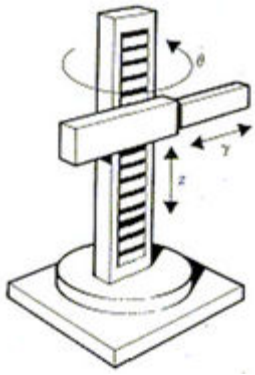
ROBOT CARTESIANO Y ROTACIONAL CON 3 GRADOS DE LIBERTAD



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

Declaramos cada una de las variables involucradas en el código, en este caso tenemos un robot rotacional combinado con un cartesiano, entonces declaramos las variables de las posiciones y las longitudes.

```
%Declaración de variables simbólicas
syms th1(t) l1(t) l2(t) a1 t
```

Se realiza la configuración del robot, para este caso al tratarse de una articulación rotacional y dos primáticas entonces colocamos 2 ceros en el vector de configuración y un uno.

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

%Creamos el vector de coordenadas articulares
Q= [th1, l1, l2];
%disp('Coordenadas generalizadas');
```

```
Coordenadas generalizadas
```

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

```
(th1(t), l1(t), l2(t))
```

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

```
Velocidades generalizadas
```

```
pretty (Qp);
```

```
/  d          d         d       \
| -- th1(t), -- l1(t), -- l2(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.

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

%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 vemos que ambos sistemas de referencia no coinciden, por lo cual necesitamos efectuar una rotación, dicha rotación será de 90 grados negativos sobre el eje "x" para hacer coincidir los ejes. 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(:,:,2)= [0; 0; l1];
%Matriz de rotación de la junta 2 respecto a 1 .... -90°
R(:,:,2)= [1  0  0; %Rotacion de -90 en el eje x
           0  0  1;
           0 -1  0];
```

Se declara la posición de la articulación 3 respecto a la articulación 1.

```
%Articulación 3
%Posición de la articulación 3 respecto a 2
P(:,:,3)= [0; l2; 0];
%Matriz de rotación de la junta 3 respecto a 2
R(:,:,3)= [1  0  0;
           0  1  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, a1 cos(th1(t)) \
|                                               |
| sin(th1(t)),  cos(th1(t)), 0, a1 sin(th1(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
Matriz de Transformación global T1
/ cos(th1(t)), -sin(th1(t)), 0, a1 cos(th1(t)) \
|                                               |
| sin(th1(t)),  cos(th1(t)), 0, a1 sin(th1(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
Matriz de Transformación local A2
/ 1,  0, 0,   0   \
|                 |
| 0,  0, 1,   0   |
|                 |
| 0, -1, 0, l1(t) |
|                 |
\ 0,  0, 0,   1   /
Matriz de Transformación global T2
/ cos(th1(t)),  0, -sin(th1(t)), a1 cos(th1(t)) \
|                                                |
```

```
| sin(th1(t)),  0,  cos(th1(t)), a1 sin(th1(t)) |
|                                               |
|      0,      -1,       0,           l1(t)     |
|                                               |
\      0,       0,       0,             1       /
Matriz de Transformación local A3
/ 1, 0, 0,   0   \
|                |
| 0, 1, 0, l2(t) |
|                |
| 0, 0, 1,   0   |
|                |
\ 0, 0, 0,   1   /
Matriz de Transformación global T3
/ cos(th1(t)),  0, -sin(th1(t)), a1 cos(th1(t)) \
|                                               |
| sin(th1(t)),  0,  cos(th1(t)), a1 sin(th1(t)) |
|                                               |
|      0,      -1,       0,       l1(t) - l2(t) |
|                                               |
\      0,       0,       0,             1       /
```

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 y th2
Jv11= functionalDerivative(PO(1,1,GDL), th1);
Jv12= functionalDerivative(PO(1,1,GDL), l1);
Jv13= functionalDerivative(PO(1,1,GDL), l2);
%Derivadas parciales de y respecto a th1 y th2
Jv21= functionalDerivative(PO(2,1,GDL), th1);
Jv22= functionalDerivative(PO(2,1,GDL), l1);
Jv23= functionalDerivative(PO(2,1,GDL), l2);
%Derivadas parciales de z respecto a th1 y th2
Jv31= functionalDerivative(PO(3,1,GDL), th1);
Jv32= functionalDerivative(PO(3,1,GDL), l1);
Jv33= functionalDerivative(PO(3,1,GDL), l2);

%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

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);
```

```
/ -a1 sin(th1(t)), 0, -sin(th1(t)) \
|                                  |
|  a1 cos(th1(t)), 0,  cos(th1(t)) |
|                                  |
\        0,        1,       0      /
```

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

```
Jacobiano ángular obtenido de forma analítica
```

```
pretty (Jw_a);
```

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

Imprimimos los Jacobianos lineal y angular:

Para el vector de la velocidad lineal podemos inferir que es correcto que la velocidad lineal también pueda estar afectada sobre el eje "Z", esto porque una de las joints prismáticas (l1) corre sobre dicho eje. También es importante decir que la velocidad en "X" y "Y" se ve afectada por la articulación rotacional debido a que por el tipo de movimiento que efectúa, esta también será capaz de causar una influencia en ambos ejes, esto al "barrerlos" cuando gira sobre el eje Z.

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

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

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

```
/              / _________      _________ \ \
|              |  d             d         | |
| -sin(th1(t)) | -- l2(t) + a1 -- th1(t)  | |
|              \ dt            dt        / |
|                                          |
|              / _________      _________ \ |
|              |  d             d         | |
|  cos(th1(t)) | -- l2(t) + a1 -- th1(t)  | |
|              \ dt            dt        / |
|                                          |
|                 _________                |
|                  d                       |
|                 -- l1(t)                 |
\                 dt                      /
```

Para el vector de velocidad angular es correcto decir que la velocidad angular los puede estar afectada en el eje "Z", esto porque la articulación rotacional es la única que se encuentra ubicada sobre dicho eje y generar un movimiento rotacional, cosa que las articulaciones prismáticas no presentan como cualidad. Aclaramos esto diciendo que aunque una de las articulaciones primsáticas se encuentrse sobre dicho eje esta no afectará debido a que solo efectúa un movimiento de traslación.

```
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        |
| -- th1(t) |
\ dt        /
```

Hecho por: Fredy Canseco Santos - A01735589