

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 crean variables simbolicas que se usarán en todo el programa.

```
syms l1(t) l2(t) l3(t) t
```

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

```
RP=[1 1 1];
```

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

```
Q = [l1, l2, l3];
Qp = diff(Q, 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 Y) Ambos valores se guardan en la página 1.

```
P(:, :, 1) = [l1; 0; l1];
R(:, :, 1) = [0 0 -1;
              0 1 0;
              1 0 0];
```

ARTICULACIÓN 2

Se crea el vector de traslación de la junta 2. Se crea también la matriz de rotación respecto a la junta anterior. (-90° en el eje X) Ambos valores se guardan en la página 2.

```
P(:, :, 2) = [0; 0; l2];
R(:, :, 2) = [1 0 0;
              0 0 1;
```

```
0 -1 0];
```

ARTICULACIÓN 3

Se crea el vector de traslación de la junta 3. Se crea también la matriz de rotación respecto a la junta anterior. Ambos valores se guardan en la página 3.

```
P(:,:,3)=[0; 0; 13];  
R(:,:,3)=[1 0 0;  
          0 1 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 9 derivadas parciales. Para esto se usan los vectores de posición

```

Jv11= functionalDerivative(PO(1,1,GDL), l1);
Jv12= functionalDerivative(PO(1,1,GDL), l2);
Jv13= functionalDerivative(PO(1,1,GDL), l3);

Jv21= functionalDerivative(PO(2,1,GDL), l1);
Jv22= functionalDerivative(PO(2,1,GDL), l2);
Jv23= functionalDerivative(PO(2,1,GDL), l3);

Jv31= functionalDerivative(PO(3,1,GDL), l1);
Jv32= functionalDerivative(PO(3,1,GDL), l2);
Jv33= functionalDerivative(PO(3,1,GDL), l3);

```

Se crea la matriz del Jacobiano lineal.

```

jv_d=simplify([Jv11 Jv12 Jv13; Jv21 Jv22 Jv23; Jv31 Jv32 Jv33]);

```

Método Análítico:

Se inicializan los Jacobianos Análíticos (Lineal y Angular). Posteriormente, usando de nuevo un ciclo *for* usando como argumento la configuración del robot (RP), en este caso las 3 articulaciones son prismáticas. El ciclo *for* queda de la siguiente manera:

```

Jv_a(:, GDL) = PO(:, :, GDL);
Jw_a(:, GDL) = PO(:, :, GDL);

for k= 1:GDL
    if RP(k) == 0

```

```

%Para las juntas de revolución
try
    Jv_a(:,k)= cross(RO(:,3,k-1), PO(:, :,GDL)-PO(:, :,k-1));
    Jw_a(:,k)= RO(:,3,k-1);
catch
    Jv_a(:,k)= cross([0,0,1], PO(:, :,GDL));
    Jw_a(:,k)=[0,0,1];
end
else
%
    %Para las juntas prismáticas
    try
        Jv_a(:,k)= RO(:,3,k-1);
    catch
        Jv_a(:,k)=[0,0,1];
    end
    Jw_a(:,k)=[0,0,0];
end
end
end

```

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

En la sección de *catch* se encuentran las formulas 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 diferenciándolos.

```

disp('Velocidad lineal obtenida mediante el Jacobiano lineal');

```

```

Velocidad lineal obtenida mediante el Jacobiano lineal

```

```

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

```

```

/      _____      \
|      d              |
|  - -- 12(t)         |
|      dt              |
|      _____      |
|      d              |
|  - -- 13(t)         |
|      dt              |
|      _____      |
|      d              |
|  - -- 11(t)         |
|      dt              |
\      _____      /

```

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

Velocidad angular obtenida mediante el Jacobiano angular

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

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

CONCLUSIONES

En este ejercicio se puede observar como las velocidades lineales son la derivada del desplazamiento de cada una de las 3 juntas (I1, I2 y I3 respectivamente) al ser la derivada de desplazamiento esta es la velocidad.

En el caso de la velocidad angular en todos los ejes es 0 debido a que todas las juntas son prismatica, es decir, no tienen rotación por lo tanto la velocidad angular es inexistente.