Actividad 6 | Robot prismático 3 GDL

Oscar Ortiz Torres A01769292

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

tic

% Declaración de variables simbólicas
syms l1(t) l2(t) l3(t) t   % Angulos de cada articulación
syms m1 m2 m3 Ixx1 Iyy1 Izz1 Ixx2 Iyy2 Izz2 Ixx3 Iyy3 Izz3 % Masas y matrices de
Inercia
syms lc1 lc2 lc3 % l=longitud de eslabones y lc=distancia al centro de masa de cada
eslabón
syms pi g a cero

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

% Creamos el vector de velocidades articulares
Qp= diff(Q, t); %[th1p; th2p];
%disp('Velocidades generalizadas');
%pretty (Qp);
Qp = Qp(t);

% Creamos el vector de aceleraciones articulares
Qpp= diff(Qp, t); %[th1pp; th2pp];
%disp('Aceleraciones generalizadas');
%pretty (Qpp);

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

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

Articulación 1

```
% Posición de la articulación 1 respecto a 0
P(:,:,1)= [0;
           0;
           l1];

% Matriz de rotación de la junta 1 respecto a 0....
```

```
R(:,:,1)= rotZ(0)*rotX(-90);
```

Articulación 2

```
% Posición de la articulación 2 respecto a 1
P(:,:,2)= [0;
           0;
           l2];

% Matriz de rotación de la junta 1 respecto a 0
R(:,:,2)= rotZ(0)*rotY(90);
```

Articulación 3

```
% Posición de la articulación 3 respecto a 2
P(:,:,3)= [0;
           0;
           l3];

% Matriz de rotación de la junta 3 respecto a 2
R(:,:,3)= rotZ(0);
```

```
% 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
```

CALCULAMOS LAS VELOCIDADES PARA CADA ESLABÓN

VELOCIDADES PARA ESLABÓN 3

```
% Calculamos el jacobiano lineal de forma analítica
Jv_a3(:,GDL)=PO(:,:,GDL);
Jw_a3(:,GDL)=PO(:,:,GDL);

for k= 1:GDL
    if RP(k)==0
        % Para las juntas de revolución
        try
            Jv_a3(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL)-PO(:,:,k-1));
            Jw_a3(:,k)= RO(:,3,k-1);
        catch
            Jv_a3(:,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_a3(:,k)=[0,0,1]; % Si no hay matriz de rotación previa se obtiene la
Matriz identidad
         end
     else
        % Para las juntas prismáticas
        try
            Jv_a3(:,k)= RO(:,3,k-1);
        catch
            Jv_a3(:,k)=[0,0,1];
        end
            Jw_a3(:,k)=[0,0,0];
     end
 end

% Obtenemos SubMatrices de Jacobianos
Jv_a3= simplify (Jv_a3);
Jw_a3= simplify (Jw_a3);
%disp('Jacobiano lineal obtenido de forma analítica');
%pretty (Jv_a3);
%disp('Jacobiano ángular obtenido de forma analítica');
```

```
%pretty (Jw_a3);

% Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac3= [Jv_a3;
      Jw_a3];
Jacobiano3= simplify(Jac3);
% pretty(Jacobiano3);

% Obtenemos vectores de Velocidades Lineales y Angulares para el eslabón 3
disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 3'); V3 =
simplify(Jv_a3*Qp)
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 3
V3 =
```

$$\begin{pmatrix} \frac{\partial}{\partial t}\, l_3(t) \\ \frac{\partial}{\partial t}\, l_2(t) \\ \frac{\partial}{\partial t}\, l_1(t) \end{pmatrix}$$

```
disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 3'); W3
= simplify(Jw_a3*Qp)
```

```
Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 3
W3 =
```

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

VELOCIDADES PARA ESLABÓN 2

```
% Calculamos el jacobiano lineal y angular de forma analítica
Jv_a2(:,GDL-1)=PO(:,:,GDL-1);
Jw_a2(:,GDL-1)=PO(:,:,GDL-1);

for k= 1:GDL-1
    if RP(k)==0
        % Para las juntas de revolución
        try
            Jv_a2(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-1)-PO(:,:,k-1));
            Jw_a2(:,k)= RO(:,3,k-1);
        catch
            Jv_a2(:,k)= cross([0,0,1], PO(:,:,GDL-1));% Matriz de rotación de 0 con
respecto a 0 es la Matriz Identidad, la posición previa tambien será 0
            Jw_a2(:,k)=[0,0,1]; % Si no hay matriz de rotación previa se obtiene la
Matriz identidad
         end
     else
```

```
        % Para las juntas prismáticas
        try
            Jv_a2(:,k)= RO(:,3,k-1);
        catch
            Jv_a2(:,k)=[0,0,1];
        end
            Jw_a2(:,k)=[0,0,0];
     end
 end

% Obtenemos SubMatrices de Jacobianos
Jv_a2= simplify (Jv_a2);
Jw_a2= simplify (Jw_a2);
%disp('Jacobiano lineal obtenido de forma analítica');
%pretty (Jv_a2);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a2);

% Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac2 = [Jv_a2;
        Jw_a2];
Jacobiano2= simplify(Jac2);
% pretty(Jacobiano2);

% Obtenemos vectores de Velocidades Lineales y Angulares para el eslabón 2
disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 2'); V2 =
simplify(Jv_a2*Qp(1:2))
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 2
V2 =
```

$$\begin{pmatrix} 0 \\ \frac{\partial}{\partial t}\, l_2(t) \\ \frac{\partial}{\partial t}\, l_1(t) \end{pmatrix}$$

```
disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 2'); W2
= simplify(Jw_a2*Qp(1:2))
```

```
Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 2
W2 =
```

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

VELOCIDADES PARA ESLABÓN 1

```
% Calculamos el jacobiano lineal y angular de forma analítica
Jv_a1(:,GDL-2)=PO(:,:,GDL-2);
```

```
Jw_a1(:,GDL-2)=PO(:,:,GDL-2);

for k= 1:GDL-2
    if RP(k)==0
        % Para las juntas de revolución
        try
            Jv_a1(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-2)-PO(:,:,k-1));
            Jw_a1(:,k)= RO(:,3,k-1);
        catch
            Jv_a1(:,k)= cross([0,0,1], PO(:,:,GDL-2));% Matriz de rotación de 0 con
respecto a 0 es la Matriz Identidad, la posición previa tambien será 0
            Jw_a1(:,k)=[0,0,1]; % Si no hay matriz de rotación previa se obtiene la
Matriz identidad
         end
     else
        % Para las juntas prismáticas
        try
            Jv_a1(:,k)= RO(:,3,k-1);
        catch
            Jv_a1(:,k)=[0,0,1];
        end
            Jw_a1(:,k)=[0,0,0];
     end
 end

% Obtenemos SubMatrices de Jacobianos
Jv_a1= simplify (Jv_a1);
Jw_a1= simplify (Jw_a1);
%disp('Jacobiano lineal obtenido de forma analítica');
%pretty (Jv_a1);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a1);

% Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac1 = [Jv_a1;
        Jw_a1];
Jacobiano1 = simplify(Jac1);
% pretty(Jacobiano);

% Obtenemos vectores de Velocidades Lineales y Angulares para el eslabón 2
disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 1'); V1 =
simplify(Jv_a1*Qp(1))
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 1
V1 =
```

$$\begin{pmatrix} 0 \\ 0 \\ \frac{\partial}{\partial t}\, l_1(t) \end{pmatrix}$$

```
disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 1'); W1
= simplify(Jw_a1*Qp(1))
```

```
Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 1
W1 =
```

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

Energía Cinética

```
% Omitimos la división de cada lc

% Distancia del origen del eslabón a su centro de masa

% Vectores de posición respecto al centro de masa
P01=subs(P(:,:,1), l1, lc1);    % La función subs sustituye l1 por lc1 en
P12=subs(P(:,:,2), l2, lc2);    % la expresión P(:,:,1)/2
P23=subs(P(:,:,3), l3, lc3);    % la expresión P(:,:,1)/2

% Creamos matrices de inercia para cada eslabón

I1=[Ixx1 0 0;
    0 Iyy1 0;
    0 0 Izz1];

I2=[Ixx2 0 0;
    0 Iyy2 0;
    0 0 Izz2];

I3=[Ixx3 0 0;
    0 Iyy3 0;
    0 0 Izz3];
% Función de energía cinética

% Extraemos las velocidades lineales del efector final en cada eje
%V2=V2(t);
Vx= V3(1,1);
Vy= V3(2,1);
Vz= V3(3,1);

% Extraemos las velocidades angular del efector final en cada ángulo de Euler
%W2=W2(t);
W_pitch= W3(1,1);
W_roll= W3(2,1);
W_yaw= W3(3,1);

% Calculamos la energía cinética para cada uno de los eslabones%%%%%%%%%
```

```
% Eslabón 1
V1_Total = V1+cross(W1,P01);
K1= (1/2*m1*(V1_Total))'*((V1_Total)) + (1/2*W1)'*(I1*W1);
disp('Energía Cinética en el Eslabón 1'); K1= simplify (K1) %pretty (K1);
```

```
Energía Cinética en el Eslabón 1
K1 =
```

$$\frac{\left|\frac{\partial}{\partial t}\, l_1(t)\right|^2 \overline{m_1}}{2}$$

```
% Eslabón 2
V2_Total = V2+cross(W2,P12);
K2 = (1/2*m2*(V2_Total))'*((V2_Total)) + (1/2*W2)'*(I2*W2);
disp('Energía Cinética en el Eslabón 2'); K2 = simplify (K2)
```

```
Energía Cinética en el Eslabón 2
K2 =
```

$$\frac{\overline{m_2}\left(\left|\frac{\partial}{\partial t}\, l_1(t)\right|^2 + \left|\frac{\partial}{\partial t}\, l_2(t)\right|^2\right)}{2}$$

```
% Eslabón 3
V3_Total = V3+cross(W3,P23);
K3 = (1/2*m3*(V3_Total))'*((V3_Total)) + (1/2*W3)'*(I3*W3);
disp('Energía Cinética en el Eslabón 3'); K3 = simplify(K3)
```

```
Energía Cinética en el Eslabón 3
K3 =
```

$$\frac{\overline{m_3}\left(\left|\frac{\partial}{\partial t}\, l_1(t)\right|^2 + \left|\frac{\partial}{\partial t}\, l_2(t)\right|^2 + \left|\frac{\partial}{\partial t}\, l_3(t)\right|^2\right)}{2}$$

```
disp('Energía Cinética Total'); K_Total = simplify (K1+K2+K3) %pretty(K_Total);
```

```
Energía Cinética Total
K_Total =
```

$$\frac{\overline{m_2}\left(\left|\frac{\partial}{\partial t}\, l_1(t)\right|^2 + \left|\frac{\partial}{\partial t}\, l_2(t)\right|^2\right)}{2} + \frac{\left|\frac{\partial}{\partial t}\, l_1(t)\right|^2 \overline{m_1}}{2} + \frac{\overline{m_3}\left(\left|\frac{\partial}{\partial t}\, l_1(t)\right|^2 + \left|\frac{\partial}{\partial t}\, l_2(t)\right|^2 + \left|\frac{\partial}{\partial t}\, l_3(t)\right|^2\right)}{2}$$

Energía Potencial

```
% Obte+nmos las alturas respecto a la gravedad
h1= P01(2); %Tomo la altura paralela al eje y
h2= P12(3); %Tomo la altura paralela al eje z
h3= P23(1); %Tomo la altura paralela al eje x

U1=m1*g*h1
```

U1 = 0

```
U2=m2*g*h2
```

U2 = $g\,\mathrm{lc}_2\,m_2$

```
U3=m3*g*h3
```

U3 = 0

```
% Calculamos la energía potencial total
U_Total= U1 + U2 +U3
```

U_Total = $g\,\mathrm{lc}_2\,m_2$

```
% Obtenemos el Lagrangiano
Lagrangiano= simplify (K_Total-U_Total) %pretty (Lagrangiano);
```

Lagrangiano =

$$\frac{\overline{m_2}\left(\left|\frac{\partial}{\partial t}\,l_1(t)\right|^2+\left|\frac{\partial}{\partial t}\,l_2(t)\right|^2\right)}{2}+\frac{\left|\frac{\partial}{\partial t}\,l_1(t)\right|^2\overline{m_1}}{2}+\frac{\overline{m_3}\left(\left|\frac{\partial}{\partial t}\,l_1(t)\right|^2+\left|\frac{\partial}{\partial t}\,l_2(t)\right|^2+\left|\frac{\partial}{\partial t}\,l_3(t)\right|^2\right)}{2}-g\,\mathrm{lc}_2\,m_2$$

```
% Modelo de Energía
H= simplify (K_Total+U_Total) %pretty (H)
```

H =

$$\frac{\overline{m_2}\left(\left|\frac{\partial}{\partial t}\,l_1(t)\right|^2+\left|\frac{\partial}{\partial t}\,l_2(t)\right|^2\right)}{2}+\frac{\left|\frac{\partial}{\partial t}\,l_1(t)\right|^2\overline{m_1}}{2}+\frac{\overline{m_3}\left(\left|\frac{\partial}{\partial t}\,l_1(t)\right|^2+\left|\frac{\partial}{\partial t}\,l_2(t)\right|^2+\left|\frac{\partial}{\partial t}\,l_3(t)\right|^2\right)}{2}+g\,\mathrm{lc}_2\,m_2$$

```
function r_x = rotX(th)
    r_x = [1 0 0; 0 cosd(th) -sind(th); 0 sind(th) cosd(th)];
end

function r_y = rotY(th)
    r_y = [cosd(th) 0 sind(th); 0 1 0; -sind(th) 0 cosd(th)];
end

function r_z = rotZ(th)
    r_z = [cosd(th) -sind(th) 0; sind(th) cosd(th) 0; 0 0 1];
end
```