Actividad 1 (Velocidades Lineales y angulares)

Primero limpiamos el workspace para no tener errores de otras variables que podrían interferir y cerramos ventana

```
close all
clc

syms l1 l2 t th1(t) th2(t)
```

Configuramos nuestras variables para dos grados de libertad debido a que hay 2 articulaciones th1 y th2

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

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

```
Coordenadas generalizadas
```

```
pretty(Q);
```

```
(th1(t), th2(t))
```

```

% Creamos el vector de velocidades generalizadas
Qp = [diff(th1, t); diff(th2, t)];

disp('Velocidades generalizadas');
```

```
Velocidades generalizadas
```

```
pretty(Qp);
```

```
/  d       \
| -- th1(t) |
| dt        |
|           |
|  d        |
| -- th2(t) |
\ dt        /
```

```

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

% Junta 1
% Posición de la junta 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];

% Junta 2
% Posición de la junta 2 respecto a 1
P(:,:,2) = [l2 * cos(th2);
            l2 * sin(th2);
            0];

% Matriz de rotación de la junta 2 respecto a 1
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 homogeneas para ambas articulaciones, controlando con GDL

```
%Inicializamos las matrices de transformación de 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 los vectores de posición vistos desde el marco de referencia
%inercial

PO(:,:,GDL)= P(:,:,GDL);

%Inicializamos las matrcides de rotación vistas desde el marco de
%referencia inercial
RO(:,:,GDL)= R(:,:,GDL);
```

Tenemos que calcular las matrices de tranformación homogénea para ambas articulaciones

```
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));

    try
        T(:,:,i)= T(:,:,i-1)*A(:,:,i);
```

```
    catch
        T(:,:,i)= A(:,:,i); %Caso específico cuando i=1 nos marcarìa error el try
    end
    disp(strcat('Matriz de transformación global T ', i_str));
    T(:,:,i)= simplify(T(:,:,i));
    pretty(T(:,:,i));

%Obtenemos la matriz de rotación "RO" y el vector de translación PO de la
%matriz de transformación homogénea globla T(:,:,GDL)
    RO(:,:,i)= T(1:3,1:3,i);
    PO(:,:,i)= T(1:3,4,i);
    pretty(RO(:,:,i));
    pretty(PO(:,:,i));
end
```

```
Matriz de transformación local A1
/ cos(th1(t)), -sin(th1(t)), 0, l1 cos(th1(t)) \
|                                               |
| sin(th1(t)),  cos(th1(t)), 0, l1 sin(th1(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
Matriz de transformación global T1
/ cos(th1(t)), -sin(th1(t)), 0, l1 cos(th1(t)) \
|                                               |
| sin(th1(t)),  cos(th1(t)), 0, l1 sin(th1(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
/ cos(th1(t)), -sin(th1(t)), 0 \
|                              |
| sin(th1(t)),  cos(th1(t)), 0 |
|                              |
\      0,            0,      1 /
/ l1 cos(th1(t)) \
|                |
| l1 sin(th1(t)) |
|                |
\        0       /
Matriz de transformación local A2
/ cos(th2(t)), -sin(th2(t)), 0, l2 cos(th2(t)) \
|                                               |
| sin(th2(t)),  cos(th2(t)), 0, l2 sin(th2(t)) |
|                                               |
|      0,            0,      1,        0       |
|                                               |
\      0,            0,      0,        1       /
Matriz de transformación global T2
/ #2, -#1, 0, l1 cos(th1(t)) + l2 #2 \
|                                     |
| #1,  #2, 0, l1 sin(th1(t)) + l2 #1 |
|                                     |
|  0,  0,  1,           0             |
|                                     |
\  0,  0,  0,           1             /

where
```

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

   #2 == cos(th1(t) + th2(t))
/ cos(th1(t) + th2(t)), -sin(th1(t) + th2(t)), 0 \
|                                               |
| sin(th1(t) + th2(t)),  cos(th1(t) + th2(t)), 0 |
|                                               |
\          0,                   0,           1 /
/ l1 cos(th1(t)) + l2 cos(th1(t) + th2(t)) \
|                                          |
| l1 sin(th1(t)) + l2 sin(th1(t) + th2(t)) |
|                                          |
\                    0                     /
```

```
%Fin de cinematica directa%

%Calculemos el jacobiano lineal de forma diferencial
disp('Jacobiano lineal obtenido de forma diferencial');
```

```
Jacobiano lineal obtenido de forma diferencial
```

```
% Derivadas parciales respecto a th1 y th2
Jv11 = functionalDerivative(PO(1,1,2), th1);
Jv21 = functionalDerivative(PO(2,1,2), th1);
Jv31 = functionalDerivative(PO(3,1,2), th1);

Jv12 = functionalDerivative(PO(1,1,2), th2);
Jv22 = functionalDerivative(PO(2,1,2), th2);
Jv32 = functionalDerivative(PO(3,1,2), th2);

% Creamos la matriz del Jacobiano lineal
jv_d = simplify([Jv11 Jv12;
                 Jv21 Jv22;
                 Jv31 Jv32]);
pretty(jv_d);
```

```
/ - l1 sin(th1(t)) - l2 sin(th1(t) + th2(t)), -l2 sin(th1(t) + th2(t)) \
|                                                                       |
|  l1 cos(th1(t)) + l2 cos(th1(t) + th2(t)),   l2 cos(th1(t) + th2(t)) |
|                                                                       |
\                    0,                                 0             /
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calculamos el jacobiano lineal y angular de forma analìtica
%Inicializamos jacobianos analíticos (lineal y angulas)
Jv_a(:,GDL)=PO(:,:,GDL);
Jw_a(:,GDL)=PO(:,:,GDL);

for k= 1:GDL
    if (RP(k)==0)%casos: articulacion rotacional y prismica
```

```
        %para las articulaciones rotacionales
        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
    elseif (RP(k)==1)%casos: articulaciones prismáticas
        try
            Jv_a(:,k)=RO(:,3,k-1);
        catch
            Jv_a(:,k)=[0,0,1];%si no hay matriz de rotacion previa, se obtiene la
matriz identidad
        end
        Jw_a(:,k)=[0,0,0];
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%DESPLIEGUE DE CINEMATICA DIFERENCIAL%%%%%%%%%%%%%
Jv_a= simplify(Jv_a);
Jw_a= simplify(Jw_a);
disp('Jacobiano lineal obtenido de forma analítica');
```

```
Jacobiano lineal obtenido de forma analítica
```

```
pretty(Jv_a);
```

```
/ - l1 sin(th1(t)) - l2 sin(th1(t) + th2(t)), -l2 sin(th1(t) + th2(t)) \
|                                                                      |
|  l1 cos(th1(t)) + l2 cos(th1(t) + th2(t)),   l2 cos(th1(t) + th2(t)) |
|                                                                      |
\                      0,                                 0            /
```

```
disp('Jacobiano angular obtenido de forma analítica');
```

```
Jacobiano angular obtenido de forma analítica
```

```
pretty(Jw_a);
```

```
/ 0, 0 \
|      |
| 0, 0 |
|      |
\ 1, 1 /
```

```

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

```
Velocidad lineal obtenida mediante el Jacobiano lineal
```

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

```
/                            d                  d        \
| - (l1 sin(th1(t)) + l2 #1) -- th1(t) - l2 #1 -- th2(t) |
|                            dt                 dt       |
|                                                        |
|                            d                  d        |
|  (l1 cos(th1(t)) + l2 #2) -- th1(t) + l2 #2 -- th2(t)  |
|                           dt                 dt        |
|                                                        |
\                            0                           /
```

where

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

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

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

```
Velocidad angular obtenida mediante el Jacobiano angular
```

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

```
/           0           \
|                       |
|           0           |
|                       |
|  d           d        |
| -- th1(t) + -- th2(t) |
\ dt          dt        /
```