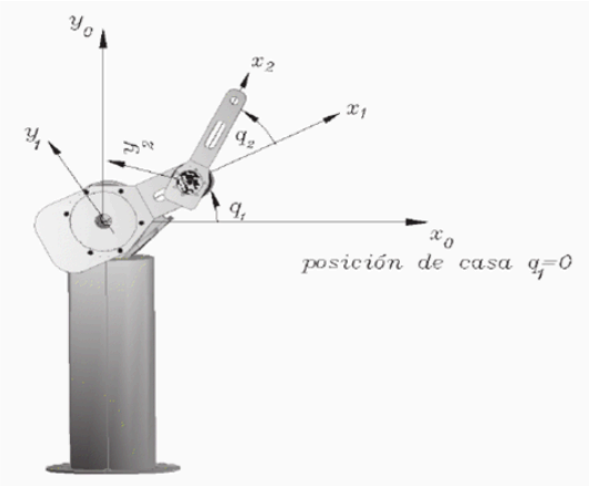
ACTIVIDAD 1: VELOCIDADES LINEALES Y ANGULARES

Víctor Manuel Vázquez Morales A01736352

ROBOT PLANAR DE 2 GDL

Parte de esta actividad es obtener y analizar las velocidades lineales y angulares de un robot planar de 2 grados de libertad, ya que sabiendo estos datos es posible predecir y conocer la posición y el estado del robot a futuro.



El sistema que analizaremos en esta actividad será el mostrado en la figura 1, el cual es un robot planar de 2 grados de libertad y está compuesto por dos articulaciones que rotan en torno al eje z. Para llegar a nuestro objetivo y obtener las velocidades lineales y angulares, implementamos el siguiente código de MATLAB.

Figura 1: Robot planar de 2 gdl

CÓDIGO DE MATLAB

Limpieza de pantalla y de variables:

```
clear all
close all
clc
```

Considerando que no contamos con un valor fijo para cada una de las variables involucradas, será necesario trabajar con variables simbólicas:

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

Ahora bien, crearemos un vector RP que contendrá la configuración del robot, en el cual agregaremos un 0 al vector si se trata de una junta rotacional y 1 si es una junta prismática. Para este caso, nuestro robot cuenta con dos juntas rotacionales:

```
RP = [0 0];
```

Coordenadas articulares

Procedemos a crear el vector de coordenadas articulares, el cual contendrá las variables de las que dependerá el estado o posición del robot:

```
Q = [th1 th2];
disp('Coordenadas articulares');
pretty(Q);
```

Respuesta de código de MATLAB

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

Observemos que las variables de las que depende la posición de nuestro robot son theta1 y theta2, que vienen siendo los ángulos de rotación para cada uno de los brazos que componen al robot.

Velocidades articulares

Ahora bien, recordemos que el objetivo de esta actividad es obtener las velocidades lineales y angulares. Para ello, es necesario obtener el vector de velocidades articulares:

```
Qp = diff(Q, t);
disp('Velocidades articulares')
pretty(Qp);
```

Respuesta de código de MATLAB

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

Para este caso, observemos que las velocidades articulares dependen igualmente de theta1 y theta2.

Grados de libertad del robot

Para continuar con el análisis de nuestro robot, será necesario conocer la cantidad de grados de libertad del mismo. A simple vista podemos darnos cuenta que nuestro robot cuenta con dos grados de libertad, esto debido a que su configuración se compone de dos juntas rotacionales. Sin embargo, con el fin de codificar y obtener las velocidades lineales y angulares más adelante, será necesario almacenar el número de grados de libertad en alguna variable.

```
GDL = size(RP, 2);
GDL_str = num2str(GDL);
```

Matrices homogéneas

Comenzaremos por encontrar la matriz de transformación homogénea en el efector final. Esta matriz se puede obtener analizando el extremo final con respecto al origen (al inicio del sistema). Para este caso podría resultar sencillo obtener dicha matriz analizando el sistema, esto debido a que solo contamos con 2 articulaciones. Sin embargo, el sistema podría contar con muchísimas más articulaciones, haciendo mucho más complejo el problema.

Afortunadamente, existe otra manera de atacar el problema, la cual consiste en multiplicar todas las matrices de transformación homogénea locales de cada articulación. Pero, ¿a qué se refiere una matriz de transformación homogénea local?, bien, pues esta matriz NO toma como referencia el origen de todo el sistema, si no el “extremo final” de la articulación previa:

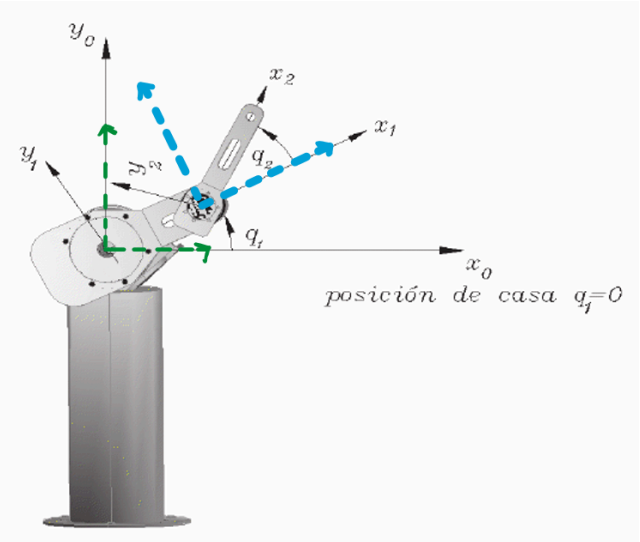


Figura 2: En este pequeño diagrama podemos observar como se analizarían cada una de las articulaciones de manera local. Nótese que para la primera articulación, el brazo toma como eje de referencia el del sistema, mientras que en el segundo, la articulación toma como referencia un nuevo plano cartesiano cuyo origen se encuentra justo al final de la primera articulación y que, además, su eje x coincide con el brazo de la primera articulación.

Recordemos que la matriz de transformación homogénea se compone de :

$$\begin{bmatrix} \text{Matriz de rotación} & \vdots & \text{Vector de Traslación} \\ \cdots & \vdots & \cdots \\ \mathbf{0}^T & & 1 \end{bmatrix}$$

En nuestro código, declararemos por separado cada uno de los elementos de esta matriz:

- Vectores de traslación (vistos de manera local):

```
%Posición de la junta 1 respecto a 0
P(:,:,1) = [l1*cos(th1);
            l1*sin(th1);
                     0];

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

- Matrices de rotación (analizados de manera local):

```
%Matriz de rotación de la articulación 1 respecto a 0
R(:,:,1) = [cos(th1) -sin(th1) 0;
            sin(th1) cos(th1)  0;
            0           0      1];
%Matriz de rotación de la articulación 2 respecto a 1
R(:,:,2) = [cos(th2) -sin(th2) 0;
            sin(th2) cos(th2)  0;
            0           0      1];
```

- Vector de ceros

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

Ahora bien, crearemos una matriz de 3D para almacenar las matrices de transformación homogénea locales:

```
%Inicializamos las matrices de transformación Homogénea locales
A(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
```

De igual forma, crearemos otra matriz para almacenar las matrices de transformación homogénea globales para cada una de las articulaciones:

```
%Inicializamos las matrices de transformación Homogénea globales
T(:,:,GDL) = simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
```

Esta matriz 3D contendrá en cada una de sus páginas la matriz de transformación homogénea global considerando la articulación actual en análisis y las previas a esta.

Finalmente, inicializamos los vectores de posición y las matrices de rotación, ambas vistas desde el marco de referencia inercial:

```
%Vectores de posición vistos desde el marco de referencia inercial
PO(:,:,GDL) = P(:,:,GDL);

%Matrices de rotación vistas desde el marco de referencia inercial
```

```
RO(:,:,GDL) = R(:,:,GDL);
```

Hasta el momento, únicamente hemos declarado las variables que contendrán las matrices de nuestro interés. Sin embargo, no hemos realizado el cálculo respectivo para obtener los datos de nuestro interés. Procedemos a declarar las matrices de transformación homogénea local para cada articulación y, de igual forma, iremos obteniendo la matriz global en cada una de las articulaciones hasta llegar a la del efector final:

A continuación, el código para obtener la matriz:

```
for i = 1:GDL
   i_str = num2str(i);

   %Declaración de las matrices de transformación local para cada
   %articulación, concatenando cada uno de sus elementos:
   disp(strcat('Matriz de Transformación local A', i_str));
   A(:,:,i) = simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
   pretty(A(:,:,i));

   %Calculamos la matriz de transformación homogénea global multiplicando
   %la matriz actual con las matrices de las articulaciones previas:
   try
       T(:,:,i) = T(:,:,i-1)*A(:,:,i);
   catch
       T(:,:,i) = A(:,:,i); %Para la primer articulación, la matriz global
                            %es equivalente a la local
   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 de PO
   %de la matriz de transformación homogénea global T(:,:,GDL);

   RO(:,:,i) = T(1:3,1:3,i);
   PO(:,:,i) = T(1:3,4, i);
   pretty(RO(:,:,i));
   pretty(PO(:,:,i));

end
```

Respuesta de código de MATLAB

Articulación 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 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       /
```

Observemos que en esta primera articulación la matriz de transformación local y la matriz de transformación global son exactamente las mismas. Esto se debe a que no hay articulaciones previas a esta.

Notemos de igual forma que la matriz de rotación obtenida indica que hay una rotación en el eje z únicamente en función al ángulo theta1 ya que, al ser la primera articulación, no hay más ángulos involucrados.

Por último, podemos identificar un vector de traslación que indica un movimiento tanto en el eje x como en el eje y provocado por la rotación de la articulación. Observemos igualmente que dichos valores del vector de traslación están en función de theta1.

Articulación 2:

```
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                      /
```

Para esta segunda articulación, la matriz de transformación local y la matriz de transformación global, evidentemente, no coinciden. Observemos que en el caso de la matriz local, únicamente está en función de l2 y theta2, mientras que en la matriz global toma en consideración todas las variables del sistema.

Notemos que la matriz de rotación obtenida indica que hay una rotación en el eje z en función de theta1 y theta2, considerando ya el ángulo de rotación de la articulación 1.

Finalmente, observemos que el vector de traslación indica movimiento tanto en x como en y. Además, este vector de traslación considera igualmente las variables de la articulación previa, ya que es respecto al origen del sistema.

Jacobiano de la matriz

Para calcular la velocidad angular y la velocidad lineal, debemos antes calcular el jacobiano lineal y angular. Para esto, lo haremos de manera analítica siguiendo las fórmulas mostradas a continuación:

Para articulaciones rotacionales:

- Jacobiano lineal. Se realiza el producto cruz entre la tercera columna de la matriz de rotación previa (Z_{i-1}) y el vector de posición del efector final respecto al origen menos el vector de posición previa:

$$Jv_i = Z_{i-1} \times (O_n - O_{i-1})$$

- Jacobiano angular. Es la tercer columna de la matriz de rotación previa:

$$J\omega_i = Z_{i-1}$$

Para articulación prismáticas:

- Jacobiano lineal. Es la tercer columna de la matriz de rotación previa:

$$Jv_i = Z_{i-1}$$

- Jacobiano angular. Es un vector de ceros:

$$J\omega_i = \vec{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

```
Jv_a(:,GDL)=PO(:,:,GDL);
Jw_a(:,GDL)=PO(:,:,GDL);

for k = 1:GDL
   if (((RP(k)==0))|(RP(k)==1))
   %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]; %No hay matriz de rotación previa se
                                 %obtiene la Matriz identidad
        end
    else
        %Para las articulaciones prismáticas
        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 angular obtenido de forma analítica');
pretty(Jw_a)
```

Respuesta de código de MATLAB

```
Jacobiano lineal obtenido de forma analítica
/ - 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           /

Jacobiano angular obtenido de forma analítica
/ 0, 0 \
|      |
| 0, 0 |
|      |
\ 1, 1 /
```

Observemos que el jacobiano lineal obtenido nos indica y reafirma nuevamente que existe únicamente movimiento en el eje x y en el eje y, pero no en z. Por otro lado, el jacobiano angular sugiere (como era de esperarse) que hay una rotación únicamente sobre el eje z.

Finalmente, notemos que para el eje x, el jacobiano se compone únicamente de senos y, para el eje y, se compone solo de cosenos. Esto se debe a que el jacobiano se obtuvo a partir de derivadas parciales del vector de traslación, el cual se compone de cosenos para el eje x y senos para el eje y, según las funciones trigonométricas.

Velocidades lineales y angulares

Finalmente, ahora que hemos calculado tanto el jacobiano angular como el jacobiano lineal, basta con multiplicarlos con el vector de velocidades articulares para obtener la velocidad lineal y la velocidad angular:

```
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);
```

```
Velocidad lineal obtenida mediante el Jacobiano lineal
/   _________                                  _________     \
|    d                                         d           |
| - -- th1(t) (l1 sin(th1(t)) + l2 #1) - l2 -- th2(t) #1  |
|   dt                                        dt           |
|                                                          |
|  _________                                 _________     |
|   d                                         d            |
|  -- th1(t) (l1 cos(th1(t)) + l2 #2) + l2 -- th2(t) #2    |
|  dt                                        dt            |
|                                                          |
\                            0                             /

where

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

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

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

Respuesta de código de MATLAB

Hemos llegado justo al final de nuestro programa, obteniendo resultados congruentes con los que esperábamos. Obtuvimos un vector de velocidades lineales que, como podemos observar, se ve afectado por todas las variables de nuestro sistema y que además, nos indica que al solo haber movimiento en x y y, entonces también únicamente tendremos velocidad lineal en estos ejes, obteniendo entonces una velocidad lineal nula en z.

Por otro lado, para el caso de las velocidades angulares, observamos que de igual manera se consideran todos los ángulos presentes en nuestro sistema y, además, nos indica que únicamente hay velocidad angular sobre el eje z debido a que solo existe rotación en este eje.