```
%Limpieza de pantalla
clear
close all
clc

tic
%Declaración de variables simbólicas
syms th1(t) th2(t) t   %Angulos de cada articulación
syms th1p(t) th2p(t)   %Velocidades de cada articulación
syms th1pp(t) th2pp(t) %Aceleraciones de cada articulación
syms m1 m2 Ixx1 Iyy1 Izz1 Ixx2 Iyy2 Izz2  %Masas y matrices de Inercia
syms l1 l2 lc1 lc2 %l=longitud de eslabones y lc=distancia al centro de
masa de cada eslabón
syms pi g a cero
```

En esta sección, se limpian las variables y se declaran las variables simbólicas necesarias para el modelado dinámico del robot.

```
 %Creamos el vector de coordenadas articulares
  Q= [th1; th2];
 %disp('Coordenadas generalizadas');
 %pretty (Q);

 %Creamos el vector de velocidades articulares
  Qp= [th1p; th2p];
 %disp('Velocidades generalizadas');
 %pretty (Qp);
 %Creamos el vector de aceleraciones articulares
  Qpp= [th1pp; th2pp];
 %disp('Aceleraciones generalizadas');
 %pretty (Qpp);

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

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

Se define el vector de coordenadas articulares, velocidades y aceleraciones. También se configura el tipo de juntas (rotacionales o prismáticas) y el número de grados de libertad (GDL).

```

%Articulación 1
%Posición de la articulación 1 respecto a 0
P(:,:,1)= [l1*cos(th1); l1*sin(th1);0];
%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];

%Articulación 2
%Posición de la articulación 2 respecto a 1
P(:,:,2)= [l2*cos(th2); l2*sin(th2);0];
%Matriz de rotación de la junta 1 respecto a 0
R(:,:,2)= [cos(th2) -sin(th2)  0;
            sin(th2)  cos(th2)  0;
            0         0         1];

%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
```

Se calculan las matrices de transformación homogénea para cada articulación, que describen la posición y orientación de cada eslabón respecto al sistema de referencia inercial.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%CALCULAMOS LAS VELOCIDADES PARA CADA
ESLABÓN%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

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

for k= 1:GDL
    if RP(k)==0
       %Para las juntas de revolución
        try
            Jv_a2(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL)-PO(:,:,k-1));
            Jw_a2(:,k)= RO(:,3,k-1);
        catch
            Jv_a2(:,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_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);
pretty(V2)
```

```
/ - th1p(t) (l1 sin(th1(t)) + l2 #1) - l2 #1 th2p(t) \
|                                                     |
|  th1p(t) (l1 cos(th1(t)) + l2 #2) + l2 #2 th2p(t)  |
|                                                     |
\                          0                          /

where

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

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

```
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);
pretty(W2)
```

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

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

%Calculamos el jacobiano lineal y angular de forma analítica
Jv_a1(:,GDL-1)=PO(:,:,GDL-1);
Jw_a1(:,GDL-1)=PO(:,:,GDL-1);

for k= 1:GDL-1
    if RP(k)==0
       %Para las juntas de revolución
        try
            Jv_a1(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-1)-PO(:,:,k-1));
            Jw_a1(:,k)= RO(:,3,k-1);
        catch
            Jv_a1(:,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_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 * th1p(t));  % Multiplicamos Jv_a1 por th1p(t)
pretty(V1)
```

```
/ -l1 sin(th1(t)) th1p(t) \
|                         |
|  l1 cos(th1(t)) th1p(t) |
|                         |
\            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 * th1p(t));  % Multiplicamos Jw_a1 por th1p(t)
pretty(W1)
```

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

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

Se calculan los Jacobianos lineales y angulares para cada eslabón, que relacionan las velocidades articulares con las velocidades lineales y angulares del efector final.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%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

%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];

%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             2                                                2           2
Izz1 |th1p(t)|     |th1p(t)|  cos(th1(t) - th1(t)) m1 (l1 + lc1) (lc1 |l1|  + l1 |lc1| )
--------------- + -------------------------------------------------------------------
       2                                          2 l1 lc1
```

```


%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);
```

```
__                                                                                       ________          __
m2 (th1p(t) (l1 sin(th1(t)) + l2 sin(#4)) + l2 sin(#4) th2p(t) + lc2 sin(th2(t)) #1) (th1p(t) (sin(#3) l2
-------------------------------------------------------------------------------------------------------------
                                                                                   2

where

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

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

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

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

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

```
Energía Cinética Total
```

```
pretty (K_Total);
```

```
                 __                                                                                       ________
Izz1 #5          m2 (th1p(t) (l1 sin(th1(t)) + l2 sin(#4)) + l2 sin(#4) th2p(t) + lc2 sin(th2(t)) #1) (th1p(t) (s
------- + -------------------------------------------------------------------------------------------------------
   2                                                                                         2

where

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

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

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

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

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

```
%Energia 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

 U1=m1*g*h1
```

U1 = $g\,\mathrm{lc}_1\,m_1\sin(\mathrm{th}_1(t))$

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

U2 = $g\,\mathrm{lc}_2\,m_2\,\sin(\mathrm{th}_2(t))$

```

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

Se calcula la energía cinética y potencial para cada eslabón, que son necesarias para obtener el Lagrangiano del sistema.

```

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

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

Se calcula el Lagrangiano como la diferencia entre la energía cinética y potencial, y se obtiene el modelo de energía total del sistema.