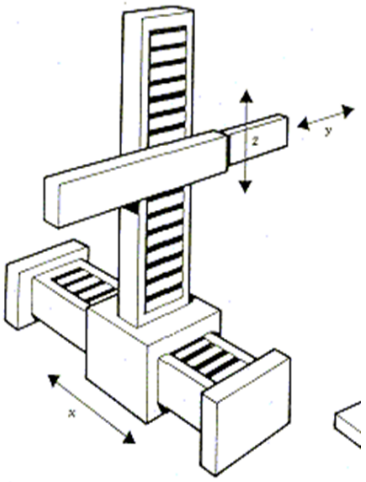
Fundamentación de robótica (Gpo 101) Martes 28 de Febrero del 2023
Antonio Silva Martínez
A01173663
José Jezarel Sánchez Mijares
A01735226

Reto 1 (Robot Cartesiano de 3GDL)

Los robots cartesianos son una categoría de robots industriales que se utilizan para automatizar tareas en la producción y fabricación. Estos robots se llaman cartesianos porque se mueven en tres ejes cartesianos (X, Y y Z), que se corresponden con los tres ejes de coordenadas en un sistema de coordenadas cartesianas. El movimiento en cada eje es lineal y puede ser controlado con precisión.



Los grados de libertad son una medida de la capacidad de un robot para moverse en diferentes direcciones y planos. En el caso de los robots cartesianos, la cantidad de grados de libertad que tienen depende de cuántos ejes pueden moverse. Por ejemplo, un robot cartesiano que solo se mueve en los tres ejes cartesianos tendría tres grados de libertad.

La cantidad de grados de libertad que tiene un robot es importante porque determina la complejidad de las tareas que puede realizar. Cuantos más grados de libertad tenga un robot, más versátil será su capacidad para manipular objetos en diferentes posiciones y ángulos. Por lo tanto, los robots cartesianos con mayor cantidad de grados de libertad son más adecuados para trabajos complejos y precisos en la fabricación y producción.

En este trabajo se desarrolló un código en matlab que nos permite conocer los jacobianos de un robot cartesiano de tres grados de libertad.

Código:

Para poder calcular este código es necesario el poder definir como variables las 3 articulaciones utilizadas en el robot, por ende se tienen que declarar para poder definir su configuración y posteriormente se guardará en un vector de coordenadas.

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

%Declaración de variables simbólicas
syms l1(t) l2(t) l3(t) t

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

```
%Creamos el vector de coordenadas articulares
Q= [l1, l2, l3];
%disp('Coordenadas generalizadas');
%pretty (Q);

%Creamos el vector de velocidades generalizadas
Qp= diff(Q, t);
```

Para poder calcular las matrices de rotación es necesario el visual en un plano tridimensional en donde el desplazamiento se realizará siempre entorno a el eje z, esto se realizará en todas las articulaciones del sistema, a su vez se tendrán que rotar para poder emparejar los ejes para obtener la matriz de rotación.

```
%disp('Velocidades generalizadas');
%pretty (Qp);
%Número de grado de libertad del robot
GDL= size(RP,2);
GDL_str= num2str(GDL);

%Articulación 1 (Articulacion en x con respecto a y)
%Posición de la articulación 1 respecto a 0
P(:,:,1)= [l1; 0; 0];
%Matriz de rotación de la junta 1 respecto a 0.... 90º
R(:,:,1)= [0  0  -1;
           0  1   0;
           1  0   0];

%Articulación 2 (x)
%Posición de la articulación 2 respecto a 1
P(:,:,2)= [0; 0; l2];
%Matriz de rotación de la junta 2 respecto a 1
R(:,:,2)= [1  0   0;
           0  0   1;
           0 -1   0];

%Articulación 3 (y)
%Posición de la articulación 2 respecto a 1
P(:,:,3)= [0; 0; l3];
%Matriz de rotación de la junta 3 respecto a 2
R(:,:,3)= [1 0 0;
           0 1 0;
           0 0 1];
```

Se inicializan y calcula las matrices de transformación homogénea para el sistema, para

posteriormente almacenar los valores en las matrices RO y PO

```
%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);


for i = 1:GDL
      i_str= num2str(i);
   %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);
      PO(:,:,i)= T(1:3,4,i);
      %pretty(RO(:,:,i));
      %pretty(PO(:,:,i));
end
```

Posteriormente se calculan los jacobianos y se crea la matriz por medio de derivadas con respecto a las juntas de nuestro sistema.

```
%Calculamos el jacobiano lineal de forma diferencial
%disp('Jacobiano lineal obtenido de forma diferencial');
%Derivadas parciales de x respecto a th1 y th2
Jv11= functionalDerivative(PO(1,1,GDL), l1);
```

```
Jv12= functionalDerivative(PO(1,1,GDL), l2);
Jv13= functionalDerivative(PO(1,1,GDL), l3);
%Derivadas parciales de y respecto a th1 y th2
Jv21= functionalDerivative(PO(2,1,GDL), l1);
Jv22= functionalDerivative(PO(2,1,GDL), l2);
Jv23= functionalDerivative(PO(2,1,GDL), l3);
%Derivadas parciales de z respecto a th1 y th2
Jv31= functionalDerivative(PO(3,1,GDL), l1);
Jv32= functionalDerivative(PO(3,1,GDL), l2);
Jv33= functionalDerivative(PO(3,1,GDL), l3);

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

En esta parte del código se está calculando el jacobiano lineal y angular analítico para las juntas angulares y prismáticas. En esta ocasión únicamente se utilizarán las prismáticas.
Para las juntas se utiliza la función "cross" para calcular la velocidad lineal y angular en la dirección de la junta., entonces el jacobiano lineal y angular para esa junta se calcula utilizando la posición y orientación de las partes del sistema antes y después de la junta. El resultado final son las columnas correspondientes del jacobiano lineal y angular analítico para la junta específica.

```
%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
      %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));%Matriz de rotación
de 0 con respecto a 0 es la Matriz Identidad, la posición previa tambien
será 0
            Jw_a(:,k)=[0,0,1];%Si no hay matriz de rotación previa se
obtiene la Matriz identidad
      end
      else
%           %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

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

Se visualizan los jacobianos obtenidos, en este caso el Jacobiano angular no se visualizará debido a que el tipo de robot es un robot cartesiano, por ende el único desplazamiento que habrá es dentro de los ejes de una forma lineal

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

Nos basamos en códigos realizados para actividades pasadas y lo adaptamos a nuestro nuevo sistema de tres grados de libertad, donde lo más importante fue la parte de las articulaciones ya que teníamos que entender que había detrás de ellas lo que nos llevaría a poder encontrar la matriz para cada articulación. Para esto realizamos un análisis en cada joint donde básicamente íbamos a comparar que tanto se había movido el eje de joint a joint, ubicamos el eje gracias a el eje z indicará hacia donde se mueve la articulación y el x será perpendicular a este, luego de esto hacemos esto con los siguientes, una vez que tenemos los desplazamientos los sustituimos en la matriz de senos y cosenos del que obtendremos una matriz para cada joint, multiplicaremos estas matrices y nos dará el resultado de la articulación 2, y finalmente para la 3 repetimos el mismo procedimiento