



Tecnológico de Monterrey

Estudiante:

Axel Javier Rosas Rodríguez | A01738607

Profesor:

Alfredo García Suárez

Nombre de la Institución:

Instituto Tecnológico y de Estudios Superiores de
Tecnológico de Monterrey

Materia:

Fundamentación de la Robótica

Fecha:

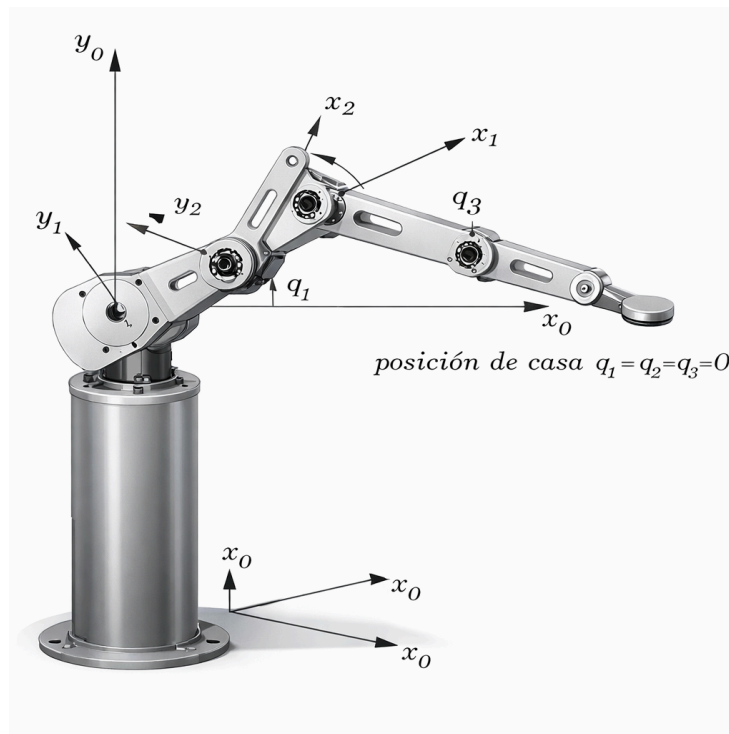
18 de Febrero de 2026

Título del trabajo:

Actividad 1 (Velocidades Lineales y angulares)
Robot Planar 3GDL

Introducción

En robótica, la capacidad de un manipulador para moverse y alcanzar posiciones precisas depende de la descripción de su cinemática y de sus velocidades. Este trabajo desarrolla un modelo de un robot planar de tres grados de libertad (3 GDL) utilizando MATLAB para calcular automáticamente las matrices de transformación, los Jacobianos y los vectores de velocidad lineal y angular.



Descripción

Se analiza un robot manipulador planar de 3 grados de libertad, donde:

- Todas las articulaciones son rotacionales.
- Todos los ejes de giro son paralelos al eje z
- El movimiento ocurre en el plano XY

Teniendo en cuenta

Coordenadas generalizadas

$$\mathbf{q} = \begin{bmatrix} \theta_1(t) \\ \theta_2(t) \\ \theta_3(t) \end{bmatrix}$$

Velocidades generalizadas

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

Metodología

1. Matrices de transformación homogénea

Partimos de obtener las matrices de transformación homogénea para cada eslabón

$$A_i = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & l_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i & 0 & l_i \sin \theta_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Obteniendo la matriz de transformación global como resultado de $T_3 = A_1 * A_2 * A_3$ ya que

- A1 transforma del marco 0 al 1
- A2 transforma del marco 1 al 2
- A3 transforma del marco 2 al 3

2. Jacobiano Angular

Cada articulación rotacional contribuye al vector de velocidad angular del efector final.

Se construye la matriz Jacobiana angular a partir de los ejes de rotación de cada articulación

3. Jacobiano Lineal

Se calcula la velocidad lineal del efector final mediante derivadas parciales de la posición respecto a las coordenadas articulares.

Se puede obtener de forma diferencial (derivadas simbólicas) o analítica (usando producto cruzado de ejes de rotación y posiciones relativas).

4. Cálculo de velocidades del efector final

La velocidad lineal V se obtiene multiplicando el Jacobiano lineal por el vector de velocidades generalizadas:

$$\mathbf{v} = J_v \dot{\mathbf{q}}$$

La velocidad angular W se obtiene multiplicando el Jacobiano angular por el vector de velocidades generalizadas:

$$\boldsymbol{\omega} = J_\omega \dot{\mathbf{q}}$$

5. Resultados

Se obtienen los vectores de posición, velocidad lineal y velocidad angular del efector final.

Estos resultados permiten analizar el comportamiento cinemático del robot en el plano XY.

$$\mathbf{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} \quad \boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix}$$

Código MATLAB

%SECCIÓN 1

```
%Declaración de variables simbólicas  
syms th1(t) th2(t) th3(t) t l1 l2 l3
```

Empezamos por declarar las variables simbólicas. Estas variables articulares se definen como funciones del tiempo (th1(t), th2(t), th3(t)), lo cual es fundamental porque posteriormente se calcularán derivadas para obtener velocidades.

También se declaran las longitudes (l1, l2, l3) lo que permite mantener el modelo completamente general y no depender de valores numéricos.

%SECCIÓN 2

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

Aquí se define la configuración del robot mediante el vector RP, donde cada elemento indica si una junta es rotacional (0) o prismática (1). En este caso, [0 0 0] significa que el robot tiene tres articulaciones rotacionales, es decir, un manipulador 3R.

%SECCIÓN 3

```
%Creamos el vector de coordenadas articulares  
Q = [th1, th2, th3];  
disp('Coordenadas generalizadas');  
pretty (Q);
```

En esta sección se construye el vector de coordenadas generalizadas Q, que contiene las variables articulares del sistema.

%SECCIÓN 4

```
%Creamos el vector de velocidades generalizadas
Qp = diff(Q, t);
disp('Velocidades generalizadas');
pretty (Qp);
```

Se calculan las velocidades generalizadas derivando el vector Q respecto al tiempo. Como las articulaciones fueron definidas como funciones de t, el comando diff genera automáticamente las velocidades angulares

%SECCIÓN 5

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

Se determina el número de grados de libertad del robot usando el tamaño del vector RP.

%SECCIÓN 6

```
%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 1 respecto a 0
P(:, :, 2) = [l2*cos(th2); l2*sin(th2); 0];
%Matriz de rotación de la junta 1 respecto a 0
R(:, :, 2) = [cos(th2) -sin(th2) 0;
              sin(th2)  cos(th2) 0;
              0         0        1];

%Junta 3
%Posición de la junta 1 respecto a 0
P(:, :, 3) = [l3*cos(th3); l3*sin(th3); 0];
%Matriz de rotación de la junta 1 respecto a 0
R(:, :, 3) = [cos(th3) -sin(th3) 0;
              sin(th3)  cos(th3) 0;
              0         0        1];
```

Se define la geometría de cada articulación mediante su vector de posición local P y su matriz de rotación R. Como el robot es planar, todas las rotaciones se realizan alrededor del eje Z, por eso las matrices tienen la forma típica de rotación en el plano XY.

%SECCIÓN 7

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

```

Se inicializan las matrices de transformación homogénea tanto locales como globales, además de las matrices de rotación y vectores de posición respecto al marco inercial.

%SECCIÓN 8

```

for i = 1:GDL
    i_str = num2str(i);
    %Locales
    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);
    RO_inv(:, :, i) = transpose(RO(:, :, i));
    PO(:, :, i) = T(1:3, 4, i);
    %pretty(RO(:, :, i));
    %pretty(RO_inv(:, :, i));
    %pretty(PO(:, :, i));
end

```

Se ejecuta un ciclo que construye las matrices de transformación homogénea locales y luego las multiplica acumulativamente para obtener las transformaciones globales.

Matemáticamente, esto implementa la cinemática directa, ya que cada transformación global es el producto de la anterior por la transformación local actual.

%SECCIÓN 9

```

%Calculamos el jacobiano lineal de forma diferencial

```

```

disp('Jacobiano lineal obtenido de forma diferencial');
%Derivadas parciales de x respecto a th1, th2 y th3
Jv11= functionalDerivative(PO(1,1,GDL), th1);
Jv12= functionalDerivative(PO(1,1,GDL), th2);
Jv13= functionalDerivative(PO(1,1,GDL), th3);
%Derivadas parciales de y respecto a th1, th2 y th3
Jv21= functionalDerivative(PO(2,1,GDL), th1);
Jv22= functionalDerivative(PO(2,1,GDL), th2);
Jv23= functionalDerivative(PO(2,1,GDL), th3);
%Derivadas parciales de z respecto a th1, th2 y th3
Jv31= functionalDerivative(PO(3,1,GDL), th1);
Jv32= functionalDerivative(PO(3,1,GDL), th2);
Jv33= functionalDerivative(PO(3,1,GDL), th3);
%Creamos la matriz del Jacobiano lineal
jv_d=simplify([Jv11 Jv12 Jv13;
               Jv21 Jv22 Jv23;
               Jv31 Jv32 Jv33]);
pretty(jv_d);

```

Se calcula el Jacobiano lineal mediante el método diferencial, derivando cada componente de la posición final respecto a cada variable articular. Esto sigue directamente la definición matemática del Jacobiano como matriz de derivadas parciales.

%SECCIÓN 10

```

%Calculamos el jacobiano lineal de forma analítica
Jv_a(:,GDL) = PO(:, :,GDL);
Jw_a(:,GDL) = PO(:, :,GDL);
for k = 1:GDL
    if RP(k) == 0 %Casos: articulación rotacional
        %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

        %Para las juntas prismáticas
    elseif RP(k) == 1 %Casos: articulación prismática
        %
        try
            Jv_a(:,k)= RO(:,3,k-1);
        catch
            Jv_a(:,k)=[0,0,1];
        end
        Jw_a(:,k)=[0,0,0];
    end
end
Jv_a = simplify (Jv_a);
Jw_a = simplify (Jw_a);

```

Por último, se calcula el Jacobiano de forma analítica. Primero se inicializan las matrices J_{v_a} y J_{ω_a} , que corresponderán al Jacobiano lineal y angular. Luego, dentro del ciclo for se analiza cada articulación según su tipo usando el vector RP.

Si la articulación es rotacional ($RP(k)=0$), se aplica las siguientes fórmulas:

$$\mathbf{J}_{v_i} = \mathbf{Z}_{i-1} \times (\mathbf{O}_n - \mathbf{O}_{i-1}), \quad \mathbf{J}_{\omega_i} = \mathbf{Z}_{i-1}$$

En cambio, si la articulación es prismática ($RP(k)=1$), la interpretación cambia, ya que, una junta prismática no genera velocidad angular, solo traslación a lo largo de su eje

$$\mathbf{J}_{v_i} = \mathbf{Z}_{i-1}, \quad \mathbf{J}_{\omega_i} = \mathbf{0}$$