Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Puebla

Materia: Implementación de redes seguras

Clave: TC2036

Grupo: 501

Profesor: Alfredo García Suárez

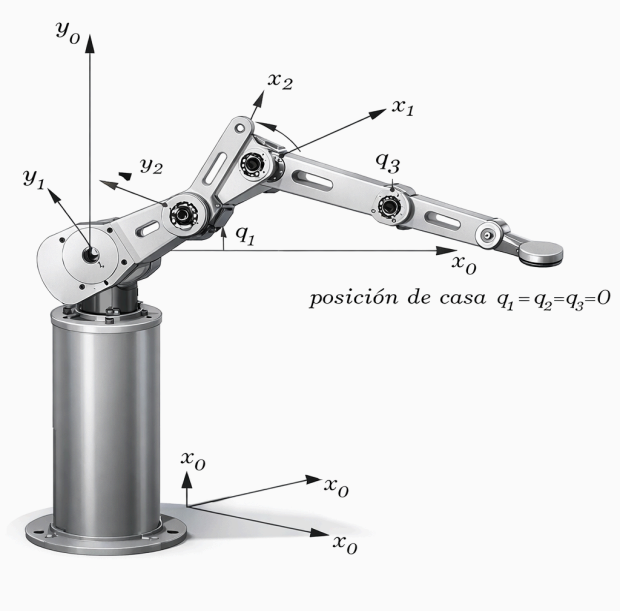
Actividad 1 (Velocidades Lineales y angulares) Robot Planar 3GDL

Alumnos
Omar Perez Dominguez | A01738306

16 de Febrero De 2026

Introducción: Modelado Cinemático y Análisis de Velocidades de un Robot de 3 GDLE

En la robótica la manipulación se fundamenta en la capacidad de describir matemáticamente la posición, orientación y movimiento de un mecanismo. En el siguiente trabajo, se realiza la extensión de un modelo robótico de dos grados de libertad (2 GDL) hacia un sistema de tres grados de libertad (3 GDL), empleando herramientas de computación simbólica en MATLAB.



Análisis del Movimiento y la Cinemática de Velocidades en 3 GDL

El análisis de las velocidades en un manipulador de tres grados de libertad comienza con la comprensión de la matriz Jacobiana, la cual actúa como el núcleo matemático que vincula dos mundos distintos: el espacio de las articulaciones y el espacio cartesiano. En términos sencillos, el Jacobiano es una matriz de transformación que traduce la velocidad a la que giran los motores (radianes por segundo) en la velocidad a la que realmente se desplaza y orienta la punta del robot en el mundo real. Al haber extendido el modelo de dos a tres grados de libertad, esta matriz se vuelve más compleja y completa, permitiendo que el robot no solo se mueva en un plano, sino que pueda operar en un volumen tridimensional, coordinando sus tres eslabones para seguir trayectorias precisas.

Dentro de este esquema, la velocidad lineal representa la rapidez con la que el actuador final cambia su posición de un punto de coordenadas a otro. Es lo que comúnmente entendemos como desplazamiento: qué tan rápido viaja la "mano" del robot a través del aire. En el desarrollo del código, esta velocidad se calcula derivando la posición final respecto al tiempo, lo que demuestra que no depende únicamente de la rapidez de los motores, sino también de la configuración física del robot. Por ejemplo, si los brazos están totalmente extendidos, un pequeño giro en la base producirá una velocidad lineal mucho mayor en la punta que si los brazos estuvieran encogidos, debido al incremento en el radio de giro.

Por otro lado, la velocidad angular es el componente que define qué tan rápido cambia la orientación del extremo del robot mientras se mueve. A diferencia de la lineal, que mide traslación, la angular se encarga de medir la rotación sobre los ejes del propio actuador. En un robot de 3 GDL, esta capacidad es fundamental, ya que permite que el robot mantenga una herramienta en una posición específica o que gire sobre una pieza con precisión quirúrgica. Mientras los dos primeros eslabones suelen encargarse de posicionar el robot en un lugar del espacio, el tercer grado de libertad potencia la capacidad de orientar el extremo, permitiendo que el Jacobiano angular determine cómo la suma de los giros de cada motor resulta en una rotación final combinada.

Pasos realizados en matlab para obtener la velocidad lineal y angular

Esta sección del código realiza la inicialización del entorno simbólico y la configuración del robot. En este bloque, se definen las variables articulares como funciones del tiempo para permitir el cálculo de derivadas, se establece que el sistema posee tres grados de libertad rotacionales mediante el vector RP, y se generan los vectores de posición y velocidad generalizada. Aquí se prepara la estructura.

```
%Limpieza de pantalla
clear all
close all
clc
%Declaración de variables simbólicas
syms th1(t) th2(t) th3(t) t l1 l2 l3
%Configuración del robot, 0 para junta rotacional, 1 para junta prismática
RP=[0 0 0];
%Creamos el vector de coordenadas articulares
Q= [th1, th2 th3];
disp('Coordenadas generalizadas');
pretty (Q);
%Creamos el vector de velocidades generalizadas
Qp= diff(Q, t);
disp('Velocidades generalizadas');
pretty (Qp);
%Número de grado de libertad del robot
GDL= size(RP,2);
GDL_str= num2str(GDL);
```

Este bloque realiza el modelado geométrico local de cada eslabón. En él se declaran los vectores de posición y las matrices de rotación individuales para las tres juntas. Utilizando trigonometría, el código define la longitud y la orientación de cada brazo por separado, traduciendo las propiedades físicas del robot en coordenadas matemáticas base.

```
%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];
%Creamos un vector de ceros
Vector_Zeros= zeros(1, 3);
```

Este bloque se encarga del ensamblaje de la cadena cinemática. En él, el ciclo for combina las matrices de rotación y posición para crear las matrices de transformación homogénea. Primero genera las matrices locales (A) de cada eslabón y luego las multiplica sucesivamente para obtener las matrices globales (T). Esto permite calcular la posición (PO) y orientación (RO) final del robot respecto a su base.

```
%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);
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
```

Este bloque realiza el análisis de velocidades mediante la obtención de la matriz Jacobiana. En él, se calculan los componentes del Jacobiano de dos formas: diferencial, derivando la posición respecto a los ángulos, y analítica, empleando productos cruz basados en la geometría del brazo. Finalmente, el código multiplica estas matrices por las velocidades de los motores para obtener la velocidad lineal (V) y angular (W) real del extremo del robot.

```
%Calculamos el jacobiano lineal de forma diferencial
disp('Jacobiano lineal obtenido de forma diferencial');
%Derivadas parciales de x respecto a th1 y th2
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 y th2
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 y th2
Jv31= functionalDerivative(PO(3,1,GDL), th1);
Jv32= functionalDerivative(PO(3,1,GDL), th2);
Jv33= functionalDerivative(PO(3,1,GDL), th3);
%Creamos la matríz del Jacobiano lineal
jv_d=simplify([Jv11 Jv12 Jv13;
               Jv21 Jv22 Jv23;
               Jv31 Jv32 Jv33]);
pretty(jv_d);
%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);
disp('Jacobiano lineal obtenido de forma analítica');
pretty (Jv_a);
disp('Jacobiano ángular obtenido de forma analítica');
pretty (Jw_a);
disp('Velocidad lineal obtenida mediante el Jacobiano lineal');
V=simplify (Jv_a*Qp');
pretty(V);
disp('Velocidad angular obtenida mediante el Jacobiano angular');
W=simplify (Jw_a*Qp');
pretty(W);
```

Resultado

Velocidad angular

```
Velocidad angular obtenida mediante el Jacobiano angular
/                 0                 \
|                                     |
|                 0                   |
|                                     |
| _________   _________   _________   |
|  d           d           d          |
| -- th1(t) + -- th2(t) + -- th3(t)   |
\ dt          dt          dt         /
```

Velocidad lineal

```
Velocidad lineal obtenida mediante el Jacobiano lineal
/ - #4 (l3 sin(#1) + l2 sin(#2)) - #5 (l1 sin(th1(t)) + l3 sin(#1) + l2 sin(#2)) - l3 #3 sin(#1) \
|                                                                                               |
|  #4 (l3 cos(#1) + l2 cos(#2)) + #5 (l1 cos(th1(t)) + l3 cos(#1) + l2 cos(#2)) + l3 #3 cos(#1)  |
|                                                                                               |
\                                          0                                                    /
```