      0   ];
alfa = [0     -pi/2    0      0   ];

% ---------------------------------------------------------------
%   Factores de posicionamiento de los centros de gravedad
% ---------------------------------------------------------------
factor1 = -0.5; factor2 = -0.5; factor3 = -0.5; factor4 = -0.5;

% ---------------------------------------------------------------
%           Masa de cada elemento (Kg)
% ---------------------------------------------------------------
m1 = 10;  m2 = 5;  m3 = 5;  m4 = 3;

% ---------------------------------------------------------------
%   Coeficiente de rozamiento viscoso de cada articulacion
% ---------------------------------------------------------------
b1 = 0.05;  b2 = 0.05;  b3 = 0.05;  b4 = 0.05;

% ---------------------------------------------------------------
%   Matrices de Inercia (Kg-m^2)
% ---------------------------------------------------------------
r10I_r01 = [0.0191 0 0;0 0.0191 0;0 0 0.0068];
r20I_r02 = [0.0031 0 0;0 0.0484 0;0 0 0.0484];
r30I_r03 = [0.0606 0 0;0 0.0053 0;0 0 0.0606];
r40I_r04 = [0.0022 0 0;0 0.0022 0;0 0 0.0014];

% ---------------------------------------------------------------
%   Vectores ri0pi, ri0si.
% ---------------------------------------------------------------
r10p1 = ri0pi(a(1), d(1), alfa(1));
r20p2 = ri0pi(a(2), d(2), alfa(2));
r30p3 = ri0pi(a(3), d(3), alfa(3));
r40p4 = ri0pi(a(4), d(4), alfa(4));
r50p5 = zeros(3,1);
r10s1 = ri0si(a(1), d(1), alfa(1), factor1);
r20s2 = ri0si(a(2), d(2), alfa(2), factor2);
r30s3 = ri0si(a(3), d(3), alfa(3), factor3);
r40s4 = ri0si(a(4), d(4), alfa(4), factor4);
r50s5 = zeros(3,1);
```

```
% ------------------------------------------------------------
%     Matrices de transformacion
% ------------------------------------------------------------
r01 = dh(teta(1), alfa(1));          r10 = r01';
r12 = dh(teta(2), alfa(2));          r21 = r12';
r23 = dh(teta(3), alfa(3));          r32 = r23';
r34 = dh(teta(4), alfa(4));          r43 = r34';
r45 = eye(3);                        r54 = r45';

% ------------------------------------------------------------
%        Velocidad angular de las articulaciones
% ------------------------------------------------------------
r00w0 = zeros(3,1);
r10w1 = ri0wi(r10, r00w0, qp(1));
r20w2 = r21*r10w1;
r30w3 = r32*r20w2;
r40w4 = ri0wi(r43, r30w3, qp(4));
r50w5 = ri0wi(r54, r40w4, 0);

% ------------------------------------------------------------
%     Aceleracion angular de las articulaciones
% ------------------------------------------------------------
r00wp0 = zeros(3,1);
r10wp1 = ri0wpi(r10, r00wp0, r00w0, qp(1), qpp(1));
r20wp2 = r21*r10wp1;
r30wp3 = r32*r20wp2;
r40wp4 = ri0wpi(r43, r30wp3, r30w3, qp(4), qpp(4));
r50wp5 = ri0wpi(r54, r40wp4, r40w4, 0, 0);

% ------------------------------------------------------------
%     Aceleracion lineal articular
% ------------------------------------------------------------
r00vp0 = [0; 0; g];
r10vp1 = ri0vpi_r(r10, r00vp0, r10wp1, r10w1, r10p1);
r20vp2 = ri0vpi_p(r21, r10vp1, r20wp2, r20w2, r20p2, qp(2), qpp(2));
r30vp3 = ri0vpi_p(r32, r20vp2, r30wp3, r30w3, r30p3, qp(3), qpp(3));
r40vp4 = ri0vpi_r(r43, r30vp3, r40wp4, r40w4, r40p4);
r50vp5 = ri0vpi_r(r54, r40vp4, r50wp5, r50w5, r50p5);

% ------------------------------------------------------------
%  Aceleracion del centro de masa de cada elemento
% ------------------------------------------------------------
r10a1 = ri0ai(r10vp1, r10wp1, r10w1, r10s1);
r20a2 = ri0ai(r20vp2, r20wp2, r20w2, r20s2);
r30a3 = ri0ai(r30vp3, r30wp3, r30w3, r30s3);
r40a4 = ri0ai(r40vp4, r40wp4, r40w4, r40s4);
r50a5 = ri0ai(r50vp5, r50wp5, r50w5, r50s5);

% ------------------------------------------------------------
%     Fuerza en el centro de masa de cada elemento
% ------------------------------------------------------------
r50f5 = ri0fi(r50a5, m5);
r40f4 = ri0fi(r40a4, m4);
r30f3 = ri0fi(r30a3, m3);
r20f2 = ri0fi(r20a2, m2);
r10f1 = ri0fi(r10a1, m1);

% ------------------------------------------------------------
%     Par en el centro de masa de cada elemento
% ------------------------------------------------------------
r50n5 = ri0ni(r50wp5, r50w5, Iexter);
```

```
r40n4 = ri0ni(r40wp4, r40w4, r40I_r04);
r30n3 = ri0ni(r30wp3, r30w3, r30I_r03);
r20n2 = ri0ni(r20wp2, r20w2, r20I_r02);
r10n1 = ri0ni(r10wp1, r10w1, r10I_r01);

% ------------------------------------------------------------
%         Fuerzas articulares
% ------------------------------------------------------------
r50f5a = r50f5;
r40f4a = ri0fia(r45, r50f5a, r40f4);
r30f3a = ri0fia(r34, r40f4a, r30f3);
r20f2a = ri0fia(r23, r30f3a, r20f2);
r10f1a = ri0fia(r12, r20f2a, r10f1);

% ------------------------------------------------------------
%        Pares articulares
% ------------------------------------------------------------
r20p1 = r21*(r10p1);    r30p2 = r32*(r20p2);
r40p3 = r43*(r30p3);    r50p4 = r54*(r40p4);

r50n5a = r50n5;
r40n4a = ri0nia(r45, r50n5a, r50f5a, r40n4, r40f4, r50p4, r40p4,
r40s4);
r30n3a = ri0nia(r34, r40n4a, r40f4a, r30n3, r30f3, r40p3, r30p3,
r30s3);
r20n2a = ri0nia(r23, r30n3a, r30f3a, r20n2, r20f2, r30p2, r20p2,
r20s2);
r10n1a = ri0nia(r12, r20n2a, r20f2a, r10n1, r10f1, r20p1, r10p1,
r10s1);

% ------------------------------------------------------------
%        Fuerzas  y pares de accionamientos
% ------------------------------------------------------------
t_1 = t_r(r10, r10n1a, qp(1), b1);
t_2 = f_p(r21, r20f2a, qp(2), b2);
t_3 = f_p(r32, r30f3a, qp(3), b3);
t_4 = t_r(r43, r40n4a, qp(4), b4);

tau = [t_1; t_2; t_3; t_4];
```

⇒ NOTA: Debe notarse que los parámetros físicos y geométricos del robot se han introducido en el código. Se recuerda que los parámetros de D-H son característicos de cada robot. El factor de posición del centro de masas de cada eslabón, su masa y su inercia son datos del robot, así como los coeficientes de rozamiento viscoso aplicables en cada articulación.

Se recomienda que compruebe la función para velocidades y aceleraciones nulas, observando los resultados que se obtienen. En el ejemplo siguiente, se introducen velocidades y aceleraciones nulas en todas las articulaciones. Se puede comprobar como la fuerza en la articulación 2 (156 N) corresponde con el peso de los eslabones 2,3,4 y la masa exterior que debe ser soportado por la articulación.

```
» q=[0 0.5 0.4 0];
» qp=[0 0 0 0];
» qpp=qp;
» m4=3;
» iext=0.05*eye(3);
» tau=newtoneuler4(q,qp,qpp,9.8,m4,iext)

tau =

    0.0000
  156.8000
    0.0000
         0

»
```

Ejemplo 3.2

Se muestra a continuación la implementación del algoritmo de N-E para el robot rotacional de 6 gdl.

Se observa que a diferencia del ejemplo anterior, en este caso solamente se utilizan las librerías de las funciones de enlaces rotacionales.

Código en MatLab. A continuación se presenta la función NEWTONEULER6, utilizada para resolver la dinámica inversa del robot de 6 gdl. Realizando help NEWTONEULER6, la función nos informa sobre los parámetros necesarios para realizar los cálculos.

```
%   NEWTONEULER6  Dinámica inversa utilizando el método de Newton-Euler.
%       TAU = NEWTONEULER6(Q, QP, QPP, G, M7, IEXTER) calcula el vector
%       6x1 de pares/fuerzas de entrada a las articulaciones. Q el vector
%       6x1 de coordenadas articulares. QP es el vector 6x1 que representa
%       la velocidad de cada articulación. QPP es el vector 6x1 que indica
%       la aceleración de cada articulación. G es el valor de la gravedad (m/s^2).
%       M7 es la masa de la carga externa(Kg) que transporta el brazo robot.
%       IEXTER es la matriz 3x3 de inercia de la carga exterior(Kg-m^2).
%
%       See also DH, RI0PI, RI0SI, RI0WI, RI0WIP, RI0VPI_R, RI0AI, RI0FI, RI0NI,
%       RI0FIA, RI0NIA, T_R.



function tau = newtoneuler6(q,qp,qpp,g,m7,Iexter);

% ------------------------------------------------------------
%  Parámetros Denavit-Hartenberg del robot
% ------------------------------------------------------------
```

```
teta = q;
d    = [0.315  0      0     0.5     0    0.08];
a    = [0      0.45   0     0       0    0   ];
alfa = [-pi/2  0    pi/2  -pi/2   pi/2   0   ];

% ------------------------------------------------------------
%   Factores de posicionamiento de los centros de gravedad
% ------------------------------------------------------------
factor1 = -0.5; factor2 = -0.5; factor3 = -0.5;
factor4 = -0.5; factor5 = -0.5; factor6 = -0.5;

% ------------------------------------------------------------
%           Masa de cada elemento (Kg)
% ------------------------------------------------------------
m1 = 2.78;  m2 = 10.25;  m3 = 0;
m4 = 5.57;  m5 = 0;  m6 = 1.98;

% ------------------------------------------------------------
%   Coeficiente de rozamiento viscoso de cada articulacion
% ------------------------------------------------------------
b1 = 0.05;  b2 = 0.05;  b3 = 0.05;
b4 = 0.05;  b5 = 0.05;  b6= 0.05;

% ------------------------------------------------------------
%   Matrices de Inercia (Kg-m^2)
% ------------------------------------------------------------
r10I_r01 = [0.0191 0 0;0 0.0191 0;0 0 0.0068];
r20I_r02 = [0.0031 0 0;0 0.0484 0;0 0 0.0484];
r30I_r03 = zeros(3,3);
r40I_r04 = [0.0606 0 0;0 0.0053 0;0 0 0.0606];
r50I_r05 = zeros(3,3);
r60I_r06 = [0.0022 0 0;0 0.0022 0;0 0 0.0014];
% ------------------------------------------------------------
%   Vectores ri0pi, ri0si.
% ------------------------------------------------------------
r10p1 = ri0pi(a(1), d(1), alfa(1));
r20p2 = ri0pi(a(2), d(2), alfa(2));
r30p3 = ri0pi(a(3), d(3), alfa(3));
r40p4 = ri0pi(a(4), d(4), alfa(4));
r50p5 = ri0pi(a(5), d(5), alfa(5));
r60p6 = ri0pi(a(6), d(6), alfa(6));
r70p7 = zeros(3,1);

r10s1 = ri0si(a(1), d(1), alfa(1), factor1);
r20s2 = ri0si(a(2), d(2), alfa(2), factor2);
r30s3 = ri0si(a(3), d(3), alfa(3), factor3);
r40s4 = ri0si(a(4), d(4), alfa(4), factor4);
r50s5 = ri0si(a(5), d(5), alfa(5), factor5);
r60s6 = ri0si(a(6), d(6), alfa(6), factor6);
r70s7 = zeros(3,1);

% ------------------------------------------------------------
%     Matrices de transformacion
% ------------------------------------------------------------
r01 = dh(teta(1), alfa(1));          r10 = r01';
r12 = dh(teta(2), alfa(2));          r21 = r12';
r23 = dh(teta(3), alfa(3));          r32 = r23';
r34 = dh(teta(4), alfa(4));          r43 = r34';
r45 = dh(teta(5), alfa(5));          r54 = r45';
r56 = dh(teta(6), alfa(6));          r65 = r56';
r67 = eye(3);                        r76 = r67';
```

```
% ------------------------------------------------------------
%        Velocidad angular de las articulaciones
% ------------------------------------------------------------
r00w0 = zeros(3,1);
r10w1 = ri0wi(r10, r00w0, qp(1));
r20w2 = ri0wi(r21, r10w1, qp(2));
r30w3 = ri0wi(r32, r20w2, qp(3));
r40w4 = ri0wi(r43, r30w3, qp(4));
r50w5 = ri0wi(r54, r40w4, qp(5));
r60w6 = ri0wi(r65, r50w5, qp(6));
r70w7 = ri0wi(r76, r60w6, 0);

% ------------------------------------------------------------
%     Aceleracion angular de las articulaciones
% ------------------------------------------------------------
r00wp0 = zeros(3,1);
r10wp1 = ri0wpi(r10, r00wp0, r00w0, qp(1), qpp(1));
r20wp2 = ri0wpi(r21, r10wp1, r10w1, qp(2), qpp(2));
r30wp3 = ri0wpi(r32, r20wp2, r20w2, qp(3), qpp(3));
r40wp4 = ri0wpi(r43, r30wp3, r30w3, qp(4), qpp(4));
r50wp5 = ri0wpi(r54, r40wp4, r40w4, qp(5), qpp(5));
r60wp6 = ri0wpi(r65, r50wp5, r50w5, qp(6), qpp(6));
r70wp7 = ri0wpi(r76, r60wp6, r60w6, 0, 0);

% ------------------------------------------------------------
%     Aceleracion lineal articular
% ------------------------------------------------------------
r00vp0 = [0; 0; g];
r10vp1 = ri0vpi_r(r10, r00vp0, r10wp1, r10w1, r10p1);
r20vp2 = ri0vpi_r(r21, r10vp1, r20wp2, r20w2, r20p2);
r30vp3 = ri0vpi_r(r32, r20vp2, r30wp3, r30w3, r30p3);
r40vp4 = ri0vpi_r(r43, r30vp3, r40wp4, r40w4, r40p4);
r50vp5 = ri0vpi_r(r54, r40vp4, r50wp5, r50w5, r50p5);
r60vp6 = ri0vpi_r(r65, r50vp5, r60wp6, r60w6, r60p6);
r70vp7 = ri0vpi_r(r76, r60vp6, r70wp7, r70w7, r70p7);

% ------------------------------------------------------------
%  Aceleracion del centro de masa de cada elemento
% ------------------------------------------------------------
r10a1 = ri0ai(r10vp1, r10wp1, r10w1, r10s1);
r20a2 = ri0ai(r20vp2, r20wp2, r20w2, r20s2);
r30a3 = ri0ai(r30vp3, r30wp3, r30w3, r30s3);
r40a4 = ri0ai(r40vp4, r40wp4, r40w4, r40s4);
r50a5 = ri0ai(r50vp5, r50wp5, r50w5, r50s5);
r60a6 = ri0ai(r60vp6, r60wp6, r60w6, r60s6);
r70a7 = ri0ai(r70vp7, r70wp7, r70w7, r70s7);

% ------------------------------------------------------------
%     Fuerza en el centro de masa de cada elemento
% ------------------------------------------------------------
r70f7 = ri0fi(r70a7, m7);
r60f6 = ri0fi(r60a6, m6);
r50f5 = ri0fi(r50a5, m5);
r40f4 = ri0fi(r40a4, m4);
r30f3 = ri0fi(r30a3, m3);
r20f2 = ri0fi(r20a2, m2);
r10f1 = ri0fi(r10a1, m1);
% ------------------------------------------------------------
%     Par en el centro de masa de cada elemento
% ------------------------------------------------------------
```

```
r70n7 = ri0ni(r70wp7, r70w7, Iexter);
r60n6 = ri0ni(r60wp6, r60w6, r60I_r06);
r50n5 = ri0ni(r50wp5, r50w5, r50I_r05);
r40n4 = ri0ni(r40wp4, r40w4, r40I_r04);
r30n3 = ri0ni(r30wp3, r30w3, r30I_r03);
r20n2 = ri0ni(r20wp2, r20w2, r20I_r02);
r10n1 = ri0ni(r10wp1, r10w1, r10I_r01);

% ------------------------------------------------------------
%         Fuerzas articulares
% ------------------------------------------------------------
r70f7a = r70f7;
r60f6a = ri0fia(r67, r70f7a, r60f6);
r50f5a = ri0fia(r56, r60f6a, r50f5);
r40f4a = ri0fia(r45, r50f5a, r40f4);
r30f3a = ri0fia(r34, r40f4a, r30f3);
r20f2a = ri0fia(r23, r30f3a, r20f2);
r10f1a = ri0fia(r12, r20f2a, r10f1);

% ------------------------------------------------------------
%        Pares articulares
% ------------------------------------------------------------
r20p1 = r21*(r10p1);    r30p2 = r32*(r20p2);
r40p3 = r43*(r30p3);    r50p4 = r54*(r40p4);
r60p5 = r65*(r50p5);    r70p6 = r76*(r60p6);

r70n7a = r70n7;
r60n6a = ri0nia(r67, r70n7a, r70f7a, r60n6, r60f6, r70p6, r60p6,
r60s6);
r50n5a = ri0nia(r56, r60n6a, r60f6a, r50n5, r50f5, r60p5, r50p5,
r50s5);
r40n4a = ri0nia(r45, r50n5a, r50f5a, r40n4, r40f4, r50p4, r40p4,
r40s4);
r30n3a = ri0nia(r34, r40n4a, r40f4a, r30n3, r30f3, r40p3, r30p3,
r30s3);
r20n2a = ri0nia(r23, r30n3a, r30f3a, r20n2, r20f2, r30p2, r20p2,
r20s2);
r10n1a = ri0nia(r12, r20n2a, r20f2a, r10n1, r10f1, r20p1, r10p1,
r10s1);

% ------------------------------------------------------------
%        Fuerzas  y pares de accionamientos
% ------------------------------------------------------------
t_1 = t_r(r10, r10n1a, qp(1), b1);
t_2 = t_r(r21, r20n2a, qp(2), b2);
t_3 = t_r(r32, r30n3a, qp(3), b3);
t_4 = t_r(r43, r40n4a, qp(4), b4);
t_5 = t_r(r54, r50n5a, qp(5), b5);
t_6 = t_r(r65, r60n6a, qp(6), b6);

tau = [t_1; t_2; t_3; t_4; t_5; t_6];
```

⇒ NOTA: Debe notarse que los parámetros físicos y geométricos del robot se han introducido en el código. Se recuerda que los parámetros de D-H son característicos de cada robot. El factor de posición del centro de masas de cada eslabón, su masa y su inercia son datos del robot, así como los coeficientes de rozamiento viscoso aplicables en cada articulación.

3.3.-Dinámica directa. Método de Walker-Orin.

Las ecuaciones de movimiento planteadas en el apartado anterior permiten resolver el problema de la dinámica directa.

M.W.Walker y D.E.Orin [3] presentaron en 1982 cuatro métodos para la resolución del problema dinámico directo de una cadena cinemática abierta utilizando la formulación de N-E. En el artículo se realiza una comparación de la eficiencia computacional de los cuatro métodos presentados, concluyendo que el tercero de los presentados es el más eficiente frente al tiempo de cómputo. Para la realización de esta práctica los autores han implementado el tercer método de Walker-Orin. (los ficheros se proporcionan durante la práctica).

⇒ NOTA: Se recomienda al lector interesado la lectura del artículo de M.W.Walker y D.E.Orin, dónde se presentan el resto de métodos que no han sido usados en este libro.

Walker y Orin resuelven la ecuación general del robot:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + K(q)^T k = \tau \tag{3.34}$$

donde:

$H(q)$	Matriz no singular NxN de los momentos de inercia.
$C(q,\dot{q})$	Matriz NxN que contabiliza los efectos de las aceleraciones centrífugas y de Coriolis.
$G(q)$	Vector Nx1 que contabiliza los efectos de la gravedad.
$K(q)$	Matriz Jacobiana 6xN que especifica los pares (fuerzas) creados en cada articulación debido a las fuerzas y momentos externos aplicados sobre el elemento N.
k	Vector 6x1 de los momentos y fuerzas externas ejercidos sobre el elemento N.
τ	Vector Nx1 de los pares (fuerzas) de cada articulación. (tau en los ejemplos anteriores).
q	Vector Nx1 de las variables articulares.

De la ecuación (3.34) se observa que las fuerzas y pares en las articulaciones son funciones lineales de las aceleraciones articulares. Se define *b* como un vector equivalente a los efectos de la gravedad, las aceleraciones centrifugas y de Coriolis y las fuerzas externas aplicadas sobre el elemento N:

$$b = C(q,\dot{q})\dot{q} + G(q) + K(q)^T k \tag{3.35}$$

Entonces, la ecuación (3.34) se puede poner como:

$$H(q)\ddot{q} = (\tau - b) \tag{3.36}$$

El vector *b* se puede calcular fácilmente utilizando la función NEWTONEULER del apartado anterior. Se llama a la función NEWTONEULER con los parámetros q, qp, masaext e inerciaext (efectos de las fuerzas externas) correspondientes a la configuración estudiada, y colocando a cero el parámetro qpp correspondiente a la aceleración. De la ecuación (3.36) se observa que en este caso $\tau = b$.

En el cálculo de la matriz $H(q)$ es donde los diferentes métodos de Walker y Orin difieren.

Para el cálculo de $H(q)$ se va a utilizar el tercer método de Walker y Orin. Este algoritmo aprovecha la simetría de la matriz $H(q)$ para calcular la diagonal principal y los términos de la mitad superior. Estos componentes se calculan con el siguiente orden $H_{N,N}$, $H_{N-1,N}$,....., $H_{1,N}$,; $H_{N-1,N-1}$, $H_{N-2,N-1}$,....., $H_{1,N-1}$,;.....etc. Para el cálculo de la articulación *j*, los últimos *N-j+1* elementos aparecen como un cuerpo rígido, luego la articulación *j* es la única con movimiento. En este caso, y conocida la localización del centro de masas y el momento de inercia de este elemento ficticio, la fuerza total F_j y el momento N_j ejercidos en el sistema compuesto por los elementos *j* hasta *N* se calcula como:

$$\left.\begin{aligned} F_j &= M_j \dot{v}_j = M_j (z_{j-1} \times c_j) = z_{j-1} \times (M_j c_j) \\ N_j &= E_j z_{j-1} \end{aligned}\right\} \quad \text{para } j \text{ rotacional} \qquad (3.37)$$

$$\left.\begin{aligned} F_j &= M_j v_{j-1} \\ N_j &= 0 \end{aligned}\right\} \quad \text{para } j \text{ traslacional} \qquad (3.38)$$

donde:

- M_j masa total de los elementos *j* hasta *N*
- $\dot{v}_j$ aceleración lineal del centro de masas del cuerpo rígido compuesto por los elementos *j* hasta *N*
- c_j localización del c.d.m. del cuerpo rígido compuesto respecto a las coordenadas del elemento *j-1*
- E_j la matriz de momentos de inercia del cuerpo rígido compuesto por los elementos *j* hasta *N*

Puesto que F_i y N_i son cero para $i<j$, utilizando las ecuaciones (3.24) y (3.25) se obtiene que:

$$\left.\begin{aligned} f_i &= f_{i+1} \\ n_i &= n_{i+1} + p_i^* \times f_{i+1} \end{aligned}\right\} \quad \text{para } i=1 \ldots j-1 \quad \text{y}$$

$$\left.\begin{aligned} f_j &= F_j \\ n_j &= N_j + c_j + F_j \end{aligned}\right\} \qquad (3.39)$$

luego empezando con $i=j$, las ecuaciones anteriores pueden ser usadas para obtener n_i y f_i para $i \leq j$.

Los componentes de la matriz de momentos de inercia a lo largo de la columna j son iguales a los pares y fuerzas generados en las articulaciones. Luego para $i \leq j$:

$$H_{ij} \begin{cases} z_{i-1} n_i & \text{para articulación } i \text{ rotacional} \\ \\ z_{i-1} f_i & \text{para articulación } j \text{ traslacional} \end{cases} \qquad (3.40)$$

Los parámetros M_j,c_j yE_j utilizados en las ecuaciones (3.37) y (3.38) se calculan con las siguientes ecuaciones:

- para el elemento N:

$$M_N = m_N$$
$$c_N = s_N + p_N^*$$
$$E_N = J_N$$

- y para el resto:

$$M_j = M_{j+1} + m_j$$
$$c_j = \frac{1}{M_j}\left[m_j\left(s_j + p_j^*\right) + M_{j+1}\left(c_{j+1} + p_j^*\right)\right]$$
$$E_j = E_{j-1} + M_{j-1}\left[\left(c_{j+1} + p_j^* - c_j\right) * \left(c_{j+1} + p_j^* - c_j\right) I - \left(c_{j+1} + p_j^* - c_j\right)\left(c_{j+1} + p_j^* - c_j\right)^T\right] +$$
$$+ J_j + m_j\left[\left(s_j + p_j^* - c_j\right) * \left(s_j + p_j^* - c_j\right) I - \left(s_j + p_j^* - c_j\right)\left(c_{j+1} + p_j^* - c_j\right)^T\right]$$

donde:

m_j masa del elemento j
s_j posición del centro de masas del elemento i respecto a las coordenadas del elemento j
J_j matriz de momentos de inercia del elemento j
I matriz identidad 3x3

Ejemplo 3.3

Se muestra a continuación un ejemplo en el que se resuelve la dinámica directa del robot prismático de 4 gdl.

Código en MatLab®. A continuación se presenta la función WALKERORIN4, utilizada para resolver la dinámica directa del robot prismático de 4 gdl. Realizando help WALKERORIN4, la función nos informa sobre los parámetros necesarios para realizar los cálculos.

```
% WALKERORIN4 Tercer método de Walker & Orin.
%   QPP = WALKERORIN4(Q, QP, TAU, MASAEXT, INERCIAEXT) calcula la cinemática
%   inversa del robot de 4GDL devolviendo el vector 4x1 que representa la
%   aceleración de cada articulación utilizando el tercer método de Walker y
Orin.
%   Q es el vector 4x1 de variables articulares. QP es el vector 4x1 que
%   representa la velocidad de cada articulación. TAU es el vector 4x1
%   que representa el par de entrada a cada articulación. MASAEXT es
%   la masa de la carga externa. INERCIAEXT es la inercia de la carga externa.
%
%   See also NEWTONEULER4, H4.

function qpp = walkerorin4(q,qp,tau,masaext,inerciaext)

% Se calcula el vector b.
b = newtoneuler4(q,qp,zeros(4,1),9.8,masaext,inerciaext);

% Se calcula la matriz de momentos de inercia H.
H = h4(q,masaext,inerciaext);

% Se calcula el vector de aceleración de cada articulación.
qpp = inv(H)*(tau-b);
```

Para comprobar el funcionamiento del código presentado se sugiere al lector que realice tanto la dinámica inversa como la directa del robot estudiado, tal y como muestra el siguiente ejemplo.

```
» q=rand(4,1);
» qp=rand(4,1);
» qpp=rand(4,1);
» m4=3;
» iext=0.05*eye(3);
» tau=newtoneuler4(q,qp,qpp,9.8,m4,iext)

tau =

    0.5229
  160.0619
    2.0591
    0.0315

» acel=walkerorin4(q,qp,tau,m4,iext)

acel =

    0.1389
    0.2028
    0.1987
    0.6038

» qpp

qpp =

    0.1389
    0.2028
    0.1987
    0.6038

»
```

Ejemplo 3.4

Se muestra a continuación un ejemplo en el que se resuelve la dinámica directa del robot rotacional de 6 gdl.

***Código en MatLab*.** A continuación se presenta la función WALKERORIN6, utilizada para resolver la dinámica directa del robot rotacional de 6 gdl. Realizando help WALKERORIN6, la función nos informa sobre los parámetros necesarios para realizar los cálculos.

```
% WALKERORIN6 Tercer método de Walker & Orin.
%   QPP = WALKERORIN6(Q, QP, TAU, MASAEXT, INERCIAEXT) calcula la cinemática
%   inversa del robot de 6GDL devolviendo el vector 6x1 que representa la
%   aceleración de cada articulación utilizando el tercer método de Walker y
Orin.
%   Q es el vector 6x1 de variables articulares. QP es el vector 6x1 que
%   representa la velocidad de cada articulación. TAU es el vector 6x1
%   que representa el par de entrada a cada articulación. MASAEXT es
%   la masa de la carga externa. INERCIAEXT es la inercia de la carga externa.
%
%   See also NEWTONEULER6, H6.

function qpp = walkerorin6(q,qp,tau,masaext,inerciaext)

% Se calcula el vector b.
b = newtoneuler6(q,qp,zeros(6,1),9.8,masaext,inerciaext);

% Se calcula la matriz de momentos de inercia H.
H = h6(q,masaext,inerciaext);

% Se calcula el vector de aceleración de cada articulación.
qpp = inv(H)*(tau-b);
```

```
» q=rand(6,1);
» qp=rand(6,1);
» qpp=rand(6,1);
» m7=3;
» iext=0.05*eye(3);
» tau=newtoneuler6(q,qp,qpp,9.8,m7,iext);
» acel=walkerorin6(q,qp,tau,m7,iext)

acel =

    0.8537
    0.5936
    0.4966
    0.8998
    0.8216
    0.6449

» qpp

qpp =

    0.8537
    0.5936
    0.4966
    0.8998
    0.8216
    0.6449

»
```

z_0 z_1 z_2 z_3

x_0 x_1 x_2 x_3

3.4.- PRACTICA. Simulación péndulo de 3 gdl.

θ_1 y_0 θ_2 y_1 θ_3 y_2 y_3

Utilizando el método de Walker y Orin para el cálculo de la dinámica directa de un robot, se va a simular cual sería el comportamiento de un péndulo de 3GDL, ver figura 3.4, si no se le aplica ningún par en ninguna de las articulaciones. Los parámetros de Denavit – Hartenberg del péndulo se muestran en la tabla 3.2.

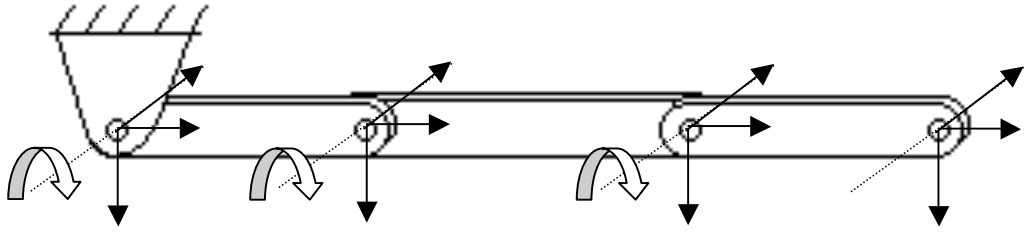


Figura 3.4. Representación D-H del péndulo de 3 gdl

Eslabón	θ_i	d_i	a_i	α_i
1	θ_1	0	1m	0
2	θ_2	0	1m	0
3	θ_3	0	1m	0

Tabla 3.2 Parámetros de D-H para el robot cilíndrico de la figura-3.1

Se ha simulado el comportamiento del péndulo suponiendo que la posición y la velocidad de cada articulación son inicialmente nulas. Así mismo se ha considerado que el péndulo no posee ninguna masa externa.

A continuación se presenta la función SIMULA3, que simula el comportamiento del péndulo de 3 GDL.

```
% SIMULA3   Simulación del péndulo de 3GDL.
%   MAT_Q = SIMULA3(TSIM, PINT) simula el comportamiento del péndulo
%   de 3 GDL suponiendo que no existe ningún par en ninguna de las
%   articulaciones. TSIM indica el tiempo total de la simulación en
%   segundos. PINT es el   periodo de integración en segundos. Este
%   periodo se utiliza para dividir el tiempo de simulación en intervalos.
%   En MAT_Q se devuelven las coordenadas articulares del robot, almacenadas
%   por columnas, correspondientes a cada intervalo de tiempo. MAT_Q es una
%   matriz de 3 filas y una columna por cada intervalo de simulación.
%
%   See also WALKERORINRUNGE3.

function mat_q = simula3(tsim, pint)

% Características de la carga externa
masaext = 0;
inerciaext = zeros(3);

% Posición y velocidad inicial de las articulaciones del robot
q  = zeros(3,1);
qp = zeros(3,1);
```

```
% Vector de tiempo dividido en intervalos
t = 0:pint:tsim;
% Número de intervalos + 1
n = length(t);

% Inicialización de la matriz de coordenadas articulares
mat_q(:,1) = q;

% Se calcula las coordenadas articulares del robot
% en cada intervalo de la simulación
for i=2:n
   % Se calcula la posición y la velocidad de cada articulación
   % combinando el tercer método de Walker & Orin con el método
   % de integración de Runge-Kutta.
    [q qp] = walkerorinrunge3(q,qp,zeros(3,1),t(i-1),t(i),masaext,inerciaext);

    % Se almacenan las coordenadas articulares
    mat_q(:,i) = q;
end
```

Hay que destacar que para actualizar la posición y la velocidad de cada articulación se utiliza el método de Walker y Orin en combinación con el método de integración de Runge-Kutta de orden cuatro. Esto se debe a que el tercer método de Walker y Orin proporciona la aceleración de cada articulación, por lo que es necesario utilizar un método de integración para obtener la posición y la velocidad de cada articulación. A continuación se muestra el código en Matlab® de la función WALKERORINRUNGE3, que combina el método de Walker y Orin junto con el de Runge-Kutta para el cálculo de la posición y velocidad de cada articulación:

```
% WALKERORINRUNGE3  Tercer método de Walker & Orin.
%   [QFIN, QPFIN] = WALKERORINRUNGE3(QINI, QPINI, PAR, T0, TF, MASAEXT,
%   INERCIAEXT) calcula la posición y la velocidad de cada articulación
%   del péndulo de 3 GDL combinando el tercer método de Walker & Orin
%   con el método de integración de Runge-Kutta de orden cuatro.
%   QINI es el vector 3x1 de variables articulares en el instante de
%   tiempo T0. QPINI es el vector 3x1 que representa la velocidad de
%   cada articulación  en el instante de tiempo T0. PAR es el vector
%   3x1 que representa el par de entrada a cada articulación. T0 y TF
%   representan, respectivamente, los valores de inicio y de fin del
%   intervalo de tiempo. MASAEXT es la masa de la carga externa.
%   INERCIAEXT es la inercia de la carga externa. En QFIN y QPFIN se
%   devuelven, respectivamente, los vectores 3x1 de posición y
%   velocidad de cada articulación en el instante de tiempo TF.
%
%   See also WALKERORIN3.

function [qfin, qpfin] =
walkerorinrunge3(qini,qpini,par,t0,tf,masaext,inerciaext);

h = (tf-t0);

% Primer termino.
t1 = t0;
q1 = qini;
qp1= qpini;
v1 = h*qpini;
w1 = h*walkerorin3(q1,qp1,par,masaext,inerciaext);

% Segundo termino.
t2 = t0 + .5*h;
q2 = qini + .5*v1;
qp2= qpini + .5*w1;
```

```
v2 = h*(qpini + .5*w1);
w2 = h*walkerorin3(q2,qp2,par,masaext,inerciaext);

% Tercer termino.
t3 = t0 + .5*h;
q3 = qini + .5*v2;
qp3= qpini + .5*w2;
v3 = h*(qpini + .5*w2);
w3 = h*walkerorin3(q3,qp3,par,masaext,inerciaext);

% Cuarto termino.
t4 = t0 + h;
q4 = qini + v3;
qp4= qpini + w3;
v4 = h*(qpini + w3);
w4 = h*walkerorin3(q4,qp4,par,masaext,inerciaext);

% Formula de Runge-Kutta.
qfin =  qini + (v1 + 2*v2 + 2*v3 + v4)/6;
qpfin =  qpini + (w1 + 2*w2 + 2*w3 + w4)/6;

% Redondeo para evitar oscilacion numerica.
qfin = round(qfin*1e13)/1e13;
qpfin = round(qpfin*1e13)/1e13;
```

En el código anterior se hacen varias llamadas a la función WALKERORIN3. Esta función calcula la dinámica directa del péndulo de 3GDL utilizando el tercer método de Walker y Orin. Para el cálculo de la dinámica directa se utilizan las funciones NEWTONEULER3 y H3.

Para simular el comportamiento del péndulo de 3GDL durante 1 segundo considerando un periodo de integración de 0.001 segundos se procedería de la siguiente manera en Matlab®:

```
» mat_q=simula3(1,0.001);
»
```

Para comprobar visualmente el comportamiento del péndulo de 3GDL se ha desarrollado la función ANIMACION3. El código de esta función se muestra a continuación:

```
%  ANIMACION3   Animación del movimiento del péndulo de 3GDL.
%     ANIMACION3(MAT_Q) realiza la animación del movimiento del
%     péndulo de 3GDL a partir de las coordenadas articulares
%     almacenadas en la matriz MAT_Q. MAT_Q contiene 3 filas
%     y una columna para cada disposición del robot durante el
%     movimiento.

function animacion3(mat_q)

%  Parámetros Denavit-Hartenberg del robot
d    = [0 0 0];
a    = [1 1 1];
alfa = [0 0 0];

% Vector de posicion (x, y) del sistema de coordenadas de referencia
x0 = 0;   y0 = 0;
```

```
% Se dibuja el sistema de coordenadas de referencia. Se asigna el modo XOR
para borrar
% sólo el robot dibujado anteriormente.
p = plot(x0,y0,'EraseMode','xor');
% Se asigna una rejilla a los ejes
grid;
% Se establecen los límites de los ejes
axis([-3.2 3.2 -3.1 1]);

% Mantiene el gráfico actual
hold on;

% Número de columnas de la matriz
n = size(mat_q,2);

% Se dibuja la disposición del robot correspondiente a cada columna
for i=1:n
   % Variables articulares del brazo robot
   teta1 = mat_q(1,i);
    teta2 = mat_q(2,i);
    teta3 = mat_q(3,i);

   %   Matrices de transformación homogénea entre sistemas de coordenadas
consecutivos
    A01 = denavit(teta1, d(1), a(1), alfa(1));
    A12 = denavit(teta2, d(2), a(2), alfa(2));
    A23 = denavit(teta3, d(3), a(3), alfa(3));

    %   Matrices de transformación del primer sistema al correspondiente
    A02 = A01 * A12;
    A03 = A02 * A23;

    %   Vector de posicion (x, y) de cada sistema de coordenadas
    x1 = A01(1,4);      y1 = A01(2,4);
    x2 = A02(1,4);      y2 = A02(2,4);
    x3 = A03(1,4);      y3 = A03(2,4);

    %   Se dibuja el robot
    x = [x0 x1 x2 x3];
    y = [y0 y1 y2 y3];
    set(p,'XData',x,'YData',y);
   % Se fuerza a MATLAB a actualizar la pantalla
   drawnow;
end
```

Para realizar la animación del comportamiento simulado anteriormente se procedería de la siguiente manera en Matlab:

```
» animacion3(mat_q)
»
```

Al ejecutar la función en Matlab® nos aparecerá una ventana similar a mostrada en la figura 3.5 en la que se visualizará la animación.

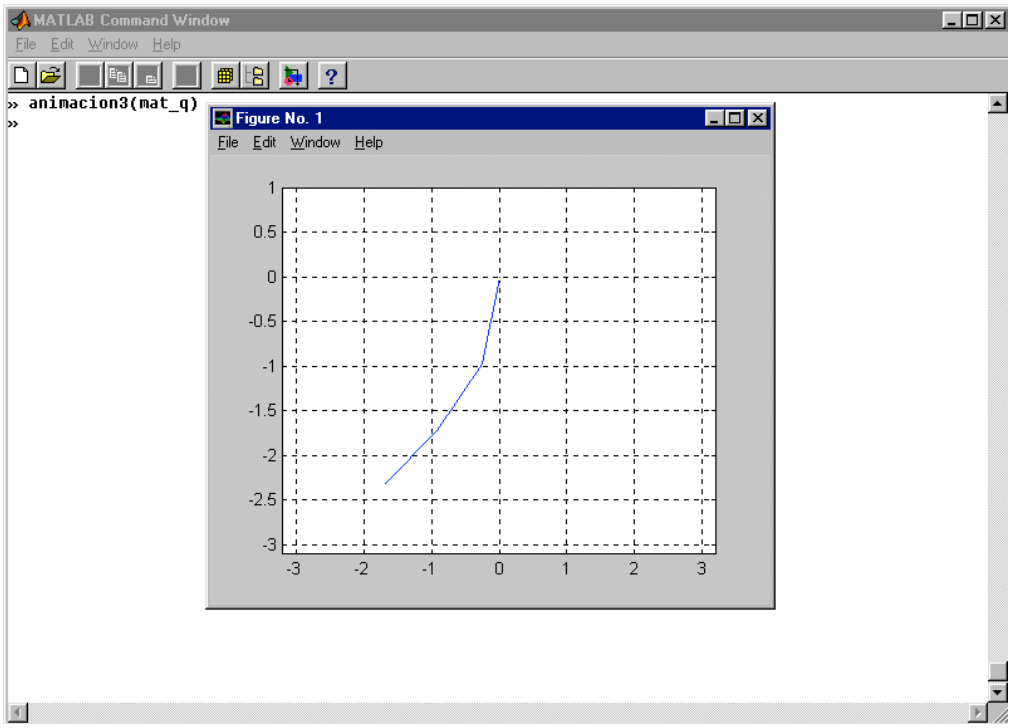


Figura. 3.5. Aspecto de la visualización de la animación del péndulo de 3GDL.