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

Declaramamos nuestras variables simbolicas que van a ser nuestras articulaciones a1, a2 y a3 con sus respectivos angulos θ_1, θ_2 y θ_3, el tiempo (t) y la constante l3 que es la longitud en la primera articulacion.

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

Declaramos que nuestras 3 articualciones son rotacionales.

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

Derivamos nuestro vector de cordenadas posiciones para obtener el vector de velocidades.

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

Se declara el numero de articulaciones del robot

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

Articulacion 1

Se crea el vector de posiciones el cual se va a ver tanto en X y Y por el angulo θ_1 y en Z por una longitud constante.

En cuanto a la matriz de rotacion solo usamos la matriz de rotacion Z que esta afectada por θ_1

```
%Articulación 1
%Posición de la articulación 3 respecto a 1
P(:,:,1)= [a1*cos(th1); a1*sin(th1);l1];
```

```
%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];
```

Articulacion 2

Se crea el vector de posiciones el cual se va a ver tanto en X y Y en este caso por θ_2

En cuanto a la matriz de rotacion solo usamos la matriz de rotacion Z que esta afectada por θ_2

```
%Articulación 2
%Posición de la articulación 1 respecto a 0
P(:,:,2)= [a2*cos(th2); a2*sin(th2);0];
%Matriz de rotación de la junta 1 respecto a 0.... -90º
R(:,:,2)= [cos(th2) -sin(th2)  0;
           sin(th2)  cos(th2)  0;
           0         0         1];
```

Articulacion 3

Se crea el vector de posiciones el cual se va a ver tanto en X y Y en este caso por θ_3

En cuanto a la matriz de rotacion solo usamos la matriz de rotacion Z que esta afectada por θ_3

```
%Articulación 3
%Posición de la articulación 2 respecto a 1
P(:,:,3)= [a3*cos(th3); a3*sin(th3);0];
%Matriz de rotación de la junta 2 respecto a 1
R(:,:,3)= [cos(th3) -sin(th3)  0;
           sin(th3)  cos(th3)  0;
           0         0         1];




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

Iniciamos la matrices de transformacion homogeneas tanto locales como globlaes

```
%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 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,       l1       |
|                                               |
\      0,            0,      0,        1       /
Matriz de Transformación global T2
/ #2, -#1, 0, a1 cos(th1(t)) + a2 #2 \
|                                    |
| #1,  #2, 0, a1 sin(th1(t)) + a2 #1 |
|                                    |
|  0,   0,  1,           l1           |
|                                    |
\  0,   0,  0,            1           /

where

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

   #2 == cos(th1(t) + th2(t))
Matriz de Transformación global T3
/ #2, -#1, 0, a1 cos(th1(t)) + a3 #2 + a2 cos(th1(t) + th2(t)) \
|                                                              |
| #1,  #2, 0, a1 sin(th1(t)) + a3 #1 + a2 sin(th1(t) + th2(t)) |
|                                                              |
|  0,   0,  1,                        l1                        |
|                                                              |
\  0,   0,  0,                         1                        /

where
```

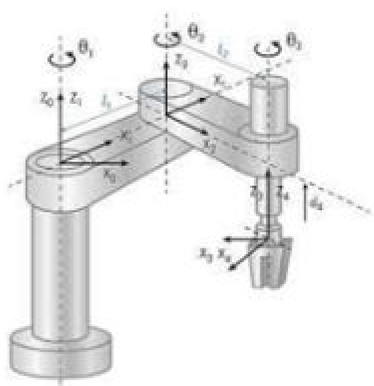
```
#1 == sin(th1(t) + th2(t) + th3(t))

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

Obtenemos el jacobiano haciendo derivadas parciales

```
%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), 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 respect
            Jw_a(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene la Matriz
         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);
```

'Velocidad lineal obtenida mediante el Jacobiano lineal'



Usando el mismo sistema de referencia

En este caso solo podemos que tanto en X como en Y, va a ve movimiento por parte de todas las articiones ya que todas se pueden rotar un cierto angulo, de tan manera que puedan moverse sobre esos ejes.

Mientras que en Z no se puede mover ninguno, ya que en todo momento van a estar perpendiculares a este eje y no podran moverse sobre el .

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

```
/ - #4 (a3 sin(#1) + a2 sin(#2)) - #5 (a1 sin(th1(t)) + a3 sin(#1) + a2 sin(#2)) - a3 #3 sin(#1) \
|                                                                                                |
|  #4 (a3 cos(#1) + a2 cos(#2)) + #5 (a1 cos(th1(t)) + a3 cos(#1) + a2 cos(#2)) + a3 #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
```

Velocidad angular obtenida mediante el Jacobiano angular

Nuevamente, usando el mismo sistema de referecia, podemo ver que el todas la articiones se mueven sobre el eje Z y tiene sentido ya que es le eje sobre el cual estan rotando todas.

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

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