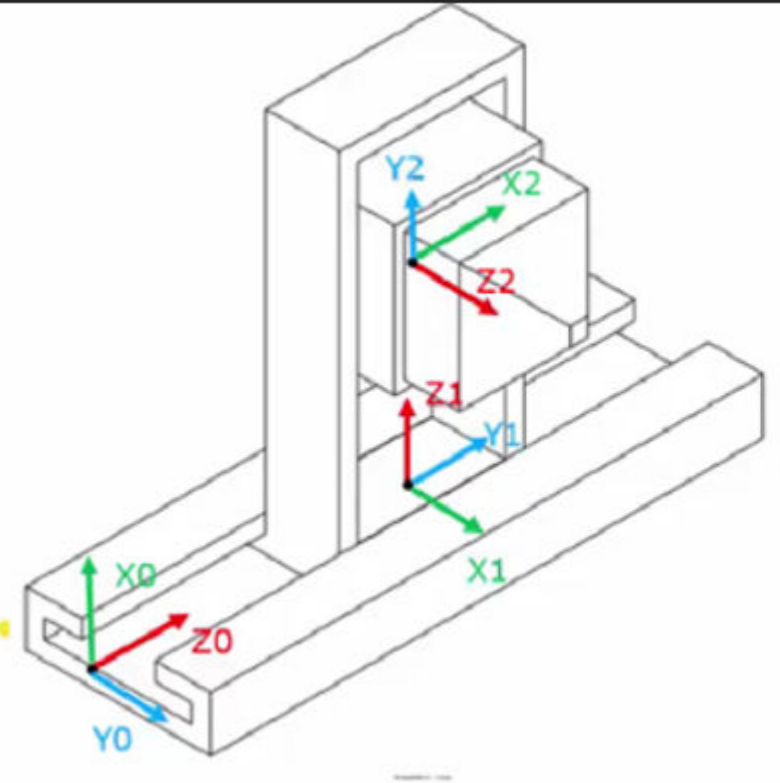


## Actividad 5: Robot\_cartesiano\_3GDL

Daniel Castillo López

A01737357



De articulación 0 a articulación 1:

Movimientos en z con  $90^\circ$

Movimiento en x en  $90^\circ$

De articulación 1 a 2:

Movimiento en y con  $90^\circ$

Movimiento en z con  $90^\circ$

```
clear all
close all
clc
```

Posteriormente, se declaran las variables simbólicas, esenciales para realizar cálculos simbólicos y analíticos.

```
syms t l1(t) l2(t) l3(t)
```

Se configura el robot con dos articulaciones rotacionales, representando su tipo con un vector donde 0 es rotacional y 1 prismático

```
RP=[1 1 1];
```

Creamos el vector de coordenadas generalizadas

```
Q= [l1, l2, l3];
disp('Coordenadas generalizadas'); pretty (Q);
```

```
Coordenadas generalizadas
(l1(t), l2(t), l3(t))
```

Creamos el vector de velocidades generalizadas.

```
Qp= diff(Q, t);
disp('Velocidades generalizadas'); pretty (Qp);
```

```
Velocidades generalizadas
/ d d d \
| -- l1(t), -- l2(t), -- l3(t) |
\ dt dt dt /
```

Posteriormente, definimos los grados de libertad (GDL), es decir, el número de parámetros independientes que definen la posición del extremo del robot, en este caso 4 GDL.

```
GDL= size(RP,2);
GDL_str= num2str(GDL);
```

## Cinemática Directa

La cinemática directa permite encontrar la posición del extremo del robot a partir de sus ángulos articulares. Para ello, se definen los vectores posición (indican la posición relativa de una articulación) y las matrices de rotación (describen la orientación de un sistema de coordenadas respecto a otro) para cada articulación.

### Articulación 1

$$R_z(90) \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_x(90) \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(90) & -\sin(90) \\ 0 & \sin(90) & \cos(90) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

```
%Articulación 1 a Articulación 2
%Posición de la articulación 1 a 2
P(:, :, 1) = [0; 0; l1];

%Matriz de rotación de la junta 1 a 2
R(:, :, 1) = [0 0 1;
              1 0 0;
              0 1 0];
```

### Articulación 2

$$R_y(90) \begin{bmatrix} \cos(90) & 0 & \sin(90) \\ 0 & 1 & 0 \\ -\sin(90) & 0 & \cos(90) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$R_z(90) \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

```
%Articulación 2 a Articulación 3
%Posición de la articulación 2 a 3
P(:, :, 2) = [0; 0; 12];

%Matriz de rotación de la junta 2 a 3
R(:, :, 2) = [0  0  1;
              1  0  0;
              0  1  0];
```

## Articulación 3

Para este caso utilizamos la matriz identidad, esto debido a que no hay transformación adicional entre estos sistemas de referencia, lo que significa que los ejes quedan alineados sin modificaciones en posición u orientación.

```
%Posición de la articulación 3 respecto a 2
P(:, :, 3) = [0; 0; 13];

%Matriz de rotación de la junta 3 respecto a 2
R(:, :, 3) = [ 1  0  0;
              0  1  0;
              0  0  1];
```

Las matrices de transformación homogénea, combinan la rotación y traslación en un solo modelo matemático. Se construyen las matrices locales y globales de transformación para obtener la posición y orientación del extremo del robot en el sistema de referencia inercial.

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

%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);
```

El bucle for recorre cada articulación del robot para calcular sus matrices de transformación homogénea locales y globales. Primero, genera la matriz de transformación local, que describe la posición y orientación de la articulación en relación con la anterior. Luego, obtiene la matriz de transformación global, que representa la posición y orientación en el sistema de referencia inercial, acumulando las transformaciones previas. Finalmente, extrae la matriz de rotación y el vector de posición, fundamentales para la cinemática directa y el cálculo del Jacobiano.

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

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

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

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

Matriz de Transformación local A1

```
/ 0, 0, 1, 0 \
| 1, 0, 0, 0 |
| 0, 1, 0, l1(t) |
| 0, 0, 0, 1 |
\ 0, 0, 0, 1 /
```

Matriz de Transformación global T1

```
/ 0, 0, 1, 0 \
| 1, 0, 0, 0 |
| 0, 1, 0, l1(t) |
```

$$\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$$

Matriz de Transformación local A2

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l2(t) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matriz de Transformación global T2

$$\begin{pmatrix} 0 & 1 & 0 & l2(t) \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & l1(t) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matriz de Transformación local A3

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l3(t) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matriz de Transformación global T3

$$\begin{pmatrix} 0 & 1 & 0 & l2(t) \\ 0 & 0 & 1 & l3(t) \\ 1 & 0 & 0 & l1(t) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

## Cálculo del jacobiano lineal de forma diferencial

El Jacobiano es una matriz que relaciona las velocidades articulares con las velocidades lineales y angulares del extremo del robot. El Jacobiano lineal ( $J_v$ ), relaciona velocidades articulares con velocidad lineal del extremo. Mientras que el Jacobiano angular ( $J_w$ ), relaciona velocidades articulares con velocidad angular.

El Jacobiano diferencial es obtenido mediante derivadas parciales de la posición.

**%Derivadas parciales de x respecto a l1, l2, l3**

```
Jv11= functionalDerivative(P0(1,1,GDL), l1);
Jv12= functionalDerivative(P0(1,1,GDL), l2);
Jv13= functionalDerivative(P0(1,1,GDL), l3);
```

**%Derivadas parciales de y respecto a l1, l2, l3**

```
Jv21= functionalDerivative(P0(2,1,GDL), l1);
Jv22= functionalDerivative(P0(2,1,GDL), l2);
Jv23= functionalDerivative(P0(2,1,GDL), l3);
```

**%Derivadas parciales de z respecto a l1, l2, l3**

```
Jv31= functionalDerivative(PO(3,1,GDL), l1);
Jv32= functionalDerivative(PO(3,1,GDL), l2);
Jv33= functionalDerivative(PO(3,1,GDL), l3);

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

disp('Jacobiano lineal obtenido de forma diferencial'); pretty(jv_d);
```

```
Jacobiano lineal obtenido de forma diferencial
/ 0, 1, 0 \
|           |
| 0, 0, 1 |
|           |
\ 1, 0, 0 /
```

## Cálculo del jacobiano lineal y angular de forma analítica

El jacobiano analítico es calculado mediante productos vectoriales, considerando el tipo de articulación

```
%Inicializamos jacobianos analíticos (lineal y angular)
Jv_a(:,GDL)=PO(:, :,GDL);
Jw_a(:,GDL)=PO(:, :,GDL);
```

Para articulaciones rotacionales, la velocidad lineal se obtiene mediante el producto cruzado entre el eje de rotación y la diferencia de posiciones, mientras que la velocidad angular se toma directamente del eje de rotación. Si no hay una matriz de rotación previa, se usa la identidad y la posición inicial. Para articulaciones prismáticas, la velocidad lineal es el eje de desplazamiento, y la angular es cero, ya que no hay rotación.

```
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

    elseif RP(k)==1 %Casos: articulación prismática
        %Para las juntas prismáticas
        try
            Jv_a(:,k)= RO(:,3,k-1);
        catch
            Jv_a(:,k)=[0,0,1];
        end
        Jw_a(:,k)=[0,0,0];
    end
end
end
```

```
%Despliegue
```

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

```
Jacobiano lineal obtenido de forma analítica  
/ 0, 1, 0 \  
|          |  
| 0, 0, 1 |  
|          |  
\ 1, 0, 0 /
```

```
disp('Jacobiano angular obtenido de forma analítica'); pretty (Jw_a);
```

```
Jacobiano angular obtenido de forma analítica  
/ 0, 0, 0 \  
|          |  
| 0, 0, 0 |  
|          |  
\ 0, 0, 0 /
```

## Velocidades Lineales y Angulares

Finalmente, se obtienen las velocidades multiplicando los jacobianos por el vector de velocidades articulares

La velocidad lineal describe el desplazamiento del efector.

```
V=simplify (Jv_a*Qp');  
disp('Velocidad lineal obtenida mediante el Jacobiano lineal'); pretty(V);
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal  
/ ----- \  
| d      |  
| -- 12(t) |  
| dt      |  
| ----- |  
| d      |  
| -- 13(t) |  
| dt      |  
| ----- |  
| d      |  
| -- 11(t) |  
| dt      |  
\ ----- /
```

Mientras que la velocidad angular describe la rotación del efector, como en nuestro robot, no se presnetan rotación , no tienen el mismo efecto como en otros, asi que su descripción se basa totalmente en cero, al no existir una rotación.

```
W=simplify (Jw_a*Qp');  
disp('Velocidad angular obtenida mediante el Jacobiano angular'); pretty(W);
```

```
Velocidad angular obtenida mediante el Jacobiano angular  
/ 0 \  
|  |  
| 0 |
```

| |  
 \ 0 /