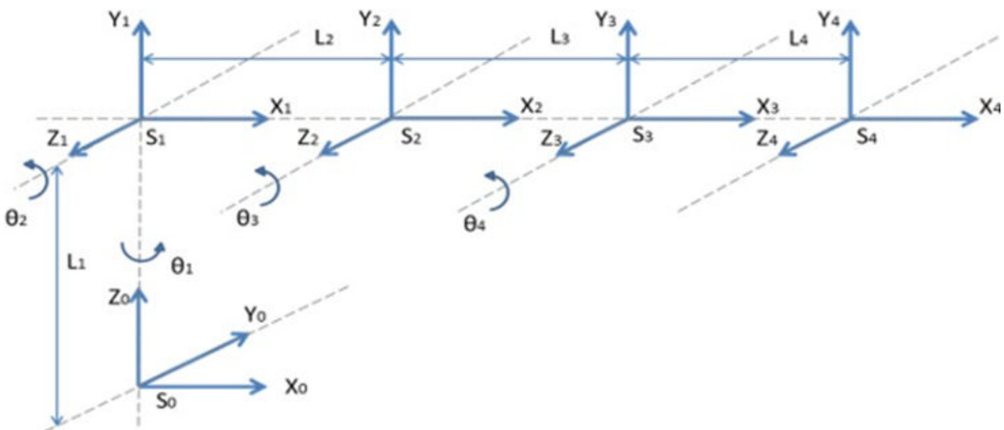
Implementación de robótica inteligente (TE3002B)

Presentación Final (Cinemática Diferencial de Piernas)

Pamela Hernández Montero A01736368

Obtener la matriz de transformación homogénea global T, empleando variables simbólicas de los siguientes sistemas la cual relacione la posición y orientación del extremo del robot respecto a su sistema de referencia fijo (la base). Simulando cada una de las transformaciones desde la trama absoluta hasta la trama final.



Para la primera figura realicé las transformaciones homogéneas que consisten en:

1. Rotación en el eje X de 90 grados y una traslación de 0.5 unidades en el eje Z.
2. Traslación de 0.5 unidades en el eje X.
3. Traslación de 0.5 unidades en el eje X.
4. Traslación de 0.5 unidades en el eje X.

Posteriormente, para todas las figuras, determiné la matriz de transformación homogénea global a partir de las matrices de transformación individuales. Esto se logra multiplicando las matrices de transformación homogénea sucesivas para obtener la posición y orientación final de la figura respecto al sistema de coordenadas global.

Detalles Adicionales:

- *Matrices de Transformación Homogénea*

Las matrices de transformación homogénea se utilizan para describir la posición y orientación de un sistema de coordenadas respecto a otro. Cada transformación homogénea incluye una rotación y una traslación, que se representa como una matriz 4×4. Para obtener la matriz de transformación global *T* se requiere de multiplicar todas las matrices individuales

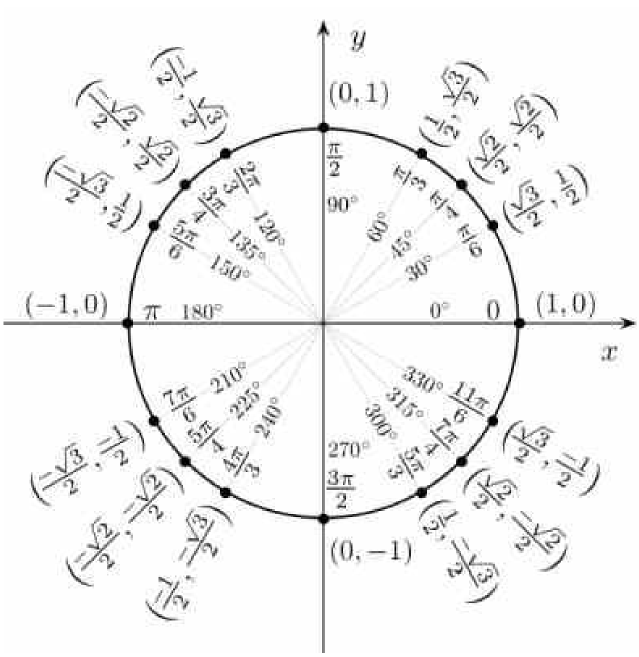
- *Representación y Gráficas*

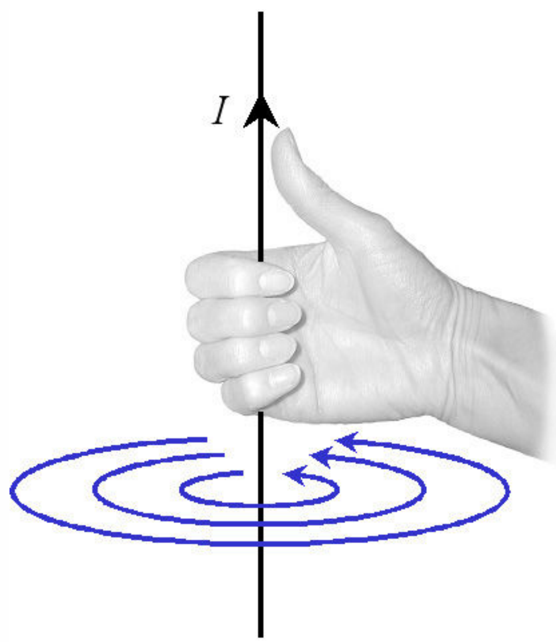
Después de obtener la matriz de transformación homogénea global para cada figura, grafiqué una estructura que representa la longitud recorrida por la matriz de transformación para observar el movimiento completo del objeto a través del espacio tridimensional. Utilicé la función `tranimate` de MATLAB para animar estas transformaciones y visualizar el movimiento de manera dinámica.

- *Uso del Círculo Unitario y la Regla de la Mano Derecha*

Es importante destacar que se hace uso del círculo unitario para definir las proyecciones de las posiciones y rotaciones. El círculo unitario permite representar cualquier ángulo en términos de sus proyecciones en los ejes �*x* e �*y*, utilizando las funciones seno y coseno. Esto es crucial para calcular las matrices de rotación.

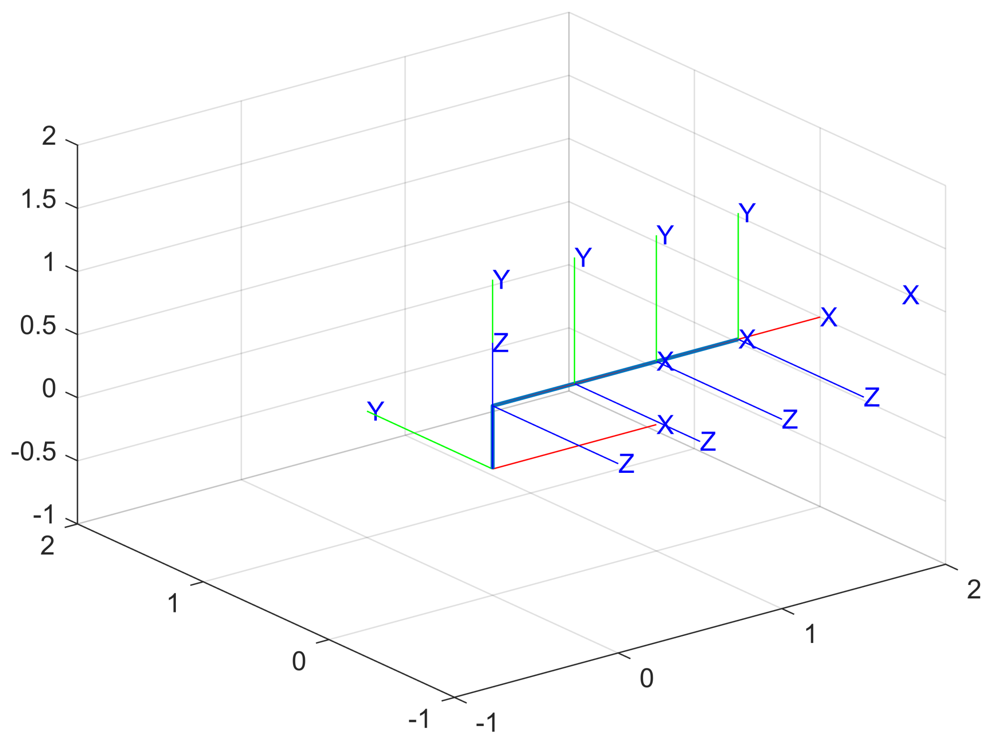
Además, se aplica la regla de la mano derecha para determinar la dirección de rotación. Según esta regla, si los dedos de la mano derecha siguen la dirección de la rotación, el pulgar indica la dirección del eje de rotación. Este concepto es esencial para asegurar que las rotaciones se realicen en la dirección correcta.



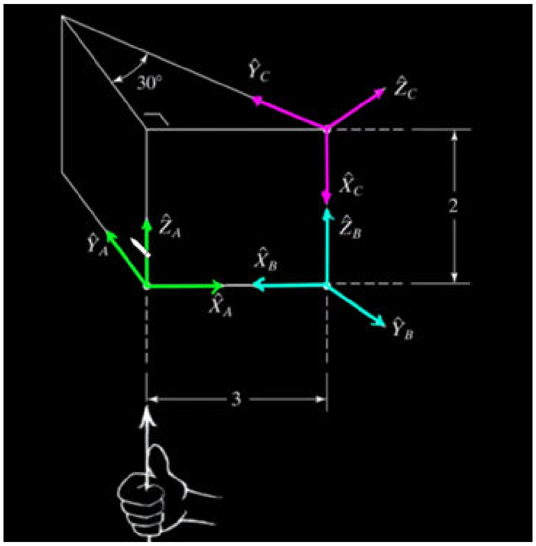



```
% Limpieza de pantalla
clear all
close all
clc

% Calculamos las matrices de transformación homogénea
H0 = SE3;
H1 = SE3(rotx(pi/2), [0 0 0.5]);
H2 = SE3(rotz(0), [0.5 0 0]);
H3 = SE3(rotz(0), [0.5 0 0]);
H4 = SE3(rotz(0), [0.5 0 0]);

H10 = H0 * H1;
```

```
H20 = H10 * H2;
H30 = H20 * H3;
H40 = H30 * H4; % Matriz de transformación homogénea global de 4 a 0

% Coordenadas de la estructura de translación y rotación
x = [0 0 0.5 1 1.5 1.5];
y = [0 0 0 0 0 0];
z = [0 0.5 0.5 0.5 0.5 0.5];

plot3(x, y, z, 'LineWidth', 1.5); axis([-1 2 -1 2 -1 2]); grid on;
hold on;

% Graficamos la trama absoluta o global
trplot(H0, 'rgb', 'axis', [-1 2 -1 2 -1 2]);

% Realizamos una animación para las tramas
tranimate(H0, H10, 'rgb', 'axis', [-1 4 -1 6 -1 2]);
tranimate(H10, H20, 'rgb', 'axis', [-1 4 -1 6 -1 2]);
tranimate(H20, H30, 'rgb', 'axis', [-1 4 -1 6 -1 2]);
tranimate(H30, H40, 'rgb', 'axis', [-1 4 -1 6 -1 2]);disp("Matriz de transformación
homogénea");disp(H40);
```




```
Matriz de transformación homogénea
    1         0         0       1.5
    0         0        -1         0
    0         1         0       0.5
    0         0         0         1
```

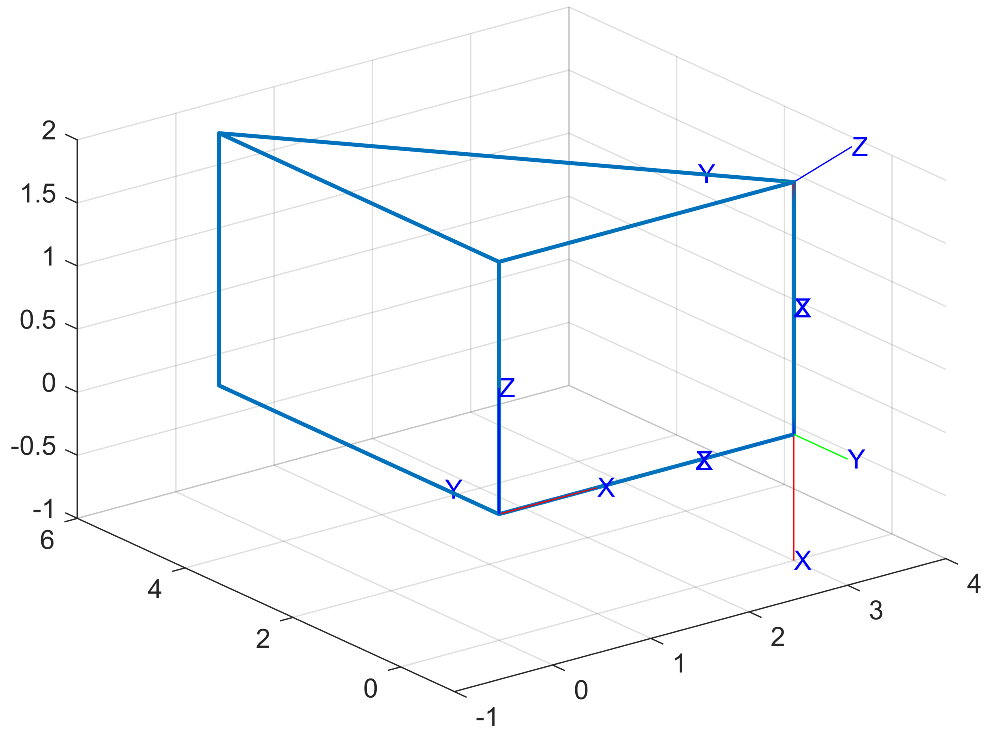


Para la segunda simulación realicé las transformaciones homogéneas que consisten en:

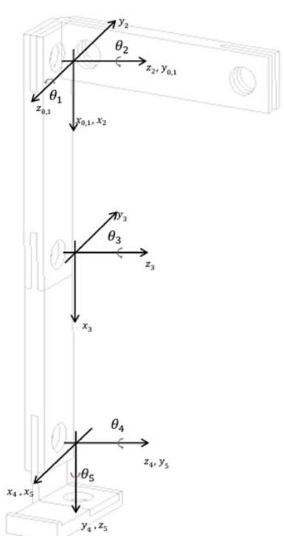
1. Rotación en el eje Z de 180 grados y traslación de 3 unidades en el eje X.
2. Rotación en el eje Y de 90 grados.
3. Rotación en el eje X de 150 grados y traslación de -2 unidades en el eje X.

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

%Calculamos las matrices de transformación homogénea
H0=SE3;
H1=SE3(rotz(pi), [3 0 0]);
H2=SE3(roty(pi/2), [0 0 0]);
H3=SE3(rotx(150*pi/180), [-2 0 0]); %150 grados

H20= H1*H2;
H30= H20*H3; %Matriz de transformación homogenea global de 3 a 0

%Coordenadas de la estructura de translación y rotación
x=[0 3 3 0 0 0     0     0 0     3];
y=[0 0 0 0 0 5.196 5.196 0 5.196 0];
z=[0 0 2 2 0 0     2     2 2     2];

plot3(x, y, z,'LineWidth', 1.5); axis([-1 4 -1 6 -1 2]); grid on;
hold on;

%Graficamos la trama absoluta o global
trplot(H0,'rgb','axis', [-1 4 -1 6 -1 2])

% Realizamos una animación para las tramas
tranimate(H0, H1,'rgb','axis', [-1 4 -1 6 -1 2])
tranimate(H1, H20,'rgb','axis', [-1 4 -1 6 -1 2])
```

```
tranimate(H20, H30,'rgb','axis', [-1 4 -1 6 -1 2]);disp("Matriz de transformación homogénea"); disp(H30)
```



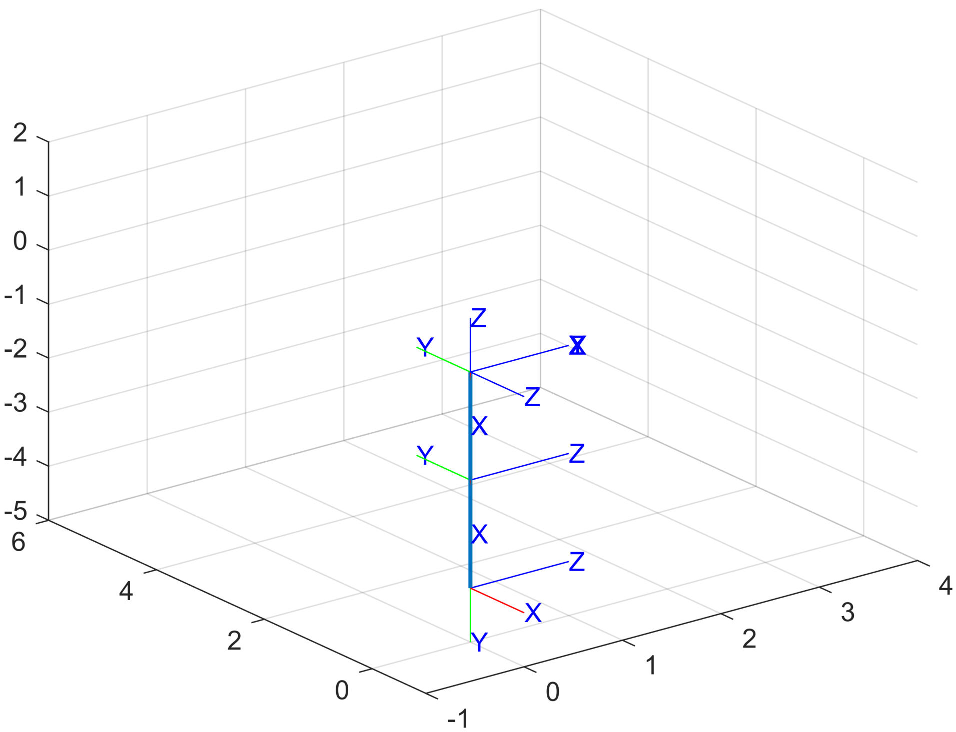
```
Matriz de transformación homogénea
     0      -0.5     0.866         3
     0     0.866       0.5         0
    -1         0         0         2
     0         0         0         1
```



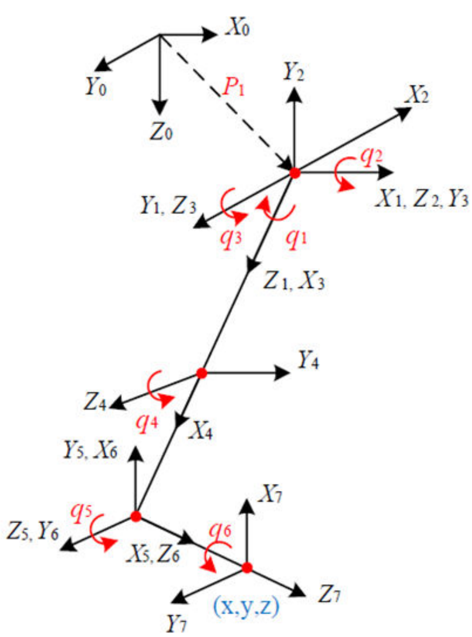
Para la tercera simulación realicé las transformaciones homogéneas que consisten en:

1. Rotación en el eje Z de -90 grados y rotación en el eje Y de 90 grados.
2. Rotación en el eje X de -90 grados.
3. Traslación de 2 unidades en el eje X.
4. Rotación en el eje Z de 270 grados y traslación de 2 unidades en el eje X.

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

%Calculamos las matrices de transformación homogénea
H0=SE3;
H1=SE3(rotz(-pi/2)*roty(pi/2), [0 0 0]);
H2=SE3(rotx(-pi/2),[0 0 0]);
H3=SE3([2 0 0]);
H4=SE3(rotz(3*pi/2), [2 0 0]);

H20= H1*H2;
H30= H20*H3;
H40= H30*H4;%Matriz de transformación homogenea global de 4 a 0

%Coordenadas de la estructura de translación y rotación
x=[0 0 0];
y=[0 0 0];
z=[0 0 -4];

plot3(x, y, z,'LineWidth', 1.5); axis([-1 4 -1 6 -5 2]); grid on;
hold on;

%Graficamos la trama absoluta o global
trplot(H0,'rgb','axis', [-1 4 -1 6 -1 2])

% Realizamos una animación para las tramas
tranimate(H0, H1,'rgb','axis', [-1 4 -1 6 -1 2])
tranimate(H1, H20,'rgb','axis', [-1 4 -1 6 -1 2])
tranimate(H20, H30,'rgb','axis', [-1 4 -1 6 -1 2]);
tranimate(H30, H40,'rgb','axis', [-1 4 -1 6 -1 2]);disp("Matriz de transformación
homogénea"); disp(H40)
```




```
Matriz de transformación homogénea
     0          0          1          0
    -1          0          0          0
     0         -1          0         -4
     0          0          0          1
```

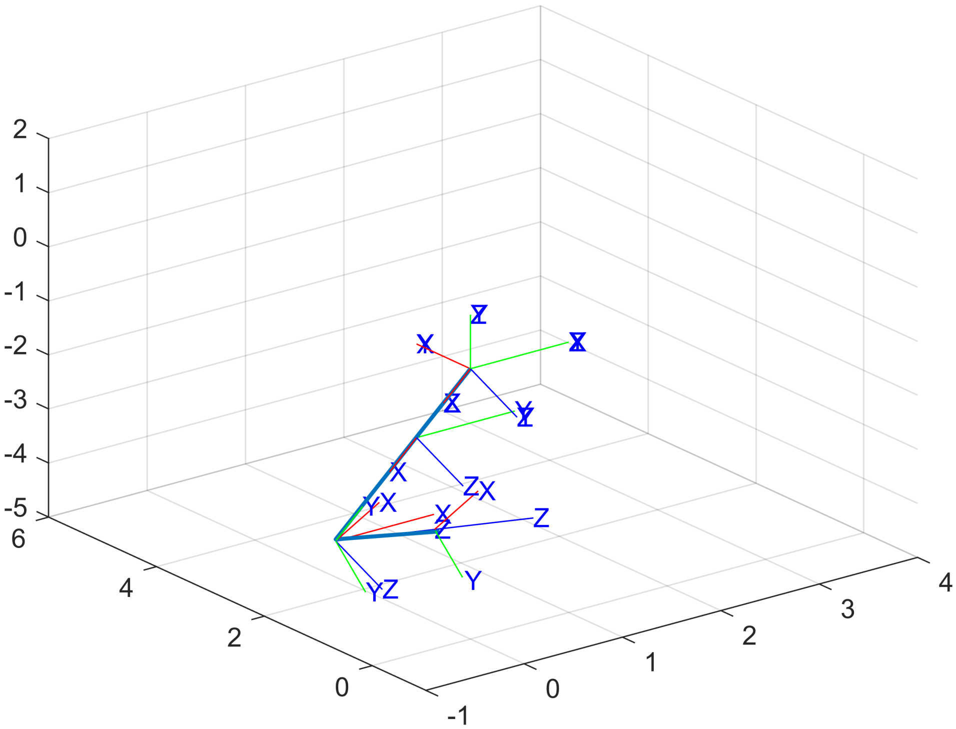


Para la ultima simulacion realicé las transformaciones homogéneas que consisten en:

1. Rotación en el eje X de 210 grados (7*pi/6 radianes).
2. Rotación en el eje X de 60 grados (pi/3 radianes), rotación en el eje Y de 90 grados (pi/2 radianes), y rotación en el eje Z de 180 grados (pi radianes).
3. Rotación en el eje Z de -60 grados (-pi/3 radianes) y rotación en el eje X de 90 grados (pi/2 radianes).
4. Traslación de 2 unidades en el eje X.
5. Rotación en el eje Z de 90 grados (pi/2 radianes) y traslación de 3 unidades en el eje X.
6. Rotación en el eje Z de 80 grados (80*pi/180 radianes) y rotación en el eje X de 80 grados (80*pi/180 radianes).
7. Traslación de 1 unidad en el eje Z.

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

%Calculamos las matrices de transformación homogénea
H0=SE3;
H1=SE3(rotx(7*pi/6),[0 0 0]);
H2=SE3(rotx(pi/3)*roty(pi/2)*rotz(pi),[0 0 0]);
H3=SE3(rotz(-pi/3)*rotx(pi/2),[0 0 0]);
H4=SE3([2 0 0]);
H5=SE3(rotz(pi/2),[3 0 0]);
H6=SE3(rotz(80*pi/180)*rotx(80*pi/180),[0 0 0]);
H7=SE3([0 0 1]);

H20= H1*H2;
H30= H20*H3;
H40= H30*H4;
H50= H40*H5;
H60= H50*H6;
H70= H60*H7;%Matriz de transformación homogenea global de 7 a 0

%Coordenadas de la estructura de translación y rotación
x=[0 0 1];
y=[0 2.5 2.4];
z=[0 -4.3 -4.6];

plot3(x, y, z,'LineWidth', 1.5); axis([-1 4 -1 6 -5 2]); grid on;
hold on;

%Graficamos la trama absoluta o global
trplot(H0,'rgb','axis', [-1 4 -1 6 -1 2])

% Realizamos una animación para las tramas
tranimate(H0, H1,'rgb','axis', [-1 4 -1 6 -1 2])
tranimate(H1, H20,'rgb','axis', [-1 4 -1 6 -1 2])
tranimate(H20, H30,'rgb','axis', [-1 4 -1 6 -1 2]);
tranimate(H30, H40,'rgb','axis', [-1 4 -1 6 -1 2]);
```

```
tranimate(H40, H50,'rgb','axis', [-1 4 -1 6 -1 2]);
tranimate(H50, H60,'rgb','axis', [-1 4 -1 6 -1 2]);
tranimate(H60, H70,'rgb','axis', [-1 4 -1 6 -1 2]);disp("Matriz de transformación homogénea"); disp(H70)
```




```
Matriz de transformación homogénea
     0.1736    -0.171    0.9698    0.9698
    -0.4924   -0.8679  -0.06488     2.435
     0.8529   -0.4663   -0.2349    -4.565
          0         0         0         1
```

Desarrollar el modelo de cinemática diferencial simbólica para el ultimo de los sistemas descritos anteriormente y obtener los vectores de la velocidad angular y velocidad lineal aplicando variables simbólicas para su análisis en cada caso.

Para desarrollar el modelo de cinemática diferencial simbólica primero se declaran las variables simbólicas que representan los ángulos de rotación (�) y longitudes (�*l*). Estas variables son funciones del tiempo *t*.

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

%Declaración de variables simbólicas (no tienen valor especifico)
syms thX_1(t) thY_1(t) thZ_1(t) l1 %Junta esferica
syms thZ_2(t) l2 %Junta rotacional
syms thZ_3(t) thY_3(t) l3 %Junta de 2 grados, Z y Y
syms t
```

Posteriormente se define un vector que indica el tipo de junta para cada articulación. En este caso, todas las juntas tienen un movimiento de rotacion

```
RP=[0 0 0]; %configuracion del robot, 0 para junta rotacional y 1 para prismatica
```

Se define el vector de coordenadas articulares y generalizadas Q y se muestra en la consola, del mismo modo se calcula el vector de velocidades articulares Qp derivando con respecto al tiempo. Finalmente se obtiene el número de grados de libertad del robot a partir del tamaño del vector RP.

```
% Vector de coordenadas articulares y generalizada
Q = [thX_1, thY_3, thZ_2];
disp('Coordenadas articulares y generalizadas:');pretty(Q)
```

```
Coordenadas articulares y generalizadas:
(thX_1(t), thY_3(t), thZ_2(t))
```

```
%vector de velocidad articulares+
Qp = diff(Q,t); %se utiliza diff para derivadas cuya variable no depende de otras
disp('Velocidades articulares'); pretty(Qp);
```

```
Velocidades articulares
/  d            d            d          \
| -- thX_1(t), -- thY_3(t), -- thZ_2(t) |
\ dt           dt           dt          /
```

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

Para la primer articulacion las matrices $AZ1$, $AY1$, y $AX1$ representan las rotaciones alrededor de los ejes Z, Y, y X respectivamente. La matriz de rotación total R se obtiene multiplicando estas matrices, debido a que representan los grados en los que la articulacion puede moverse.

```
%Articulacion 1
P(:,:,1) = [l1*cos(thX_1);  l1*sin(thY_1);  0];

% Matriz de rotación
AZ1 = [cos(thZ_1) -sin(thZ_1) 0;
       sin(thZ_1)  cos(thZ_1) 0;
           0          0       1];

AY1 = [cos(thY_1)   0   sin(thY_1);
           0        1        0    ;
       -sin(thY_1)  0  cos(thY_1)];

AX1 = [ 1    0         0    ;
        0 cos(thX_1) -sin(thX_1);
        0 sin(thX_1)  cos(thX_1)];

R(:,:,1) = AZ1 .* AY1 .* AX1
```

```
R =
```

$$\begin{pmatrix} \cos(\mathrm{thY}_1(t))\cos(\mathrm{thZ}_1(t)) & 0 & 0 \\ 0 & \cos(\mathrm{thX}_1(t))\cos(\mathrm{thZ}_1(t)) & 0 \\ 0 & 0 & \cos(\mathrm{thX}_1(t))\cos(\mathrm{thY}_1(t)) \end{pmatrix}$$

En la articulación 2 se define la matriz de rotación *R* para la segunda articulación, que es una rotación alrededor del eje *Z*.

```
%Articulacion 2
P(:,:,2) = [l2*cos(thZ_2);  l2*sin(thZ_2); 0];

% Matriz de rotación
R(:,:,2) = [cos(thZ_2) -sin(thZ_2) 0;
            sin(thZ_2)  cos(thZ_2) 0;
                0          0       1]
```

```
R(:,:,1) =
```

$$\begin{pmatrix} \cos(\mathrm{thY}_1(t))\cos(\mathrm{thZ}_1(t)) & 0 & 0 \\ 0 & \cos(\mathrm{thX}_1(t))\cos(\mathrm{thZ}_1(t)) & 0 \\ 0 & 0 & \cos(\mathrm{thX}_1(t))\cos(\mathrm{thY}_1(t)) \end{pmatrix}$$

```
R(:,:,2) =
```

$$\begin{pmatrix} \cos(\mathrm{thZ}_2(t)) & -\sin(\mathrm{thZ}_2(t)) & 0 \\ \sin(\mathrm{thZ}_2(t)) & \cos(\mathrm{thZ}_2(t)) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Para la articulación 3 las matrices *AZ*3 y *AY*3 representan las rotaciones alrededor de los ejes *Z* y *Y* respectivamente. La matriz de rotación total *R* se obtiene multiplicando estas matrices.

```
%Articulacion 3
P(:,:,3) = [l3*cos(thY_3);  l3*sin(thZ_3); 0];

% Matriz de rotación
AZ3 = [cos(thZ_3) -sin(thZ_3) 0;
       sin(thZ_3)  cos(thZ_3) 0;
           0          0       1];

AY3 = [cos(thY_3)   0   sin(thY_3);
           0        1       0    ;
       -sin(thY_3)  0  cos(thY_3)];

R(:,:,3) = AZ3 .* AY3
```

```
R(:,:,1) =
```

$$\begin{pmatrix} \cos(\mathrm{thY}_1(t))\cos(\mathrm{thZ}_1(t)) & 0 & 0 \\ 0 & \cos(\mathrm{thX}_1(t))\cos(\mathrm{thZ}_1(t)) & 0 \\ 0 & 0 & \cos(\mathrm{thX}_1(t))\cos(\mathrm{thY}_1(t)) \end{pmatrix}$$

```
R(:,:,2) =
```

$$\begin{pmatrix} \cos(\mathrm{thZ}_2(t)) & -\sin(\mathrm{thZ}_2(t)) & 0 \\ \sin(\mathrm{thZ}_2(t)) & \cos(\mathrm{thZ}_2(t)) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
R(:,:,3) =
```

$$\begin{pmatrix} \cos(\mathrm{thY}_3(t))\cos(\mathrm{thZ}_3(t)) & 0 & 0 \\ 0 & \cos(\mathrm{thZ}_3(t)) & 0 \\ 0 & 0 & \cos(\mathrm{thY}_3(t)) \end{pmatrix}$$

Se crea un vector de ceros para usar en las matrices de transformación homogénea.

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

Se inicializan las matrices de transformación homogénea locales �*A* y globales �*T*, así como las posiciones y matrices de rotación en el marco de referencia inercial.

```
%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);
```

En la siguiente función, se calculan las matrices de transformación homogénea global *T* para cada articulación multiplicando las matrices de transformación locales *A*.

```
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]);

   %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);
end
```

```
Matriz de Transformación global T1
```

```
/ cos(thY_1(t)) cos(thZ_1(t)),              0,                            0,              l1 cos(thX_1(t)) \
|                                                                                                           |
|              0,              cos(thX_1(t)) cos(thZ_1(t)),                0,              l1 sin(thY_1(t)) |
|                                                                                                           |
|              0,                            0,              cos(thX_1(t)) cos(thY_1(t)),         0         |
|                                                                                                           |
\              0,                            0,                            0,                    1         /
Matriz de Transformación global T2
/ cos(thY_1(t)) cos(thZ_1(t)) cos(thZ_2(t)), -cos(thY_1(t)) cos(thZ_1(t)) sin(thZ_2(t)),            0,
|
| cos(thX_1(t)) cos(thZ_1(t)) sin(thZ_2(t)),  cos(thX_1(t)) cos(thZ_1(t)) cos(thZ_2(t)),            0,
|
|                     0,                                           0,                     cos(thX_1(t)) cos(thY_1(t)
|
\                     0,                                           0,                                   0,
Matriz de Transformación global T3
/ cos(thY_1(t)) cos(thY_3(t)) cos(thZ_1(t)) cos(thZ_2(t)) cos(thZ_3(t)), -cos(thY_1(t)) cos(thZ_1(t)) cos(thZ_3(t))
|
| cos(thX_1(t)) cos(thY_3(t)) cos(thZ_1(t)) cos(thZ_3(t)) sin(thZ_2(t)),  cos(thX_1(t)) cos(thZ_1(t)) cos(thZ_2(t))
|
|                                   0,                                                                0,
|
\                                   0,                                                                0,
```

Se inicializan los Jacobianos lineal *Jv* y angular *Jw* con las posiciones de la última articulación.

```
%Calculamos el jacobiano lineal de forma analítica
Jv_a(:,GDL)=PO(:,:,GDL);
Jw_a(:,GDL)=PO(:,:,GDL);
```

Para cada articulación se determina si es rotacional (0) o prismática (1). Para las articulaciones rotacionales, se calcula el Jacobiano lineal utilizando el producto cruzado y el Jacobiano angular como la tercera columna de la matriz de rotación *RO*. Para las articulaciones prismáticas, el Jacobiano lineal es la tercera columna de *RO* y el Jacobiano angular es un vector de ceros.

```
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
```

Finalmente se despliegan los resultados simplificados de la forma analitica y lineal.

```
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
/ - l1 sin(thY_1(t)) - l2 cos(thX_1(t)) cos(thZ_1(t)) sin(thZ_2(t)) - l3 cos(thX_1(t)) cos(thY_3(t)) cos(thZ_1(t)) s
|
|  l1 cos(thX_1(t)) + l2 cos(thY_1(t)) cos(thZ_1(t)) cos(thZ_2(t)) - l3 cos(thY_1(t)) cos(thZ_1(t)) sin(thZ_2(t)) s
|
\                                                                                                         0,

where

                         2
   #1 == cos(thY_1(t))

                         2
   #2 == cos(thX_1(t))
```

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

```
Jacobiano ángular obtenido de forma analítica
/ 0,              0,                             0              \
|                                                               |
| 0,              0,                             0              |
|                                                               |
\ 1, cos(thX_1(t)) cos(thY_1(t)), cos(thX_1(t)) cos(thY_1(t)) /
```

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

```
Velocidad lineal obtenida mediante el Jacobiano lineal
/ - #5 (l1 sin(thY_1(t)) + l2 cos(thX_1(t)) cos(thZ_1(t)) sin(thZ_2(t)) + l3 cos(thX_1(t)) cos(thY_3(t)) cos(thZ_1(t
|
|  #5 (l1 cos(thX_1(t)) + l2 cos(thY_1(t)) cos(thZ_1(t)) cos(thZ_2(t)) - l3 cos(thY_1(t)) cos(thZ_1(t)) sin(thZ_2(t
|
\

where

                         2
   #1 == cos(thY_1(t))

                         2
   #2 == cos(thX_1(t))

           ___________
            d
   #3 == -- thZ_2(t)
           dt

           ___________
            d
   #4 == -- thY_3(t)
           dt
```

```
          ___________
           d
   #5 == -- thX_1(t)
          dt
```

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

```
Velocidad angular obtenida mediante el Jacobiano angular
/                                                 0                                                \
|                                                                                                  |
|                                                 0                                                |
|                                                                                                  |
| ___________    ___________                                 ___________                           |
|  d              d                                           d                                    |
| -- thX_1(t) + -- thY_3(t) cos(thX_1(t)) cos(thY_1(t)) + -- thZ_2(t) cos(thX_1(t)) cos(thY_1(t)) |
\ dt             dt                                          dt                                    /
```