Tecnológico de Monterrey

Velocidades Lineales y angulares

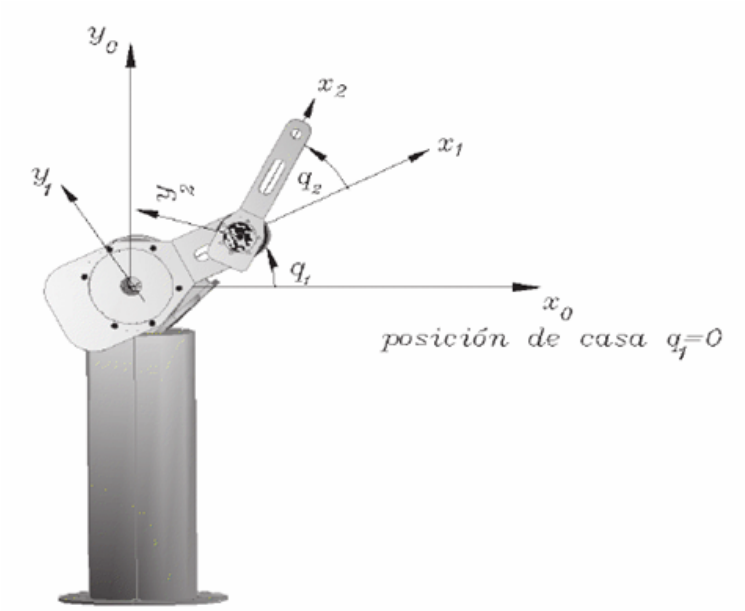
Emmanuel Lechuga Arreola || A01736241

Instituto Tecnológico y de Estudios Superiores de Monterrey

19/02/2025

Planteamiento

Dados los conocimientos que se obtuvieron en la sesión previa se requiere modificar el código empleado para que funcione teniendo en cuenta un brazo robótico con dos articulaciones. Descrito en la imagen siguiente:



brazo de dos articulaciones

Pasos:

Hay que declarar las variables simbólicas a emplear: th2 y l2 corresponden al segundo brazo

```
% Declaración de variables simbólicas
syms th1(t) th2(t) l1 l2 t  % Agregamos th2(t) y l2
```

Lo que viene es declarar los vectores que contengan la posición y la velocidad de las articulaciones. De igual forma se crea la variable de los grados de libertad (GDL).

```
% Vector de coordenadas articuladas
Q = [th1; th2];  % Agregamos th2
disp('Coordenadas articuladas:');
pretty(Q);

% Vector de velocidades articuladas
Qp = diff(Q, t);
disp('Velocidades articuladas:');
pretty(Qp);

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

```
% Posiciones de las juntas respecto al marco anterior
P(:,:,1) = [l1*cos(th1);
            l1*sin(th1);
            0];

P(:,:,2) = [l1*cos(th1) + l2*cos(th1 + th2);  % Posición junta 2
            l1*sin(th1) + l2*sin(th1 + th2);
            0];

% Matrices de rotación de cada junta
R(:,:,1) = [cos(th1) -sin(th1) 0;
            sin(th1)  cos(th1) 0;
            0          0       1];

R(:,:,2) = [cos(th1 + th2) -sin(th1 + th2) 0;  % Rotación acumulada
            sin(th1 + th2)  cos(th1 + th2) 0;
            0               0              1];
```

En subsecuente, vienen las matrices de posiciones y rotaciones de las juntas del péndulo.

Posteriormente se crean las variables que corresponden a la matriz de transformación homogénea.

En el ciclo for obtenemos las posiciones y rotaciones de los brazos.

```
% Inicialización de matrices de transformación homogénea
Vector_Zeros = zeros(1, 3);
A = sym(zeros(4, 4, GDL));
T = sym(zeros(4, 4, GDL));
PO = sym(zeros(3, 1, GDL));
RO = sym(zeros(3, 3, GDL));

% Bucle para calcular transformaciones homogéneas
for i = 1:GDL
    % Matrices locales (A_i)
    A(:,:,i) = simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
    disp(['Matriz de transformación local A', num2str(i)]);
    pretty(A(:,:,i));

    % Matrices globales (T_i = T_prev * A_i)
    if i == 1
        T(:,:,i) = A(:,:,i);
    else
        T(:,:,i) = simplify(T(:,:,i-1) * A(:,:,i));
    end
    disp(['Matriz de transformación global T', num2str(i)]);
    pretty(T(:,:,i));

    % Posiciones y rotaciones globales
    RO(:,:,i) = T(1:3, 1:3, i);
    PO(:,:,i) = T(1:3, 4, i);
end
```

```
Posición final del efector:
/ l1 cos(th1(t)) + l1 cos(2 th1(t)) + l2 cos(2 th1(t) + th2(t)) \
|                                                               |
| l1 sin(th1(t)) + l1 sin(2 th1(t)) + l2 sin(2 th1(t) + th2(t)) |
|                                                               |
\                             0                                 /
```

```
%%%%%%%%%%%%%%%%%%%% Cinemática Directa %%%%%%%%%%%%%%%%%%%%
% Posición final del efector (PO(:,:,GDL))
disp('Posición final del efector:');
pretty(PO(:,:,GDL));
```

Obtenemos los Jacobianos

```
%%%%%%%%%%%%%%%%%%%% Jacobianos y Velocidades %%%%%%%%%%%%%%%%%%%%
% Inicialización de Jacobianos analíticos
Jv_a = sym(zeros(3, GDL));
Jw_a = sym(zeros(3, GDL));

% Cálculo de Jacobianos para cada junta
for k = 1:GDL
    if RP(k) == 0  % Junta rotacional
        if k == 1
            % Para la primera junta, usamos el marco base
            Jv_a(:,k) = cross([0; 0; 1], PO(:,:,GDL));
            Jw_a(:,k) = [0; 0; 1];
        else
            % Para juntas posteriores, usamos el marco anterior
            Jv_a(:,k) = cross(RO(:,3,k-1), PO(:,:,GDL) - PO(:,:,k-1));
            Jw_a(:,k) = RO(:,3,k-1);
        end
    end
end

% Simplificación y despliegue de los jacobianos lineal y angular.
Jv_a = simplify(Jv_a);
Jw_a = simplify(Jw_a);
disp('Jacobiano lineal:');
pretty(Jv_a);
disp('Jacobiano angular:');
pretty(Jw_a);
```

```
Jacobiano lineal:
/ - l1 sin(th1(t)) - l1 sin(2 th1(t)) - #1, - l1 sin(2 th1(t)) - #1 \
|                                                                   |
|  l1 cos(th1(t)) + l1 cos(2 th1(t)) + #2,   l1 cos(2 th1(t)) + #2  |
|                                                                   |
\                    0,                                 0          /

where

   #1 == l2 sin(2 th1(t) + th2(t))

   #2 == l2 cos(2 th1(t) + th2(t))

Jacobiano angular:
/ 0, 0 \
|      |
| 0, 0 |
|      |
\ 1, 1 /
```

Mandamos tanto la velocidad lineal como la angular

```
% Velocidades lineales y angulares
V = simplify(Jv_a * Qp);
W = simplify(Jw_a * Qp);
disp('Velocidad lineal :');
pretty(V);
disp('Velocidad angular :');
pretty(W);
```

```
Velocidad lineal :
/                    d           d                                          \
| - (#2 + l2 sin(#1)) -- th2(t) - -- th1(t) (l1 sin(th1(t)) + #2 + l2 sin(#1)) |
|                     dt          dt                                         |
|                                                                            |
|                     d           d                                          |
|  (#3 + l2 cos(#1)) -- th2(t) + -- th1(t) (l1 cos(th1(t)) + #3 + l2 cos(#1))  |
|                     dt          dt                                         |
|                                                                            |
\                                      0                                     /

where

   #1 == 2 th1(t) + th2(t)

   #2 == l1 sin(2 th1(t))

   #3 == l1 cos(2 th1(t))

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