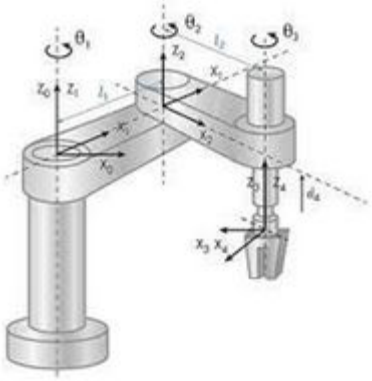
ROBOT ROTACIONAL CON 3 GRADOS DE LIBERTAD



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

Declaramos cada una de las variables involucradas en el código, en este caso tenemos que al tratarse de un robot rotacional, entonces declaramos las variables de los ángulos y las longitudes, así también como un desplazamiento al principio de la cadena cinemática sobre el eje z.

```
%Declaración de variables simbólicas
syms th1(t) a1 t
syms th2(t) a2
syms th3(t) a3
syms l1 %Desplazamiento en z (1)
```

Se realiza la configuración del robot, para este caso al tratarse de articulaciones rotacionales entonces colocamos 3 ceros en el vector de configuración.

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

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

```
Coordenadas generalizadas
```

```
%pretty (Q);
```

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

```
%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        /
```

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

Comenzamos declarando la ubicación de la articulación 1 respecto a la 0, y como podemos observar, ambos sistemas de referencia coincide, por lo cual no necesitamos efectuar ninguna rotación, lo que si es importante declarar es que existe un desplazamiento sobre el eje z, este desplazamiento que anteriormente fue declarado como "l1" lo colocamos dentro del vector de traslación en el apartado para el eje z.

```
%Articulación 1
%Posición de la articulación 1 respecto a 0
P(:,:,1)= [a1*cos(th1);
           a1*sin(th1);
                    l1];

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

Se declara la posición de la articulación 2 respecto a la articulación 1 y de igual forma vemos que las articulaciones al ser de tipo rotacional coincide su sistema de referencia en los tres ejes, por lo tanto no se requiere de alguna rotación u operación adicional. Posterior a eso se declara la matriz de rotación de la junta 2 respecto a la 1.

```
%Articulación 2
%Posición de la articulación 2 respecto a 1
P(:,:,1)= [a2*cos(th2);
           a2*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];
```

Se declara la posición de la articulación 3 respecto a la articulación 1 y de igual forma vemos que las articulaciones al ser de tipo rotacional coincide su sistema de referencia en los tres ejes, por lo tanto no se requiere de alguna rotación u operación adicional. Posterior a eso se declara la matriz de rotación de la junta 3 respecto a la 1.

```
%Articulación 3
%Posición de la articulación 3 respecto a 2
P(:,:,3)= [a3*cos(th3);
           a3*sin(th3);
```

```
                    0]
```

P(:,:,1) =

$$\begin{pmatrix} a_2\cos(\mathrm{th}_2(t)) \\ a_2\sin(\mathrm{th}_2(t)) \\ 0 \end{pmatrix}$$

P(:,:,2) =

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

P(:,:,3) =

$$\begin{pmatrix} a_3\cos(\mathrm{th}_3(t)) \\ a_3\sin(\mathrm{th}_3(t)) \\ 0 \end{pmatrix}$$

```
%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];
```

Se proceden a incializar nuestras matrices de transformación homogénea locales, globales y el marco de referencia inercial, esto con el objetivo de obtener dichas matrices mediante la implementación de un ciclo iterativo, esto nos da como resultado nuestras matrices de transformación local y global para cada una de las joints.

```
%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
```

```
Matriz de Transformación local A1
/ cos(th1(t)), -sin(th1(t)), 0, a2 cos(th2(t)) \
|                                               |
| sin(th1(t)),  cos(th1(t)), 0, a2 sin(th2(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
Matriz de Transformación global T1
/ cos(th1(t)), -sin(th1(t)), 0, a2 cos(th2(t)) \
|                                               |
| sin(th1(t)),  cos(th1(t)), 0, a2 sin(th2(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
Matriz de Transformación local A2
/ cos(th2(t)), -sin(th2(t)), 0, 0 \
|                                 |
| sin(th2(t)),  cos(th2(t)), 0, 0 |
|                                 |
|      0,            0,      1, 0 |
|                                 |
\      0,            0,      0, 1 /
Matriz de Transformación global T2
/ cos(th1(t) + th2(t)), -sin(th1(t) + th2(t)), 0, a2 cos(th2(t)) \
|                                                                 |
| sin(th1(t) + th2(t)),  cos(th1(t) + th2(t)), 0, a2 sin(th2(t)) |
|                                                                 |
|           0,                    0,           1,        0       |
|                                                                 |
\           0,                    0,           0,        1       /
Matriz de Transformación local A3
/ cos(th3(t)), -sin(th3(t)), 0, a3 cos(th3(t)) \
|                                               |
| sin(th3(t)),  cos(th3(t)), 0, a3 sin(th3(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
Matriz de Transformación global T3
/ #2, -#1, 0, a2 cos(th2(t)) + a3 #2 \
|                                     |
| #1,  #2, 0, a2 sin(th2(t)) + a3 #1 |
|                                     |
|  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))
```

A continuación se calcula el jacobiano lineal de forma diferencial, esto se hace mediante las derivadas parciales de "X" respecto a nuestras posiciones, "Y" respecto a nuestras posiciones y "Z" respecto a nuestras posiciones. Luego de estos obtendremos nuestra matriz del Jacobiano lineal que a su vez nos permitirá calcular el jacobiano linear y angular de forma analítica.

```
%Calculamos el jacobiano lineal de forma diferencial
%disp('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 y th2
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 y th2
Jv31= functionalDerivative(PO(3,1,GDL), th1);
Jv32= functionalDerivative(PO(3,1,GDL), th2);
Jv33= functionalDerivative(PO(3,1,GDL), th3);

%Creamos la matríz del Jacobiano lineal
jv_d=simplify([Jv11 Jv12 Jv13;
               Jv21  Jv22 Jv23;
               Jv31  Jv32 Jv33]);
%pretty(jv_d);
%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
            Jw_a(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene la Matriz identic
         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
```

Imprimimos los Jacobianos lineal y angular:

```
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);
```

```
/ - a2 sin(th2(t)) - a3 sin(th1(t) + th2(t) + th3(t)), -a3 sin(th1(t) + th2(t) + th3(t)), -a3 sin(th1(t) + th2(t) +
|
|                 a2 cos(th2(t)) + #1,                                    #1,                                #1
|
\                          0,                                             0,                                 0

where

   #1 == a3 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 /
```

Finalmente obtenemos la velocidad lineal y angular.

Para el vector de velocidad lineal, podemos observar que en el eje "Z" no tenemos ninguna velocidad lineal, esto es poque las joints giran sobre ese mismo eje "Z", y por el contrario si tenemos velocidad lineal en "X" y "Y" por que las joints giran "barriendo" ambos planos, por lo cual la velocidad lineal dependerá de estos ejes.

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

```
Velocidad lineal obtenida mediante el Jacobiano lineal
```

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

```
/ - #4 (a2 sin(th2(t)) + a3 sin(#1)) - a3 #3 sin(#1) - a3 #2 sin(#1) \
|                                                                    |
|  #4 (a2 cos(th2(t)) + a3 cos(#1)) + a3 #3 cos(#1) + a3 #2 cos(#1)  |
|                                                                    |
\                                 0                                  /

where

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

         _________
          d
   #2 == -- th3(t)
         dt
```

```
      ________
       d
#3 == -- th2(t)
      dt

      ________
       d
#4 == -- th1(t)
      dt
```

Para el vector de velocidad angular podemos observar que efectivamente en el eje "Z" es el único eje en donde obtendremos esta velocidad, esto porque todas las joints giran alrededor de dicho eje. Por el contrario no se tienen velocidades angulares en "X" y "Y" debido a que en estos ejes no se produce ningún tipo de giro, por lo cual no puede existir una velocidad angular.

```
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      /
```

Hecho por: Fredy Canseco Santos - A01735589