```
% Limpieza de pantalla
clear
close all
clc

tic
% Declaración de variables simbólicas
syms d1(t) d2(t) d3(t) t   % Desplazamientos de cada articulación (X, Y, Z)
syms d1p(t) d2p(t) d3p(t)   % Velocidades de cada articulación
syms d1pp(t) d2pp(t) d3pp(t) % Aceleraciones de cada articulación
syms m1 m2 m3 Ixx1 Iyy1 Izz1 Ixx2 Iyy2 Izz2 Ixx3 Iyy3 Izz3  % Masas y
matrices de Inercia
syms l1 l2 l3 lc1 lc2 lc3 % Longitudes de eslabones y distancias al centro
de masa
syms pi g a cero

% Creamos el vector de coordenadas articulares
Q = [d1; d2; d3];
% disp('Coordenadas generalizadas');
% pretty(Q);

% Creamos el vector de velocidades articulares
Qp = [d1p(t); d2p(t); d3p(t)];
% disp('Velocidades generalizadas');
% pretty(Qp);
% Creamos el vector de aceleraciones articulares
Qpp = [d1pp; d2pp; d3pp];
% disp('Aceleraciones generalizadas');
% pretty(Qpp);
```

Se limpia la pantalla y se inicia un cronómetro para medir el tiempo de ejecución. Se definen variables simbólicas para representar los desplazamientos, velocidades y aceleraciones de las articulaciones, así como las masas, momentos de inercia y longitudes de los eslabones. Además, se crean los vectores de coordenadas generalizadas y se establece que el robot tiene 3 grados de libertad (GDL) con juntas prismáticas.

```
% Configuración del robot, 1 para junta prismática
RP = [1 1 1]; % Todas las juntas son prismáticas

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

% Articulación 1 (Movimiento a lo largo del eje X)
% Posición de la articulación 1 respecto a 0
P(:,:,1) = [d1; 0; 0];
% Matriz de rotación de la junta 1 respecto a 0
R(:,:,1) = [1 0 0;
            0 1 0;
```

```
            0 0 1];

% Articulación 2 (Movimiento a lo largo del eje Y)
% Posición de la articulación 2 respecto a 1
P(:,:,2) = [0; d2; 0];
% Matriz de rotación de la junta 2 respecto a 1
R(:,:,2) = [1 0 0;
            0 1 0;
            0 0 1];

% Articulación 3 (Movimiento a lo largo del eje Z)
% Posición de la articulación 3 respecto a 2
P(:,:,3) = [0; 0; d3];
% Matriz de rotación de la junta 3 respecto a 2
R(:,:,3) = [1 0 0;
            0 1 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 determinan las posiciones y matrices de rotación de cada articulación, teniendo en cuenta que el robot es cartesiano, por lo que todas las matrices de rotación son la identidad. Se generan las matrices homogéneas locales y globales, las cuales permiten describir la transformación entre los distintos eslabones del robot.

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

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

% Calculamos el jacobiano lineal de forma analítica
Jv_a3 = jacobian(PO(:,:,GDL), [d1, d2, d3]); % Jacobiano lineal
Jw_a3 = zeros(3, GDL); % Jacobiano angular (cero porque no hay rotación)

% Obtenemos SubMatrices de Jacobianos
Jv_a3 = simplify(Jv_a3); % Simplificamos el jacobiano lineal
Jw_a3 = sym(Jw_a3); % Aseguramos que sea simbólico (aunque es cero)

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

% 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);
pretty(V3)

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

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

% Calculamos el jacobiano lineal de forma analítica
Jv_a2 = jacobian(PO(:,:,GDL-1), [d1, d2]); % Jacobiano lineal (debe ser 3x2)
Jw_a2 = zeros(3, GDL-1); % Jacobiano angular (cero porque no hay rotación)

% Aseguramos que el jacobiano angular sea simbólico
Jw_a2 = sym(Jw_a2);

% Simplificamos el jacobiano lineal (el angular no necesita simplificación)
Jv_a2 = simplify(Jv_a2);
```

```
% Verificamos las dimensiones de Jv_a2
disp('Dimensiones de Jv_a2:');
disp(size(Jv_a2)); % Debe ser 3x2

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

% 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)); % Jv_a2 es 3x2, Qp(1:2) es 2x1
pretty(V2)

disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón
2');
W2 = simplify(Jw_a2 * Qp(1:2)); % Jw_a2 es 3x2, Qp(1:2) es 2x1
pretty(W2)
% 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)); % Jv_a2 es 3x2, Qp(1:2) es 2x1
pretty(V2)

disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón
2');
W2 = simplify(Jw_a2 * Qp(1:2)); % Jw_a2 es 3x2, Qp(1:2) es 2x1
pretty(W2)
%%%%%%%%%%  VELOCIDADES PARA ESLABÓN 1  %%%%%%%%%%

% Calculamos el jacobiano lineal y angular de forma analítica
Jv_a1 = jacobian(PO(:,:,GDL-2), d1);
Jw_a1 = sym(zeros(3, 1)); % No hay rotación en un robot cartesiano

% Obtenemos SubMatrices de Jacobianos
Jv_a1 = simplify(Jv_a1);
Jw_a1 = simplify(Jw_a1);

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

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

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

Se calculan los jacobianos lineales y angulares de cada eslabón para obtener sus velocidades lineales y angulares. Dado que el robot es cartesiano, los jacobianos angulares son nulos. Finalmente, se simplifican las expresiones y se presentan los resultados.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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 = [lc1; 0; 0]; % Centro de masa del eslabón 1
P12 = [0; lc2; 0]; % Centro de masa del eslabón 2
P23 = [0; 0; lc3]; % Centro de masa del eslabón 3

% 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

% Calculamos la energía cinética para cada uno de los eslabones
% Eslabón 1
V1_Total = V1; % No hay rotación, solo traslación
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);

% Eslabón 2
V2_Total = V2; % No hay rotación, solo traslación
K2 = (1/2 * m2 * (V2_Total.' * V2_Total)) + (1/2 * W2.' * (I2 * W2));
disp('Energía Cinética en el Eslabón 2');
K2 = simplify(K2);
pretty(K2);

% Eslabón 3
V3_Total = V3; % No hay rotación, solo traslación
```

```
K3 = (1/2 * m3 * (V3_Total.' * V3_Total)) + (1/2 * W3.' * (I3 * W3));
disp('Energía Cinética en el Eslabón 3');
K3 = simplify(K3);
pretty(K3);

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

Se calcula la energía cinética de cada eslabón, considerando que el robot no tiene rotaciones. Luego, se obtiene la energía cinética total sumando la contribución de cada eslabón.

```
% Energía Potencial (p = mgh)
% Obtenemos las alturas respecto a la gravedad
h1 = P01(3); % Altura del eslabón 1
h2 = P12(3); % Altura del eslabón 2
h3 = P23(3); % Altura del eslabón 3

U1 = m1 * g * h1;
U2 = m2 * g * h2;
U3 = m3 * g * h3;

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

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

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

toc
```

Se calcula la energía potencial de cada eslabón considerando la altura respecto a la gravedad, luego se obtiene la energía total y el lagrangiano, que describe la dinámica del sistema.