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

Ximena Ortiz Gómez A01735100

16 de febrero del 2026

Fundamentación de Robótica

Profesor Alfredo García Suárez

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

Durante esta actividad se obtuvo el vector de velocidades lineales y angulares de un robot planar con 3 grados de libertad. En esta actividad se nos dió el caso de un robot que estaba formado por 3 articulaciones rotacionales y analizando ampliamente dicho robot podemos darnos cuenta que todo el movimiento ocurre en los ejes X y Y y cada una de las articulaciones rota alrededor del eje z. Es por ello que el objetivo de la actividad fue comprender cómo se modela la cinemática de un robot, partiendo de matrices de transformación homogénea hasta la obtención del Jacobiano y posteriormente obteniendo las velocidades angulares y lineales.

Como se mencionó con anterioridad, se trabajó con un robot planar de tres grados de libertad, por lo que se definieron los ángulos articulares como funciones del tiempo, ya que el robot se encuentra en movimiento. Esto permite derivarlos respecto al tiempo para obtener las velocidades angulares y lineales. Para poder obtener ambas velocidades primeramente se definieron las coordenadas como funciones del tiempo que representan los ángulos articulares de cada robot, en nuestro caso los definimos como q_1, q_2, q_3. Posteriormente se construyeron las matrices de transformación homogénea para cada grado de libertad, considerando que su matriz de rotación siempre será la misma porque el robot gira en torno a Z. Luego obtuvimos la matriz de traslación y para obtenerla se ocuparon relaciones trigonométricas para poder obtener x y y. Posteriormente se realizaron las transformaciones globales multiplicando las matrices homogéneas, y después se calculó el jacobiano mediante derivadas parciales del vector de posición.

Luego, obtuvimos el jacobiano angular considerando el eje de rotación de cada uno de los grados de libertad y finalmente se multiplicaron los jacobianos por las velocidades articulares para obtener velocidad angular y velocidad lineal. Para ejemplificar lo anterior, a continuación se muestran los apuntes realizados en clase, donde se realizó parte de este procedimiento matemáticamente para encontrar la matriz de traslación de un robot con solo 1 grado de libertad.

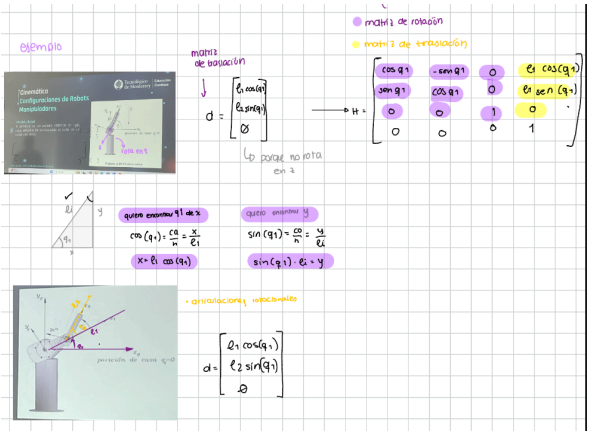

Imagen 1. Procedimientos de cálculos de matrices de rotación y traslación de un robot tipo péndulo

Dado que en esta actividad nuestro robot contaba con 3 grados de libertad, se tuvieron que realizar los cálculos y el procedimiento en Matlab para poder sacar la velocidad lineal y angular. A continuación se interpretan los resultados dados por Matlab en esta actividad.

```
Coordenadas generalizadas
(th1(t), th2(t), th3(t))

Velocidades generalizadas
/  d           d           d        \
| -- th1(t), -- th2(t), -- th3(t) |
\ dt          dt          dt       /
```

Imagen 2 coordenadas generalizadas y velocidades generalizadas

Primeramente analizando estos primeros resultados podemos decir que el modelo se construyó usando las coordenadas $\theta_1(t)$, $\theta_2(t)$ y $\theta_3(t)$, los cuales representan los ángulos articulares del robot en función del tiempo. Esto implica que el sistema es dinámico y permite obtener velocidades al derivar estos ángulos. Consecutivamente, las velocidades corresponden a $\frac{d}{dt}\theta_1(t)$, $\frac{d}{dt}\theta_2(t)$, $\frac{d}{dt}\theta_3(t)$. Estas velocidades representan la velocidad angular de cada articulación, es decir que tan rápido gira cada eslabón.

Posteriormente se obtuvieron las matrices A1, A2 y A3, las cuales representan las transformaciones entre eslabones. Cada una de las matrices contiene su matriz de rotación, la cual siempre es la misma porque el robot gira alrededor del eje z, y su matriz de traslación la cual se calcula como en la imagen 1.

```
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 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 local A3
/ cos(th3(t)), -sin(th3(t)), 0, l3 cos(th3(t)) \
|                                               |
| sin(th3(t)),  cos(th3(t)), 0, l3 sin(th3(t)) |
|                                               |
|      0,           0,      1,       0         |
|                                               |
\      0,           0,      0,       1        /
```

Imagen 3. Matrices de traslación local

Posteriormente con nuestro programa de Matlab obtuvimos las transformaciones globales las cuales representan la posición de cada grado de libertad respecto al sistema de coordenadas. Con estas matrices de transformación local podemos decir que el tercer grado de libertad no solo depende de su propio ángulo, sino de todos los grados de libertad anteriores. Por eso las orientaciones aparecen como sumas de ángulos.

```
Matriz de Transformación global T3
/ #2, -#1, 0, l1 cos(th1(t)) + l3 #2 + l2 cos(th1(t) + th2(t)) \
|                                                               |
| #1,  #2, 0, l1 sin(th1(t)) + l3 #1 + l2 sin(th1(t) + th2(t)) |
|                                                               |
|  0,  0,  1,                          0                        |
|                                                               |
\  0,  0,  0,                          1                       /
```

Imagen 4. Matriz de transformación global T3

Posteriormente, obtuvimos el jacobiano lineal. En nuestros resultado podemos observar que el jacobiano lineal cuenta con senos en la primera fila, cosenos en la segunda y ceros en la tercera, como podemos observar en la siguiente imagen. Con esto podemos decir que el movimiento del robot solo ocurre en los ejes X y Y.

```
Jacobiano lineal obtenido de forma diferencial
/ - l1 sin(th1(t)) - #1 - l2 sin(th1(t) + th2(t)), - #1 - l2 sin(th1(t) + th2(t)), -#1 \
|                                                                                      |
|  l1 cos(th1(t)) + #2 + l2 cos(th1(t) + th2(t)),   #2 + l2 cos(th1(t) + th2(t)),   #2 |
|                                                                                      |
\                    0,                                    0,                        0 /

where

   #1 == l3 sin(th1(t) + th2(t) + th3(t))

   #2 == l3 cos(th1(t) + th2(t) + th3(t))


Jacobiano lineal obtenido de forma analítica
/ - l1 sin(th1(t)) - #1 - l2 sin(th1(t) + th2(t)), - #1 - l2 sin(th1(t) + th2(t)), -#1 \
|                                                                                      |
|  l1 cos(th1(t)) + #2 + l2 cos(th1(t) + th2(t)),   #2 + l2 cos(th1(t) + th2(t)),   #2 |
|                                                                                      |
\                    0,                                    0,                        0 /
```

Imagen 5. Jacobianos obtenidos en este robot

Consecutivamente, con nuestro programa de Matlab obtuvimos el jacobiano angular, en el cual podemos observar en la imagen 6 que toda la velocidad angular está concentrada en el eje Z. Esto es muy coherente con nuestro robot analizado porque todas las articulaciones de este robot giran alrededor del eje Z. Además es por ello que también podemos observar que en la velocidad lineal en el eje z es cero, porque en ese plano no hay movimiento. La matriz de velocidad lineal nos indica que la velocidad lineal depende de la geometría y longitud del grado de libertad, los ángulos y las velocidades articulares.

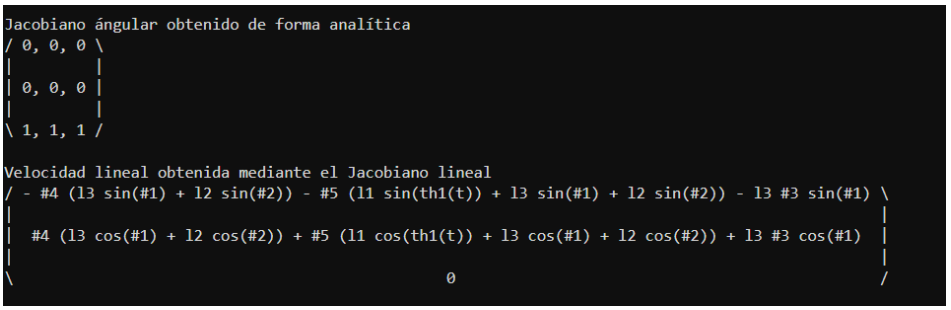

Imagen 6. Jacobiano angular y velocidad lineal

Finalmente obtuvimos la matriz de velocidad angular, en la cual podemos observar que toda esta velocidad angular se concentra en el eje Z, lo cual representa que la rotación del robot se está realizando en dicho eje.

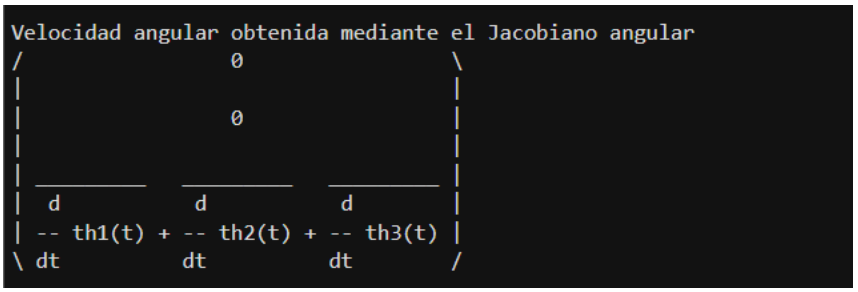

Imagen 7. Velocidad angular

Al observar los resultados obtenidos por nuestro programa de Matlab podemos concluir que los resultados son coherentes con lo visto en clase con un robot de 1 grado de libertad. Además podemos percatarnos de que la velocidad angular se concentra en el eje Z y que el que no exista componente en Z en la velocidad lineal confirma que el robot está restringido en dicho plano, por lo tanto representa el eje de rotación de los grados de libertad de nuestro robot

Explicación de código

```
%SECCIÓN 1
%Declaración de variables simbólicas
syms th1(t) th2(t) t l1 l2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 1: En esta sección se definen las variables para el análisis cinemático. Primeramente se define el ángulo 1 y el ángulo 2 como funciones del tiempo, también se define el tiempo y de igual manera se establecen las longitudes l1 y l2. Es por ello que las entradas son parámetros como lo son ángulos, tiempo y longitudes y las salidas son variables que nos ayudarán a hacer el modelo.

```
%SECCIÓN 2
%Configuración del robot, 0 para junta rotacional, 1 para junta prismática
RP=[0 0 0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 2: En esta sección se define el tipo de grado de libertad del robot, el valor 0 representa juntas rotacionales, es decir robots que tienen sus articulaciones de manera rotacional y el número 1 representa juntas prismáticas. Es por ello que la entrada es el tipo de juntas y la salida es un vector.

```
%SECCIÓN 3
%Creamos el vector de coordenadas articulares
Q= [th1, th2];
disp('Coordenadas generalizadas');
pretty (Q);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 3: En esta sección se crea el vector de coordenadas articulares, es decir las coordenadas generalizadas y posteriormente se imprimen. Por ende, la entrada de esta sección son las variables articulares simbólicas y la salida es el vector de coordenadas generalizadas.

```
%SECCIÓN 4
%Creamos el vector de velocidades generalizadas
Qp= diff(Q, t);
disp('Velocidades generalizadas');
pretty (Qp);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 4: En esta sección se obtienen las velocidades articulares derivando el vector anterior respecto al tiempo. Por ende esta sección genera el vector de velocidades generalizadas que son las derivadas temporales de los ángulos articulares. Es por ello que las entradas son los vectores Q y las salidas son el vector de velocidades generalizadas.

```
%SECCIÓN 5
%Número de grado de libertad del robot
GDL= size(RP,2);
GDL_str= num2str(GDL);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 5: En esta sección se indican los grados de libertad que tiene nuestro robot. El número de grados de libertad se calcula a partir del tamaño del vector RP y en este código este valor indica cuántas juntas tiene el robot y se utiliza posteriormente en otras partes del código. Por ende, la entrada de esta sección es el vector RP y la salida es el número de grados de libertad.

```
%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 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];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 6: En esta parte del código se definen las posiciones y las matrices de rotación de cada una de las juntas. Por ende, en la primera junta se define la posición de la junta respecto del origen y también se define su matriz de rotación. En la junta 2 de igual forma se define la posición pero esta vez de la junta 2 respecto a la 1 y también se representa la matriz de

rotación de dicha junta. Es por ello que podemos decir que estas matrices describen la geometría del robot. Sus entradas de esta sección son las longitudes y los ángulos de los grados de libertad y las salidas son las matrices de posición y rotación

```
%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);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 7: En esta sección se almacenan las matrices de transformación donde A son las transformaciones locales, B las transformaciones globales, PO son las posiciones respecto al marco base y RO son las matrices de rotación globales. También se define un vector de ceros para completar las matrices homogéneas. Es por ello que las entradas sonl as matrices P y R.

```
%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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 8: La sección 8 utiliza un ciclo for para construir las matrices de transformación. Primeramente se calcula la matriz homogénea local de cada junta, posteriormente se obtiene la transformación global acumulando los productos de matrices y finalmente se extraen las posiciones y rotaciones globales. Con esto podemos obtener la cinemática directa del robot. La entrada de esta sección son las matrices locales de rotación y posición y las matrices homogéneas locales y globales, posiciones y rotaciones.

```
%SECCIÓN 9
%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);
%Derivadas parciales de y respecto a th1 y th2
Jv21= functionalDerivative(PO(2,1,GDL), th1);
Jv22= functionalDerivative(PO(2,1,GDL), th2);
%Derivadas parciales de z respecto a th1 y th2
Jv31= functionalDerivative(PO(3,1,GDL), th1);
Jv32= functionalDerivative(PO(3,1,GDL), th2);

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

Sección 9: En esta sección se creó la matriz del jacobiano lineal derivando las coordenadas cartesianas con respecto a las variables articulares a través de derivadas parciales de la posición final. Por ende las entradas son la posición final y las salidas son el jacobiano lineal diferencial.

```
%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);
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);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sección 10: En esta sección se construye el jacobiano analítico considerando el tipo de junta definido en el vector denominado como RP. Primeramente se calcula la parte lineal mediante un producto cruz y la parte angular corresponde a la rotación. Una vez obtenidos los Jacobianos lineal y angular, se calculan las velocidades finales. Por ende las entradas son los jacobianos y las velocidades articulares y las salidas son la velocidad lineal y angular final.