Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

TE3001B: Fundamentación de robótica (Gpo 101)

Profesor: Dr. Alfredo García Suárez

Actividad 1 (Velocidades Lineales y angulares) Robot Planar 3GDL

Marcos Allen Martínez Cortés

A01737939

16 de febrero de 2026

Actividad 1 (Velocidades Lineales y angulares) Robot Planar 3GDL

Elaborado por: Marcos Allen Martínez Cortés | A01737939

Para obtener el vector de velocidades linear y angular para la configuración de un robot planar de 3GDL (grados de libertad), como el que se muestra en la imagen a continuación, primero tenemos que analizarlo.

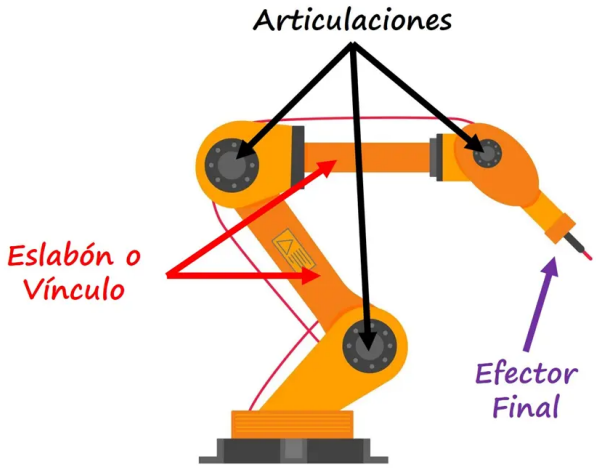


Imagen de: https://makeblock.com.ar/wp-content/uploads/que-son-los-eslabones-y-articulaciones.webp

Dicho robot está conformado por tres articulaciones rotacionales. El sistema de referencia fijo (Xo, Yo, Zo) se coloca en la base del robot, de tal manera que el eje Zo es perpendicular al plano de la imagen.

Para poder obtener la velocidad lineal y angular aplicaremos la cinemática diferencial, cuyo objetivo es establecer las relaciones entre las velocidades de las articulaciones y las velocidades lineares y angulares del efector final.

De este modo, para realizarlo en MATLAB empezaremos por limpiar el espacio de trabajo (y adicionales) y procederemos a declarar las variables simbólicas a utilizar, que incluyen:

- Las funciones theta con respecto al tiempo -th(t)- para las tres juntas, esto para la rotación de cada junta.
- t para el tiempo.
- Las tres l, que son las longitudes de los eslabones.

```
% Limpieza de pantalla
clear all
close all
clc

% Declaración de variables simbólicas
syms th1(t) th2(t) th3(t) t l1 l2 l3
```

Adicionalmente, también es necesario establecer la configuración del robot, considerando si las juntas son rotacionales o prismáticas.

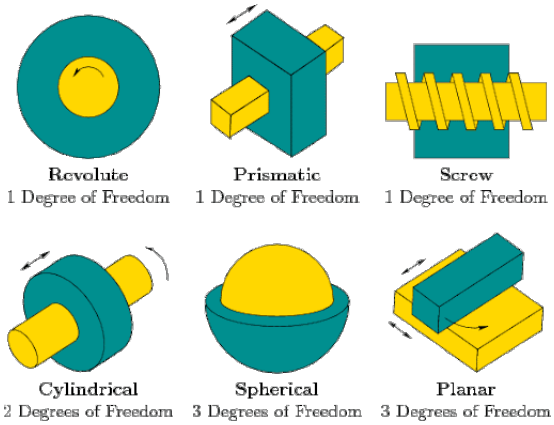


Imagen de: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRV4iX26CWioV6KqOKe7ApSo0ivnPhRJpvucw&s

En este caso las tres juntas son rotacionales.

```
% Configuración del robot, 0 para junta rotacional, 1 para junta prismática
RP = [0 0 0];
```

Adicionalmente, estableceremos (con respecto a la cantidad de juntas) la variable de grados de libertad que será utilizada más adelante.

```
% Número de grado de libertad del robot
GDL = size(RP,2);
GDL_str = num2str(GDL); % Convertir a string
```

Posteriormente, tenemos que crear el vector de coordenadas articulares; con un elemento por cada articulación/junta.

```
% Creamos el vector de coordenadas articulares
Q = [th1, th2, th3];
disp('Coordenadas generalizadas');
```

```
Coordenadas generalizadas
```

```
pretty(Q);
```

```
(th1(t), th2(t), th3(t))
```

Y obtendremos el vector de velocidades generalizadas, creándolo con la derivada del vector de coordenadas. Esto, ya que la derivada de la posición con respecto al tiempo es la velocidad.

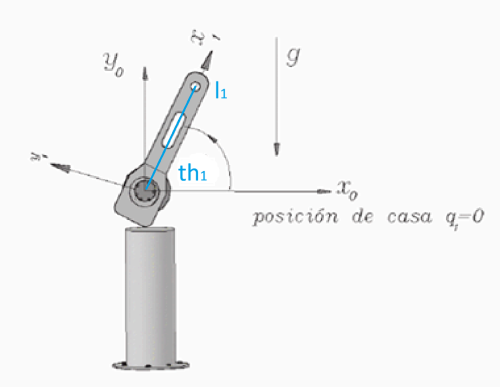
```
% Creamos el vector de velocidades generalizadas
Qp = diff(Q, t);
disp('Velocidades generalizadas');
```

```
Velocidades generalizadas
```

```
pretty(Qp);
```

```
/  d          d          d        \
| -- th1(t), -- th2(t), -- th3(t) |
\ dt         dt         dt        /
```

Para continuar, necesitamos obtener el vector de posición de la junta 1 respecto a 0 (sistema de referencia). Como se estableció, el sistema de referencia fijo (Xo, Yo, Zo) se coloca en la base del robot, de tal manera que el eje Zo es perpendicular al plano de la imagen.



De esta forma, podemos utilizar las funciones trigonométricas para obtener la posición de la junta 1 respecto a 0 (sistema de referencia).

- l1 es la longitud del primer eslabón.
- theta1 es el ángulo respecto al eje x del sistema base (x0).

De esta manera, podemos utilizar seno y coseno para expresar la posición y crear el vector de posición o vector de traslación. Además, z es 0 debido a que se trata de un robot planar.

```
% Junta 1
% Posición de la junta 1 respecto a 0
P(:,:,1) = [l1*cos(th1); l1*sin(th1); 0];
```

Adicionalmente, utilizaremos la matriz de rotación, que representa la rotación de theta grados del plano en sentido antihorario.

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\operatorname{sen}\theta \\ 0 & \operatorname{sen}\theta & \cos\theta \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \operatorname{sen}\theta \\ 0 & 1 & 0 \\ -\operatorname{sen}\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

En este caso, la rotación es alrededor del eje Z, ya que es el eje que permanece fijo (z0 = z1). Entonces, para la matriz de rotación de la junta 1 respecto a 0, utilizaremos la matriz de rotación alrededor de Z.

```
% Matriz de rotación de la junta 1 respecto a 0
R(:,:,1) = [cos(th1) -sin(th1) 0;
            sin(th1)  cos(th1) 0;
            0         0        1];
```

Para la junta 2 y tres realizaremos lo mismo; con diferencia de que el vector de posición/traslación y la matriz de rotación de la junta 2 es respecto a la 1, y para la 3 es respecto a la 2.

```
% Junta 2
% Posición de la junta 2 respecto a 1
P(:,:,2) = [l2*cos(th2); l2*sin(th2); 0];
% Matriz de rotación de la junta 2 respecto a 1
R(:,:,2) = [cos(th2) -sin(th2) 0;
            sin(th2)  cos(th2) 0;
            0         0        1];

% Junta 3
% Posición de la junta 3 respecto a 2
P(:,:,3) = [l3*cos(th3); l3*sin(th3); 0];
% Matriz de rotación de la junta 3 respecto a 2
R(:,:,3) = [cos(th3) -sin(th3) 0;
            sin(th3)  cos(th3) 0;
            0         0        1];
```

Adicionalmente, crearemos un vector de ceros, que utilizaremos para la matriz de transformación homogénea.

```
% Creamos un vector de ceros
Vector_Zeros = zeros(1, 3);
```

Con lo realizado adicionalmente, podemos crear las matrices de transformación homogénea para cada una de las juntas.

La matriz de transformación homogénea relaciona dos sistemas de coordenadas y se define mediante la transformación canónica establecida por los parámetros Denavit-Hartenberg.

Tabla 4.1 Parámetros Denavit-Hartenberg.

Características de eslabones	
l_i	longitud del eslabón i-ésimo
d_i	articulaciones lineales o prismáticas
α_i	ángulo entre los ejes z_{i-1} y z_i medido con respecto al eje x_i.
θ_i	articulaciones rotacionales; representa el ángulo entre los ejes x_{i-1} y x_i medido alrededor del eje z_{i-1}.

De este modo, la matriz de transformación homogénea se constituye de la siguiente manera:

$$H = \begin{bmatrix} R_{z,\theta} & \boldsymbol{d}_0^1 \\ \mathbf{0}^T & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta_i) & -\operatorname{sen}(\theta_i)\cos(\alpha_i) & \operatorname{sen}(\theta_i)\operatorname{sen}(\alpha_i)) & l_i\cos(\theta_i) \\ \operatorname{sen}(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\operatorname{sen}(\alpha_i) & l_i\operatorname{sen}(\theta_i) \\ 0 & \operatorname{sen}(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{ccc} \text{Matriz de rotación} & \vdots & \text{Vector de Traslación} \\ \cdots & \vdots & \cdots \\ \mathbf{0}^T & & 1 \end{array}\right]$$

Para esto, comenzaremos por inicializar las matrices de transformación homogénea tanto locales como globales. Y, también inicializaremos las posiciones (vectores de traslación) y las matrices de rotación (junto con sus inversas) vistas desde el sistema de referencia.

```
% Inicializamos las matrices de transformación Homogénea locales
A(:,:,GDL) = simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
% Inicializamos las matrices de transformación Homogénea globales
T(:,:,GDL) = simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
% Inicializamos las posiciones vistas desde el marco de referencia inercial
PO(:,:,GDL) = P(:,:,GDL);
% Inicializamos las matrices de rotación vistas desde el marco de referencia
inercial
RO(:,:,GDL) = R(:,:,GDL);
% Inicializamos las INVERSAS de las matrices de rotación vistas desde el marco de
referencia inercial
RO_inv(:,:,GDL) = R(:,:,GDL);
```

A continuación, dentro del ciclo for recorremos cada grado de libertad; es decir, cada junta.

Primero construimos la matriz de transformación homogénea local (A), tal como se explicó previamente, y esto para poder construir la global (T).

Después, las multiplica acumulativamente para ir construyendo la matriz global (T). Es decir, T1 queda de la misma manera (A1), la T2 es A1*A2, y la T3 es T2*A3 = A1*A2*A3.

Esto construye la matriz de transformación homogénea global.

Al final se extraen las posiciones (vectores de traslación) y las matrices de rotación (junto con su inversa) de la matriz de transformación global; es decir, descomponemos la matriz de transformación global para obtener dichos datos desde el sistema de referencia inercial; o de otra forma, la matriz de tranformación homogénea global permite analizar el comportamiento del efector final desde el sistema de referencia.

```
for i = 1:GDL
    i_str= num2str(i);

    % Locales
    disp(strcat('Matriz de Transformación local A', i_str));
    A(:,:,i) = simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
    pretty(A(:,:,i));
```

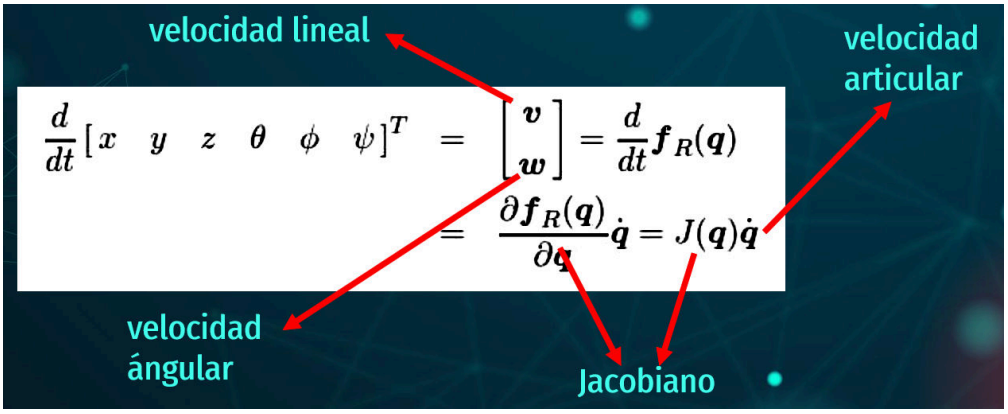
```
    % Globales
    try
        T(:,:,i)= T(:,:,i-1)*A(:,:,i);
    catch
        T(:,:,i)= A(:,:,i);
    end
    disp(strcat('Matriz de Transformación global T', i_str));
    T(:,:,i)= simplify(T(:,:,i));
    pretty(T(:,:,i))

    RO(:,:,i)= T(1:3,1:3,i);
    RO_inv(:,:,i)= transpose(RO(:,:,i));
    PO(:,:,i)= T(1:3,4,i);
    %pretty(RO(:,:,i));
    %pretty(RO_inv(:,:,i));
    %pretty(PO(:,:,i));
end
```

```
Matriz de Transformación local A1
/ cos(th1(t)), -sin(th1(t)), 0, l1 cos(th1(t)) \
|                                              |
| sin(th1(t)),  cos(th1(t)), 0, l1 sin(th1(t)) |
|                                              |
|      0,            0,      1,        0       |
|                                              |
\      0,            0,      0,        1       /
Matriz de Transformación global T1
/ cos(th1(t)), -sin(th1(t)), 0, l1 cos(th1(t)) \
|                                              |
| sin(th1(t)),  cos(th1(t)), 0, l1 sin(th1(t)) |
|                                              |
|      0,            0,      1,        0       |
|                                              |
\      0,            0,      0,        1       /
Matriz de Transformación local A2
/ cos(th2(t)), -sin(th2(t)), 0, l2 cos(th2(t)) \
|                                              |
| sin(th2(t)),  cos(th2(t)), 0, l2 sin(th2(t)) |
|                                              |
|      0,            0,      1,        0       |
|                                              |
\      0,            0,      0,        1       /
Matriz de Transformación global T2
/ #2, -#1, 0, l1 cos(th1(t)) + l2 #2 \
|                                     |
| #1,  #2, 0, l1 sin(th1(t)) + l2 #1 |
|                                     |
|  0,  0,  1,           0             |
|                                     |
\  0,  0,  0,           1            /

where

   #1 == sin(th1(t) + th2(t))

   #2 == cos(th1(t) + th2(t))
Matriz de Transformación local A3
```

```
/ cos(th3(t)), -sin(th3(t)), 0, l3 cos(th3(t)) \
|                                               |
| sin(th3(t)),  cos(th3(t)), 0, l3 sin(th3(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
Matriz de Transformación global T3
/ #2, -#1, 0, l1 cos(th1(t)) + l3 #2 + l2 cos(th1(t) + th2(t)) \
|                                                               |
| #1,  #2, 0, l1 sin(th1(t)) + l3 #1 + l2 sin(th1(t) + th2(t)) |
|                                                               |
|  0,  0,  1,                          0                        |
|                                                               |
\  0,  0,  0,                          1                       /

where

   #1 == sin(th1(t) + th2(t) + th3(t))

   #2 == cos(th1(t) + th2(t) + th3(t))
```

```
% Calculamos la matriz de transformación del marco de referencia inercial
% visto desde el actuador final
% disp(strcat('Matriz de Transformación T', GDL_str,'_O calculada de forma
manual'));
% RF_O=RO_inv(:,:,GDL);
% PF_O=-RF_O*PO(:,:,GDL);
% TF_O= simplify([RF_O PF_O; Vector_Zeros 1]);
% pretty(TF_O);

% disp(strcat('Matriz de Transformación T', GDL_str,'_O calculada de forma
automática'));
% pretty(simplify(inv(T(:,:,GDL))));
```

Con esto, tendremos que construir el Jacobiano, que establece la relación entre las velocidades articulares y la velocidad linear y angular en el extremo el efector final del robot, expresadas en el sistema inercial del robot. Es decir, sabiendo que tan rápido se mueven las articulaciones (juntas) del robot, podemos conocer qué tan rápido se mueve y rota el efector final respecto al sistema de referencia.



Primero lo haremos de forma diferencial, calculando primero el Jacobiano lineal a través de las derivadas parciales de las posiciones en x, en y, y en z respecto a th1, th2 y th3; para unirlas como se muestra en la imagen y crear la matriz del Jacobiano lineal.

$$f(x_1, x_2, \ldots, x_n) = (f_1, f_2, \ldots, f_m)$$

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

```
% Calculamos el jacobiano lineal de forma diferencial
disp('Jacobiano lineal obtenido de forma diferencial');
```

```
Jacobiano lineal obtenido de forma diferencial
```

```
% Derivadas parciales de x respecto a th1, th2 y th3
Jv11 = functionalDerivative(PO(1,1,GDL), th1);
Jv12 = functionalDerivative(PO(1,1,GDL), th2);
Jv13 = functionalDerivative(PO(1,1,GDL), th3);
% Derivadas parciales de y respecto a th1, th2 y th3
Jv21 = functionalDerivative(PO(2,1,GDL), th1);
Jv22 = functionalDerivative(PO(2,1,GDL), th2);
Jv23 = functionalDerivative(PO(2,1,GDL), th3);
% Derivadas parciales de z respecto a th1, th2 y th3
Jv31 = functionalDerivative(PO(3,1,GDL), th1);
Jv32 = functionalDerivative(PO(3,1,GDL), th2);
Jv33 = functionalDerivative(PO(3,1,GDL), th3);

% Creamos la matriz del Jacobiano lineal
jv_d = simplify([Jv11 Jv12 Jv13;
                 Jv21 Jv22 Jv23;
                 Jv31 Jv32 Jv33]);
pretty(jv_d);
```

```
/ - l1 sin(th1(t)) - #1 - l2 sin(th1(t) + th2(t)), - #1 - l2 sin(th1(t) + th2(t)), -#1 \
|                                                                                    |
|  l1 cos(th1(t)) + #2 + l2 cos(th1(t) + th2(t)),   #2 + l2 cos(th1(t) + th2(t)),   #2 |
|                                                                                    |
\                       0,                                      0,                0  /

where

   #1 == l3 sin(th1(t) + th2(t) + th3(t))

   #2 == l3 cos(th1(t) + th2(t) + th3(t))
```

Continuamos calculando el Jacobiano ahora de forma analítica.

Inicializamos las variables y después en el ciclo for iteramos sobre todas las articulaciones.

Aquí tenemos dos casos según el tipo de articulación, aunque en este caso, las tres articulaciones o juntas son rotacionales.

Lo que hace en el caso de las juntas rotacionales es:

- Para la parte angular, es la tercera columna de la matriz de rotación de la articulación anterior; es decir, el eje Z del sistema anterior, ya que la rotación se produce alrededor de ese eje.
- Para la parte lineal, se calcula el producto cruz entre el eje Z y el vector de posición, lo que representa que cuando una articulación rota, hay un movimiento circular y la velocidad lineal es tangencial a esa trayectoria.

```
% Calculamos el jacobiano lineal de forma analítica
Jv_a(:,GDL) = PO(:,:,GDL);
Jw_a(:,GDL) = PO(:,:,GDL);

for k= 1:GDL
    if RP(k)==0 %Casos: articulación rotacional
        %Para las juntas de revolución
        try
            Jv_a(:,k) = cross(RO(:,3,k-1), PO(:,:,GDL)-PO(:,:,k-1));
            Jw_a(:,k) = RO(:,3,k-1);
        catch
            Jv_a(:,k) = cross([0,0,1], PO(:,:,GDL));
            Jw_a(:,k) = [0,0,1];
        end
    %Para las juntas prismáticas
    elseif RP(k)==1 %Casos: articulación prismática
    %
        try
            Jv_a(:,k)= RO(:,3,k-1);
        catch
            Jv_a(:,k)=[0,0,1];
        end
            Jw_a(:,k)=[0,0,0];
    end
end
```

Finalmente, simplificamos los Jacobianos obtenidos de forma analítica. Y calculamos las velocidades lineales y angulares mediante el Jacobiano lineal y angular, utilizando el vector de velocidades generalizadas.

```
Jv_a = simplify (Jv_a);
Jw_a = simplify (Jw_a);
disp('Jacobiano lineal obtenido de forma analítica');
```

Jacobiano lineal obtenido de forma analítica

```
pretty(Jv_a);
```

```
/ - l1 sin(th1(t)) - #1 - l2 sin(th1(t) + th2(t)), - #1 - l2 sin(th1(t) + th2(t)), -#1 \
|                                                                                    |
|  l1 cos(th1(t)) + #2 + l2 cos(th1(t) + th2(t)),   #2 + l2 cos(th1(t) + th2(t)),   #2 |
|                                                                                    |
\                        0,                                        0,               0  /
```

where

```
   #1 == l3 sin(th1(t) + th2(t) + th3(t))

   #2 == l3 cos(th1(t) + th2(t) + th3(t))
```

```
disp('Jacobiano ángular obtenido de forma analítica');
```

Jacobiano ángular obtenido de forma analítica

```
pretty(Jw_a);
```

```
/ 0, 0, 0 \
|         |
| 0, 0, 0 |
|         |
\ 1, 1, 1 /
```

```
disp('Velocidad lineal obtenida mediante el Jacobiano lineal');
```

Velocidad lineal obtenida mediante el Jacobiano lineal

```
V=simplify (Jv_a*Qp');
pretty(V);
```

```
/ - #4 (l3 sin(#1) + l2 sin(#2)) - #5 (l1 sin(th1(t)) + l3 sin(#1) + l2 sin(#2)) - l3 #3 sin(#1) \
|                                                                                                 |
|  #4 (l3 cos(#1) + l2 cos(#2)) + #5 (l1 cos(th1(t)) + l3 cos(#1) + l2 cos(#2)) + l3 #3 cos(#1)  |
|                                                                                                 |
\                                                 0                                               /
```

where

```
   #1 == th1(t) + th2(t) + th3(t)

   #2 == th1(t) + th2(t)

         _________
          d
   #3 == -- th3(t)
         dt


         _________
          d
   #4 == -- th2(t)
         dt


         _________
          d
```

```
   #5 == -- th1(t)
         dt
```

```
disp('Velocidad angular obtenida mediante el Jacobiano angular');
```

```
Velocidad angular obtenida mediante el Jacobiano angular
```

```
W=simplify (Jw_a*Qp');
pretty(W);
```

```
/                0                \
|                                 |
|                0                |
|                                 |
| _________   _________   _______ |
|  d           d           d      |
| -- th1(t) + -- th2(t) + -- th3(t) |
\ dt          dt          dt      /
```

Con esto podemos obtener la velocidad lineal y angular, cuya interpretación se encuentra a continuación.

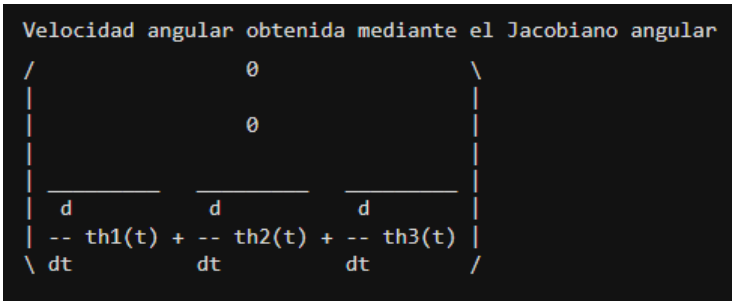
- **Velocidad lineal**:

```
Velocidad lineal obtenida mediante el Jacobiano lineal

/ - #4 (l3 sin(#1) + l2 sin(#2)) - #5 (l1 sin(th1(t)) + l3 sin(#1) + l2 sin(#2)) - l3 #3 sin(#1) \
|                                                                                               |
|  #4 (l3 cos(#1) + l2 cos(#2)) + #5 (l1 cos(th1(t)) + l3 cos(#1) + l2 cos(#2)) + l3 #3 cos(#1)  |
|                                                                                               |
\                                              0                                              /

where

   #1 == th1(t) + th2(t) + th3(t)

   #2 == th1(t) + th2(t)

         _________
          d
   #3 == -- th3(t)
         dt

         _________
          d
   #4 == -- th2(t)
         dt

         _________
          d
   #5 == -- th1(t)
         dt
```

El tercer componente (Vz) es 0 ya que se trata de un robot planar, y al no haber movimiento sobre ese eje, la velocidad será cero.

Y, para Vx y Vy, podemos interpretarlos de forma que el efector se mueve en el plano XY y que la velocidad lineal del efector final está relacionada con las velocidades articulares en el espacio cartesiano; es decir, el Jacobiano nos permite conocer que la velocidad lineal del efector final depende de las longitudes de los eslabones y las posiciones/velocidades de las articulaciones.

- **Velocidad angular**:



Para la velocidad angular, como todas las articulaciones (o juntas) rotan sobre el eje Z, no hay rotación sobre el eje X ni sobre el eje Y, por lo que tienen un valor de 0.

Y el efector rota sobre el eje Z con la suma de las velocidades angulares de todas las articulaciones.