

## INTEGRANTES:

A01735217 - Diego García Rueda

A01734225 - Jonathan Josafat Vázquez Suárez

A01734193 - Jhonatan Yael Martinez Vargas

## DESARROLLO DE LA ACTIVIDAD:

Se usan comandos de limpieza de *workspace*

```
clear all  
close all  
clc
```

Se genera el comando *tic* para inicializar el tiempo de ejecución del programa

```
tic
```

Se crean variables simbolicas que se usarán en todo el programa.

```
syms th1(t) t
```

Se crean las variables simbolicas de las masas y las matrices de inercia. (h es la altura del soporte)

```
syms m1 Ixx1 Iyy1 Izz1 g h
```

Se crean las variables simbolicas que representan la longitud de cada junta y la distancia de cada junta al centro de masa.

```
syms l1 lc1
```

Se establece la configuración del robot, en este caso las 3 articulaciones son prismaticas.

```
RP=[0];
```

Se crea un vector de coordenadas particulares. Posteriormente este vector es derivado para obtener un vector de velocidades articulares y aceleraciones articulares.

```
Q = [th1];  
Qp = diff(Q, t);  
Qpp = diff(Qp, t);
```

Se establece el número de los Grados De Libertad (GDL) tanto como valor numerico como dato de tipo *String*

```
GDL= size(RP,2);  
GDL_str= num2str(GDL);
```

## ARTICULACIÓN 1

Se crea el vector de traslación de la junta 1. Se crea también la matriz de rotación respecto al origen. (-90° en el eje X) Ambos valores se guardan en la página 1.

```
P(:,:,1)= [l1*cos(th1); l1*sin(th1); 0];  
R(:,:,1)= [cos(th1) -sin(th1) 0;  
            sin(th1) cos(th1) 0;  
            0         0         1];
```

## CREACIÓN DE MATRICES GLOBALES Y LOCALES

Se crea un vector de ceros, Este vector se usa para "completar" las matrices homogéneas locales y globales, este vector permite que esta matriz sea una matriz cuadrada.

```
Vector_Zeros= zeros(1, 3);
```

Se inicializan las matrices de transformación locales y global usando los vectores de traslación y las matrices de rotación. Para realizar las matrices cuadradas se agrega el vector de ceros.

```
A(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);  
T(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
```

Se inicializan los valores de los vectores de posición y de rotación (vistos desde el marco de referencia inercial).

```
PO(:,:,GDL)= P(:,:,GDL);  
RO(:,:,GDL)= R(:,:,GDL);
```

Se realiza un ciclo *for* en el que usando los Grados De Libertad (GDL) como iterador. Usando este iterador se van creando las matrices locales de cada junta.

Posteriormente se van creando las matrices de transformación globales. Aquí existen 2 casos en los que se pueden generar estas matrices:

- Caso #1: Solo hay 1 GDL por lo que la matriz global es igual a la matriz de transformación local de esa junta.
- Caso #2: Hay más de 1 GDL, en ese caso, la matriz global se genera al multiplicar la matriz global anterior (es decir, de la articulación anterior) por la matriz local actual (de la articulación actual) generando así la nueva matriz de transformación global.

Obtenemos la matriz de rotación (RO) y el vector de traslación (PO) a partir de la Matriz de Transformación Homogénea Global.

```
for i = 1:GDL  
    i_str= num2str(i);  
    %Matrices Locales  
    A(:,:,i)=simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);  
  
    %Matrices Globales
```

```

try
    T(:, :, i) = T(:, :, i-1) * A(:, :, i);
catch
    T(:, :, i) = A(:, :, i);    % GDL == 1
end

T(:, :, i) = simplify(T(:, :, i));

%Da la matriz de rotacion de la matriz global
RO(:, :, i) = T(1:3, 1:3, i);
%Vector de traslación de la matriz global
PO(:, :, i) = T(1:3, 4, i);

end

```

## CALCULO DE JACOBIANOS

### Método de diferenciación:

Primero obtenemos el Jacobiano Lineal usando el método de diferenciación. Se realizan derivadas parciales respecto a x, y, z obteniendo así 3 derivadas parciales. Para esto se usan los vectores de posición

```

Jv11= functionalDerivative(PO(1,1,GDL), th1);

Jv21= functionalDerivative(PO(2,1,GDL), th1);

Jv31= functionalDerivative(PO(3,1,GDL), th1);

```

Se crea la matriz del Jacobiano lineal.

```

jv_d=simplify([Jv11; Jv21; Jv31]);

```

### Método Análítico:

Se inicializan los Jacobianos Análíticos (Lineal y Angular). Para esto se utiliza una función llamada *calculo\_analitico\_Jacobianos* quedando de la siguiente manera:

```

[Jv_a, Jw_a] = calculo_analitico_Jacobianos(PO, RO, RP, GDL);

```

En la sección de *try* se encuentran las fórmulas del Jacobiano Lineal Analítico y del Jacobiano Angular Analítico respectivamente.

En la sección de *catch* se encuentran las fórmulas para los Jacobianos en caso de que solo haya 1 GDL, debido a que no tenemos articulaciones previas usando así una matriz identidad.

Se simplifican los Jacobianos obtenidos:

```

Jv_a= simplify (Jv_a);
Jw_a= simplify (Jw_a);

```

Se mandan a imprimir las velocidades lineal y angular obtenidas usando sus Jacobianos correspondientes diferenciandolos.

```
disp('Velocidad lineal obtenida mediante el Jacobiano lineal (Última junta)');
```

Velocidad lineal obtenida mediante el Jacobiano lineal (Última junta)

```
V=simplify (Jv_a*Qp');
pretty(V);
```

$$\begin{pmatrix} \frac{d}{dt} (-l_1 \sin(\theta_1(t))) \\ \frac{d}{dt} (l_1 \cos(\theta_1(t))) \\ 0 \end{pmatrix}$$

```
disp('Velocidad angular obtenida mediante el Jacobiano angular (Última junta)');
```

Velocidad angular obtenida mediante el Jacobiano angular (Última junta)

```
W=simplify (Jw_a*Qp');
pretty(W);
```

$$\begin{pmatrix} 0 \\ 0 \\ \frac{d}{dt} \theta_1(t) \end{pmatrix}$$

## ENERGÍA CINÉTICA

Se crean los vectores de posición (distancia del origen de la junta a su centro de masa).

```
P01=subs(P(:, :, 1)/2, l1, lc1);
```

Se crean las matrices de inercia de cada junta.

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

Se extraen las velocidades lineales y angulares de cada eje.

```
%Velocidades lineales
V=V(t);
Vx= V(1,1);
```

```
Vy= V(2,1);  
Vz= V(3,1);  
  
%Velocidades angulares  
W=W(t);  
W_pitch= W(1,1);  
W_roll= W(2,1);  
W_yaw= W(3,1);
```

### CALCULO ENERGÍA CINÉTICA DE CADA JUNTA

## ESLABON 1:

Usando las formulas de energía cinetica

```
V1_Total = V+cross(W,P01);  
pretty(V1_Total*m1)
```

$$\begin{aligned}
& \frac{\sin(\theta_1(t))}{l_{c1}} + \frac{l_{c1} \sin(\theta_1(t))}{2} \\
& \frac{\cos(\theta_1(t))}{l_{c1}} + \frac{l_{c1} \cos(\theta_1(t))}{2}
\end{aligned}$$

```
pretty((1/2*m1*(Vl_Total))')
```

$$\frac{1}{2} m_1 \left( \dot{\theta}_1(t) \right)^2 + \frac{1}{2} m_1 \left( \dot{\theta}_1(t) \right)^2 \sin^2(\theta_1(t))$$

where

$$\#1 == \sin(\theta_1(t))$$

$$\#2 == \cos(\theta_1(t))$$

```
%K1 = (1/2*m1*(Vl_Total.^2))+(1/2*I1*(W.^2))
K1 = (1/2*m1*(Vl_Total))'*(1/2*m1*(Vl_Total)) + (1/2*W)'*(I1*W);
K1 = simplify (K1);
disp('Energía Cinética de la Junta 1:')
```

Energía Cinética de la Junta 1:

```
pretty(K1)
```

$$\frac{1}{2} m_1 \left( \dot{\theta}_1(t) \right)^2 + \frac{1}{2} I_1 \left( \dot{\theta}_1(t) \right)^2$$

$$\frac{\cos(\theta_1(t)) - \theta_1(t)) |m_1|}{(l_1 |l_{c1}|^2 + 2 l_{c1} |l_1|^2) \sqrt{(2 l_1 + l_{c1}) / (16 l_1 l_{c1})}}$$

## SUMA ENERGÍAS CINÉTICAS

Se suman las energías cinéticas de todos los eslabones quedando de la siguiente manera:

```
K_Total= simplify (K1);
disp('Energía Cinética Total (todas las juntas)')
```

Energía Cinética Total (todas las juntas)

```
pretty(K_Total)
```

$$\frac{I_{zz1} \left| \frac{d}{dt} \theta_1(t) \right|^2}{2} + \frac{\cos(\theta_1(t)) - \theta_1(t)) |m_1|}{(l_1 |l_{c1}|^2 + 2 l_{c1} |l_1|^2) \sqrt{(2 l_1 + l_{c1}) / (16 l_1 l_{c1})}}$$

## CALCULO ENERGÍA POTENCIAL TOTAL:

Primero se obtienen las alturas respecto a la gravedad de cada una de las juntas.

```
h1= P01(2) + h;
```

A partir de las alturas se calcula la energía potencial. Utilizando como parametros la masa del eslabon, su altura y la gravedad

```
U1=m1*g*h1;
```

Se calcula la energía potencial total sumando la energía potencial de cada eslabon, quedando de la siguiente manera:

```
U_Total= U1
```

```
U_Total =
```

$$\frac{g l c_1 m_1 \sin(\theta_1(t))}{2}$$

Se manda a imprimir la suma de la energía cinética total y de la energía potencial total.

```
H= simplify (K_Total+U_Total);
pretty (H)
```

$$\frac{I_{zz1} \left( \frac{d}{dt} \theta_1(t) \right)^2}{2} + \frac{g l c_1 m_1 \sin(\theta_1(t))}{2} + \frac{\sqrt{\frac{d}{dt} \theta_1(t)}}{\cos(\theta_1(t) - \theta_1(t)) |m_1|} \frac{(l_1 |l c_1|^2 + 2 l c_1 |l_1|^2)}{(2 l_1 + l c_1) \sqrt{(16 l_1 l c_1)}}$$

## CALCULO DEL LANGRAGIANO

Ya teniendo los valores de la energia cinetica total y de la energía potencial total podemos realizar el calculo del Langragiano:

```
Lagrangiano= simplify (K_Total-U_Total);
pretty (Lagrangiano);
```

$$\frac{I_{zz1} \left( \frac{d}{dt} \theta_1(t) \right)^2}{2} - \frac{g l c_1 m_1 \sin(\theta_1(t))}{2} + \frac{\sqrt{\frac{d}{dt} \theta_1(t)}}{\cos(\theta_1(t) - \theta_1(t)) |m_1|} \frac{(l_1 |l c_1|^2 + 2 l c_1 |l_1|^2)}{(2 l_1 + l c_1) \sqrt{(16 l_1 l c_1)}}$$



$$(2 \ 11 \ + \ 1c1) \ \left| \begin{array}{c} \backslash \\ / \end{array} \right. / (16 \ 11 \ 1c1)$$

Se usa el comando *toc* para mostrar el tiempo de ejecución del programa.

```
toc
```

```
Elapsed time is 2.887723 seconds.
```