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

tic
% Declaración de variables simbólicas
syms th1(t) th2(t) d3(t) th4(t) th5(t) th6(t) t  % Coordenadas generalizadas
syms th1p(t) th2p(t) d3p(t) th4p(t) th5p(t) th6p(t)  % Velocidades generalizadas
syms th1pp(t) th2pp(t) d3pp(t) th4pp(t) th5pp(t) th6pp(t) % Aceleraciones
generalizadas
syms m1 m2 m3 m4 m5 m6 Ixx1 Iyy1 Izz1 Ixx2 Iyy2 Izz2 Ixx3 Iyy3 Izz3 Ixx4 Iyy4 Izz4
Ixx5 Iyy5 Izz5 Ixx6 Iyy6 Izz6
syms l1 l2 l3 l4 l5 l6 lc1 lc2 lc3 lc4 lc5 lc6
syms pi g a cero

% Vector de coordenadas articulares
Q = [th1; th2; d3; th4; th5; th6];

% Vector de velocidades articulares
Qp = [th1p; th2p; d3p; th4p; th5p; th6p];

% Vector de aceleraciones articulares
Qpp = [th1pp; th2pp; d3pp; th4pp; th5pp; th6pp];

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

% Número de grados de libertad (GDL)
GDL = size(RP, 2);

% ------------------- ARTICULACIONES -------------------

% Articulación 1 (Rotación en Z)
P(:,:,1) = [0; 0; l1];
R(:,:,1) = [cos(th1) -sin(th1)  0;
            sin(th1)  cos(th1)  0;
            0         0         1];

% Articulación 2 (Rotación en Z, pero con -90° en X respecto a la anterior)
P(:,:,2) = [l2*cos(th2); l2*sin(th2); 0];
R(:,:,2) = [cos(th2) -sin(th2)  0;
            sin(th2)  cos(th2)  0;
            0         0         1] * ...
           [1  0   0;
            0  0  -1;
            0  1   0]; % Rotación de -90° en X

% Articulación 3 (Prismática en Z, con +90° en X respecto a la anterior)
P(:,:,3) = [0; 0; d3];
```

```matlab
R(:,:,3) = [1  0   0;
            0  0   1;
            0 -1   0]; % Rotación de +90° en X

% Articulación 4 (Rotación en Z, mantiene marco)
P(:,:,4) = [0; 0; l4];
R(:,:,4) = [cos(th4) -sin(th4)  0;
            sin(th4)  cos(th4)  0;
            0         0         1];

% Articulación 5 (Rotación en Z, pero con -90° en X respecto a la anterior)
P(:,:,5) = [0; 0; l5];
R(:,:,5) = [cos(th5) -sin(th5)  0;
            sin(th5)  cos(th5)  0;
            0         0         1] * ...
           [1  0   0;
            0  0  -1;
            0  1   0]; % Rotación de -90° en X

% Articulación 6 (Rotación en Z, con +90° en X respecto a la anterior)
P(:,:,6) = [0; 0; 0];
R(:,:,6) = [cos(th6) -sin(th6)  0;
            sin(th6)  cos(th6)  0;
            0         0         1] * ...
           [1  0   0;
            0  0   1;
            0 -1   0]; % Rotación de +90° en X


%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 6  %%%%%%%%%%

%Calculamos el jacobiano lineal de forma analítica
Jv_a6(:,GDL)=PO(:,:,GDL);
Jw_a6(:,GDL)=PO(:,:,GDL);

for k= 1:GDL
    if RP(k)==0
       %Para las juntas de revolución
        try
            Jv_a6(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL)-PO(:,:,k-1));
            Jw_a6(:,k)= RO(:,3,k-1);
        catch
            Jv_a6(:,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_a6(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene la
Matriz identidad
         end
     else
%         %Para las juntas prismáticas
        try
            Jv_a6(:,k)= RO(:,3,k-1);
        catch
            Jv_a6(:,k)=[0,0,1];
        end
            Jw_a6(:,k)=[0,0,0];
     end
 end

%Obtenemos SubMatrices de Jacobianos
Jv_a6= simplify (Jv_a6);
Jw_a6= simplify (Jw_a6);
%disp('Jacobiano lineal obtenido de forma analítica');
```

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

%Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac6= [Jv_a6;
      Jw_a6];
Jacobiano6= simplify(Jac6);
% pretty(Jacobiano);
Qp=Qp(t)
```

Qp =

$$\begin{pmatrix} \mathrm{th1p}(t) \\ \mathrm{th2p}(t) \\ \mathrm{d3p}(t) \\ \mathrm{th4p}(t) \\ \mathrm{th5p}(t) \\ \mathrm{th6p}(t) \end{pmatrix}$$

```
%Obtenemos vectores de Velocidades Lineales y Angulares para el eslabón 6
 disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 6');
```

Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 6

```
V6=simplify (Jv_a6*Qp);
pretty(V6)
```

```
/ sin(th1(t) + th2(t)) d3p(t) - th1p(t) #1 - th2p(t) #1 \
|                                                        |
| th1p(t) #2 - cos(th1(t) + th2(t)) d3p(t) + th2p(t) #2 |
|                                                        |
\                           0                            /

where

   #1 == l2 sin(th1(t) + th2(t)) - cos(th1(t) + th2(t)) d3(t)

   #2 == l2 cos(th1(t) + th2(t)) + sin(th1(t) + th2(t)) d3(t)
```

```
 disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 6');
```

Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 6

```
W6=simplify (Jw_a6*Qp);
pretty(W6)
```

```
/  sin(th1(t) + th2(t) + th4(t) + th5(t)) th6p(t) \
|                                                  |
| -cos(th1(t) + th2(t) + th4(t) + th5(t)) th6p(t) |
|                                                  |
\      th1p(t) + th2p(t) + th4p(t) + th5p(t)      /
```

```
%%%%%%%%%%  VELOCIDADES PARA ESLABÓN 5  %%%%%%%%%%

%Calculamos el jacobiano lineal y angular de forma analítica
Jv_a5(:,GDL-1)=PO(:,:,GDL-1);
Jw_a5(:,GDL-1)=PO(:,:,GDL-1);

for k= 1:GDL-1
    if RP(k)==0
       %Para las juntas de revolución
        try
            Jv_a5(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-1)-PO(:,:,k-1));
            Jw_a5(:,k)= RO(:,3,k-1);
        catch
            Jv_a5(:,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_a5(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene la
Matriz identidad
         end
     else
%         %Para las juntas prismáticas
        try
            Jv_a5(:,k)= RO(:,3,k-1);
        catch
            Jv_a5(:,k)=[0,0,1];
        end
            Jw_a5(:,k)=[0,0,0];
     end
 end

%Obtenemos SubMatrices de Jacobianos
Jv_a5= simplify (Jv_a5);
Jw_a5= simplify (Jw_a5);
%disp('Jacobiano lineal obtenido de forma analítica');
%pretty (Jv_a);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a);

%Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac5= [Jv_a5;
      Jw_a5];
Jacobiano5= simplify(Jac5);
% pretty(Jacobiano);

%Obtenemos vectores de Velocidades Lineales y Angulares para el eslabón 5
disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 5');
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 5
```

```
V5=simplify (Jv_a5*Qp(1:5));
```

```
pretty(V5)
```

```
/ sin(th1(t) + th2(t)) d3p(t) - th1p(t) #1 - th2p(t) #1 \
|                                                        |
| th1p(t) #2 - cos(th1(t) + th2(t)) d3p(t) + th2p(t) #2 |
|                                                        |
\                           0                            /

where

   #1 == l2 sin(th1(t) + th2(t)) - cos(th1(t) + th2(t)) d3(t)

   #2 == l2 cos(th1(t) + th2(t)) + sin(th1(t) + th2(t)) d3(t)
```

```
 disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 5');
```

```
Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 5
```

```
W5=simplify (Jw_a5*Qp(1:5));
pretty(W5)
```

```
/                  0                  \
|                                      |
|                  0                   |
|                                      |
\ th1p(t) + th2p(t) + th4p(t) + th5p(t) /
```

```




%%%%%%%%%%  VELOCIDADES PARA ESLABÓN 4  %%%%%%%%%%%%%%%%%%%%%%%%%%

%Calculamos el jacobiano lineal y angular de forma analítica
Jv_a4(:,GDL-2)=PO(:,:,GDL-2);
Jw_a4(:,GDL-2)=PO(:,:,GDL-2);

for k= 1:GDL-2
    if RP(k)==0
       %Para las juntas de revolución
        try
            Jv_a4(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-2)-PO(:,:,k-1));
            Jw_a4(:,k)= RO(:,3,k-1);
        catch
            Jv_a4(:,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_a4(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene la
Matriz identidad
         end
     else
%         %Para las juntas prismáticas
        try
            Jv_a4(:,k)= RO(:,3,k-1);
        catch
            Jv_a4(:,k)=[0,0,1];
        end
```

```
            Jw_a4(:,k)=[0,0,0];
     end
 end

%Obtenemos SubMatrices de Jacobianos
Jv_a4= simplify (Jv_a4);
Jw_a4= simplify (Jw_a4);
%disp('Jacobiano lineal obtenido de forma analítica');
%pretty (Jv_a);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a);

%Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac4= [Jv_a4;
      Jw_a4];
Jacobiano4= simplify(Jac4);
% pretty(Jacobiano);

%Obtenemos vectores de Velocidades Lineales y Angulares para el eslabón 4
 disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 4');
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 4
```

```
V4=simplify (Jv_a4*Qp(1:4));
pretty(V4)
```

```
/ sin(th1(t) + th2(t)) d3p(t) - th1p(t) #1 - th2p(t) #1 \
|                                                        |
| th1p(t) #2 - cos(th1(t) + th2(t)) d3p(t) + th2p(t) #2 |
|                                                        |
\                           0                            /

where

   #1 == l2 sin(th1(t) + th2(t)) - cos(th1(t) + th2(t)) d3(t)

   #2 == l2 cos(th1(t) + th2(t)) + sin(th1(t) + th2(t)) d3(t)
```

```
 disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 4');
```

```
Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 4
```

```
W4=simplify (Jw_a4*Qp(1:4));
pretty(W4)
```

```
/              0             \
|                            |
|              0             |
|                            |
\ th1p(t) + th2p(t) + th4p(t) /
```

```
%%%%%%%%%%  VELOCIDADES PARA ESLABÓN 3  %%%%%%%%%%
```

```
%Calculamos el jacobiano lineal de forma analítica
Jv_a3(:,GDL-3)=PO(:,:,GDL-3);
Jw_a3(:,GDL-3)=PO(:,:,GDL-3);

for k= 1:GDL-3
    if RP(k)==0
       %Para las juntas de revolución
        try
            Jv_a3(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-3)-PO(:,:,k-1));
            Jw_a3(:,k)= RO(:,3,k-1);
        catch
            Jv_a3(:,k)= cross([0,0,1], PO(:,:,GDL-3));%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_a);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a);

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

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

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

```
V3=simplify (Jv_a3*Qp(1:3));
pretty(V3)
```

```
/ sin(th1(t) + th2(t)) d3p(t) - th1p(t) #1 - th2p(t) #1 \
|                                                      |
| th1p(t) #2 - cos(th1(t) + th2(t)) d3p(t) + th2p(t) #2 |
|                                                      |
\                          0                           /

where

   #1 == l2 sin(th1(t) + th2(t)) - cos(th1(t) + th2(t)) d3(t)

   #2 == l2 cos(th1(t) + th2(t)) + sin(th1(t) + th2(t)) d3(t)
```

```
disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 3');
```

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

```
W3=simplify (Jw_a3*Qp(1:3));
pretty(W3)
```

```
/         0         \
|                   |
|         0         |
|                   |
\ th1p(t) + th2p(t) /
```

```
%%%%%%%%%%  VELOCIDADES PARA ESLABÓN 2  %%%%%%%%%%

%Calculamos el jacobiano lineal y angular de forma analítica
Jv_a2(:,GDL-4)=PO(:,:,GDL-4);
Jw_a2(:,GDL-4)=PO(:,:,GDL-4);

for k= 1:GDL-4
    if RP(k)==0
       %Para las juntas de revolución
        try
            Jv_a2(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-4)-PO(:,:,k-1));
            Jw_a2(:,k)= RO(:,3,k-1);
        catch
            Jv_a2(:,k)= cross([0,0,1], PO(:,:,GDL-4));%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_a);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a);

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

%Obtenemos vectores de Velocidades Lineales y Angulares para el eslabón 2
disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 2');
```

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

```
V2=simplify (Jv_a2*Qp(1:2));
pretty(V2)
```

```
/ -l2 sin(th1(t) + th2(t)) (th1p(t) + th2p(t)) \
|                                              |
|  l2 cos(th1(t) + th2(t)) (th1p(t) + th2p(t)) |
|                                              |
\                       0                      /
```

```
 disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 2');
```

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

```
W2=simplify (Jw_a2*Qp(1:2));
pretty(W2)
```

```
/         0         \
|                   |
|         0         |
|                   |
\ th1p(t) + th2p(t) /
```

```




%%%%%%%%%%  VELOCIDADES PARA ESLABÓN 1  %%%%%%%%%%%%%%%%%%%%%%%%%%

%Calculamos el jacobiano lineal y angular de forma analítica
Jv_a1(:,GDL-5)=PO(:,:,GDL-5);
Jw_a1(:,GDL-5)=PO(:,:,GDL-5);
```

```
for k= 1:GDL-5
    if RP(k)==0
       %Para las juntas de revolución
        try
            Jv_a1(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-5)-PO(:,:,k-1));
            Jw_a1(:,k)= RO(:,3,k-1);
        catch
            Jv_a1(:,k)= cross([0,0,1], PO(:,:,GDL-5));%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_a);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a);

%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 1
 disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón 1');
```

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

```
V1=simplify (Jv_a1*Qp(1:1));
pretty(V1)
```

```
/ 0 \
|   |
| 0 |
|   |
\ 0 /
```

```
 disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón 1');
```

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

```
W1=simplify (Jw_a1*Qp(1:1));
pretty(W1)
```

```
/    0    \
|         |
|    0    |
|         |
\ th1p(t) /
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%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);
 P34=subs(P(:,:,4), l4, lc4);
 P45=subs(P(:,:,5), l5, lc5);
 P56=subs(P(:,:,6), l6, lc6);

%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];

I4=[Ixx4 0 0;
    0 Iyy4 0;
    0 0 Izz4];

I5=[Ixx5 0 0;
    0 Iyy5 0;
    0 0 Izz5];

I6=[Ixx6 0 0;
    0 Iyy6 0;
    0 0 Izz6];
%Función de energía cinética
```

```
% 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');
```

```
Energía Cinética en el Eslabón 1
```

```
K1 = simplify(K1);
pretty(K1);
```

```
                2
Izz1 |th1p(t)|
---------------
       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');
```

```
Energía Cinética en el Eslabón 2
```

```
K2 = simplify(K2);
pretty(K2);
```

```
                                            __                               2            ______          2
     /     #7           #6    \             m2 #1 #5 (lc2 sin(#3) |l2|  + l2 sin(th2(t)) |lc2| ) (lc2 sin(th2(t)) + l2 sin(
Izz2 | --------- + --------- | #1 + -------------------------------------------------------------------------------------
     \ 2 th1p(t)   2 th2p(t) /                                                    #2

where

   #1 == th1p(t) + th2p(t)

   #2 == 2 l2 lc2 th1p(t) th2p(t)

         _______   _______
   #3 == th1(t) + th2(t)

   #4 == th1(t) + th2(t)

   #5 == th2p(t) #7 + th1p(t) #6

                    2
   #6 == |th2p(t)|

                    2
   #7 == |th1p(t)|
```

```
%Eslabón 3
V3_Total = V3 + cross(W3, P23);
K3 = (1/2 * m3 * (V3_Total))' * (V3_Total) + (1/2 * W3)' * (I3 * W3);  % Cambié I2
a I3 ya que es el eslabón 3
```

```
disp('Energía Cinética en el Eslabón 3');
```

```
Energía Cinética en el Eslabón 3
```

```
K3 = simplify(K3);
pretty(K3);
```

```
                            / _______    _______ \   __  _______      _______      ______
                            | th1p(t)    th2p(t) |   m3 (th1p(t) #2 + th2p(t) #2 - d3p(t) cos(#5)) (th1p(t) #4 - cos(#6
Izz3 (th1p(t) + th2p(t)) | ------- + ------- | + ---------------------------------------------------------------------
                            \    2          2    /                                               2

where

                    __   _____
   #1 == sin(#5) l2 - d3(t) cos(#5)

         _____                  __
   #2 == d3(t) sin(#5) + cos(#5) l2

   #3 == l2 sin(#6) - cos(#6) d3(t)

   #4 == l2 cos(#6) + sin(#6) d3(t)

         ______   ______
   #5 == th1(t) + th2(t)

   #6 == th1(t) + th2(t)
```

```
%Eslabón 4
V4_Total = V4 + cross(W4, P34);  % Añadí las variables correspondientes para el
eslabón 4
K4 = (1/2 * m4 * (V4_Total))' * (V4_Total) + (1/2 * W4)' * (I4 * W4);  % Añadí las
matrices de inercia y velocidades para el eslabón 4
disp('Energía Cinética en el Eslabón 4');
```

```
Energía Cinética en el Eslabón 4
```

```
K4 = simplify(K4);
pretty(K4);
```

```
                                      / _______    _______    _______ \   __  _______      _______      ______
                                      | th1p(t)    th2p(t)    th4p(t) |   m4 (th1p(t) #2 + th2p(t) #2 - d3p(t) cos(#5)) (
Izz4 (th1p(t) + th2p(t) + th4p(t)) | ------- + ------- + ------- | + ---------------------------------------------------
                                      \    2          2          2    /                                                 2

where

                    __   _____
   #1 == sin(#5) l2 - d3(t) cos(#5)

         _____                  __
   #2 == d3(t) sin(#5) + cos(#5) l2

   #3 == l2 sin(#6) - cos(#6) d3(t)

   #4 == l2 cos(#6) + sin(#6) d3(t)

         ______   ______
   #5 == th1(t) + th2(t)
```

```
    #6 == th1(t) + th2(t)
```

```
%Eslabón 5
V5_Total = V5 + cross(W5, P45);  % Añadí las variables correspondientes para el
eslabón 5
K5 = (1/2 * m5 * (V5_Total))' * (V5_Total) + (1/2 * W5)' * (I5 * W5);  % Añadí las
matrices de inercia y velocidades para el eslabón 5
disp('Energía Cinética en el Eslabón 5');
```

```
Energía Cinética en el Eslabón 5
```

```
K5 = simplify(K5);
pretty(K5);
```

```
__   _______    _______         ______
m5 (th1p(t) #2 + th2p(t) #2 - d3p(t) cos(#5)) (th1p(t) #4 - cos(#6) d3p(t) + th2p(t) #4)
------------------------------------------------------------------------------------- + Izz5 (th1p(t) + th2p(t) -
                                           2

where

                      __   _____
    #1 == sin(#5) l2 - d3(t) cos(#5)

           _____                 __
    #2 == d3(t) sin(#5) + cos(#5) l2

    #3 == l2 sin(#6) - cos(#6) d3(t)

    #4 == l2 cos(#6) + sin(#6) d3(t)

           ______    ______
    #5 == th1(t) + th2(t)

    #6 == th1(t) + th2(t)
```

```
%Eslabón 6
V6_Total = V6 + cross(W6, P56);  % Añadí las variables correspondientes para el
eslabón 6
K6 = (1/2 * m6 * (V6_Total))' * (V6_Total) + (1/2 * W6)' * (I6 * W6);  % Añadí las
matrices de inercia y velocidades para el eslabón 6
disp('Energía Cinética en el Eslabón 6');
```

```
Energía Cinética en el Eslabón 6
```

```
K6 = simplify(K6);
pretty(K6);
```

```
__   _______    _______         ______
m6 (th1p(t) #4 + th2p(t) #4 - d3p(t) cos(#7)) (th1p(t) #6 - cos(#8) d3p(t) + th2p(t) #6)
------------------------------------------------------------------------------------- + Izz6 (th1p(t) + th2p(t) -
                                           2

where
```

```
  #1 == ______ + ______ + ______ + ______
        th1(t)   th2(t)   th4(t)   th5(t)

  #2 == th1(t) + th2(t) + th4(t) + th5(t)

                   __     _____
  #3 == sin(#7) l2 - d3(t) cos(#7)

        _____                    __
  #4 == d3(t) sin(#7) + cos(#7) l2

  #5 == l2 sin(#8) - cos(#8) d3(t)

  #6 == l2 cos(#8) + sin(#8) d3(t)

        ______   ______
  #7 == th1(t) + th2(t)

  #8 == th1(t) + th2(t)
```

```
% Calculamos la energía cinética total considerando los 6 eslabones
K_Total = simplify(K1 + K2 + K3 + K4 + K5 + K6);
disp('Energía Cinética Total');
```

```
Energía Cinética Total
```

```
pretty(K_Total);
```

```
                                                                                                __           __           __
Izz1 #19                                                                                        m3 #7 #9     m4 #7 #9     m5 #7
-------- + Izz4 (th1p(t) + th2p(t) + th4p(t)) (#17 + #16 + #18) + Izz3 #2 (#17 + #16) + -------- + -------- + -----
    2                                                                                       2            2            2

where

        ______   ______   ______   ______
  #1 == th1(t) + th2(t) + th4(t) + th5(t)

  #2 == th1p(t) + th2p(t)

  #3 == th1p(t) + th2p(t) + th4p(t) + th5p(t)

  #4 == th1(t) + th2(t) + th4(t) + th5(t)

  #5 == 2 l2 lc2 th1p(t) th2p(t)

        _______         ______             _______
  #6 == th1p(t) #12 - d3p(t) sin(#21) + th2p(t) #12

        _______         _______         ______
  #7 == th1p(t) #13 + th2p(t) #13 - d3p(t) cos(#21)

  #8 == th1p(t) #14 - sin(#22) d3p(t) + th2p(t) #14

  #9 == th1p(t) #15 - cos(#22) d3p(t) + th2p(t) #15

                              _______
                              th5p(t)
  #10 == #17 + #16 + #18 + -------
                                 2
```

```
  #11 == th2p(t) #19 + th1p(t) #20

                    __   _____
  #12 == sin(#21) l2 - d3(t) cos(#21)

         _____                    __
  #13 == d3(t) sin(#21) + cos(#21) l2

  #14 == l2 sin(#22) - cos(#22) d3(t)

  #15 == l2 cos(#22) + sin(#22) d3(t)

         _______
         th2p(t)
  #16 == -------
            2

         _______
         th1p(t)
  #17 == -------
            2

         _______
         th4p(t)
  #18 == -------
            2

                  2
  #19 == |th1p(t)|

                  2
  #20 == |th2p(t)|

         ______   ______
  #21 == th1(t) + th2(t)

  #22 == th1(t) + th2(t)
```

```
%Energía Potencial  p = mgh

% Obtenemos las alturas respecto a la gravedad
h1 = P01(2); % Tomo la altura paralela al eje z
h2 = P12(2); % Tomo la altura paralela al eje y
h3 = P23(2); % Tomo la altura paralela al eje y
h4 = P34(2); % Tomo la altura paralela al eje y (eslabón 4)
h5 = P45(2); % Tomo la altura paralela al eje y (eslabón 5)
h6 = P56(2); % Tomo la altura paralela al eje y (eslabón 6)

% Energías potenciales para cada eslabón
U1 = m1 * g * h1;
U2 = m2 * g * h2;
U3 = m3 * g * h3;
U4 = m4 * g * h4;  % Energía potencial para el eslabón 4
U5 = m5 * g * h5;  % Energía potencial para el eslabón 5
U6 = m6 * g * h6;  % Energía potencial para el eslabón 6

% Calculamos la energía potencial total
```

```
U_Total = U1 + U2 + U3 + U4 + U5 + U6;

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

```
                                                                                                   __                __                __
Izz1 #19                                                                                           m3 #7 #9          m4 #7 #9          m5 #7
-------- + Izz4 (th1p(t) + th2p(t) + th4p(t)) (#17 + #16 + #18) + Izz3 #2 (#17 + #16) + -------- + -------- + ------
    2                                                                                      2                 2                 2

where

          _______   _______   _______   _______
   #1 == th1(t) + th2(t) + th4(t) + th5(t)

   #2 == th1p(t) + th2p(t)

   #3 == th1p(t) + th2p(t) + th4p(t) + th5p(t)

   #4 == th1(t) + th2(t) + th4(t) + th5(t)

   #5 == 2 l2 lc2 th1p(t) th2p(t)

          ________       _______              ________
   #6 == th1p(t) #12 - d3p(t) sin(#21) + th2p(t) #12

          ________       ________       _______
   #7 == th1p(t) #13 + th2p(t) #13 - d3p(t) cos(#21)

   #8 == th1p(t) #14 - sin(#22) d3p(t) + th2p(t) #14

   #9 == th1p(t) #15 - cos(#22) d3p(t) + th2p(t) #15

                                ________
                                th5p(t)
   #10 == #17 + #16 + #18 + -------
                                   2

   #11 == th2p(t) #19 + th1p(t) #20

                         __     _____
   #12 == sin(#21) l2 - d3(t) cos(#21)

           _____                      __
   #13 == d3(t) sin(#21) + cos(#21) l2

   #14 == l2 sin(#22) - cos(#22) d3(t)

   #15 == l2 cos(#22) + sin(#22) d3(t)

           ________
           th2p(t)
   #16 == -------
              2

           ________
           th1p(t)
   #17 == -------
              2

           ________
           th4p(t)
```

```
   #18 == -------
             2

                     2
   #19 == |th1p(t)|

                     2
   #20 == |th2p(t)|

           ______    ______
   #21 == th1(t) + th2(t)

   #22 == th1(t) + th2(t)
```

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

```
                                                                                                   __            __            __
Izz1 #19                                                                                           m3 #7 #9      m4 #7 #9      m5 #7
-------- + Izz4 (th1p(t) + th2p(t) + th4p(t)) (#17 + #16 + #18) + Izz3 #2 (#17 + #16) + -------- + -------- + ------
    2                                                                                       2             2             2

where

          ______   ______   ______   ______
   #1 == th1(t) + th2(t) + th4(t) + th5(t)

   #2 == th1p(t) + th2p(t)

   #3 == th1p(t) + th2p(t) + th4p(t) + th5p(t)

   #4 == th1(t) + th2(t) + th4(t) + th5(t)

   #5 == 2 l2 lc2 th1p(t) th2p(t)

          _______        ______            _______
   #6 == th1p(t) #12 - d3p(t) sin(#21) + th2p(t) #12

          _______       _______       ______
   #7 == th1p(t) #13 + th2p(t) #13 - d3p(t) cos(#21)

   #8 == th1p(t) #14 - sin(#22) d3p(t) + th2p(t) #14

   #9 == th1p(t) #15 - cos(#22) d3p(t) + th2p(t) #15

                                   _______
                                   th5p(t)
   #10 == #17 + #16 + #18 + -------
                                      2

   #11 == th2p(t) #19 + th1p(t) #20

                             __     _____
   #12 == sin(#21) l2 - d3(t) cos(#21)

           _____                      __
   #13 == d3(t) sin(#21) + cos(#21) l2

   #14 == l2 sin(#22) - cos(#22) d3(t)

   #15 == l2 cos(#22) + sin(#22) d3(t)
```

```
       _______
       th2p(t)
#16 == -------
          2

       _______
       th1p(t)
#17 == -------
          2

       _______
       th4p(t)
#18 == -------
          2

                     2
#19 == |th1p(t)|

                     2
#20 == |th2p(t)|

       ______   ______
#21 == th1(t) + th2(t)

#22 == th1(t) + th2(t)
```