ACTIVIDAD 1. Velocidades Lineales y Angulares

Jesús Alejandro Gómez Bautista - A01736171

El siguiente live script contiene la solución a la actividad 1 y su correspondiente documentación de código, así como la explicación de los resultados obtenidos y de la implementación o codificación de ciertos elementos que permiten su funcionamiento conforme a lo visto durante clase.

ETAPA 0. Definiendo el entorno de trabajo y sus variables.

```
clear all
close all
clc
```

Explicación: Estas primeras líneas de código dentro del programa nos permiten limpiar el entorno de trabajo, cerrar las figuras preexistentes (en caso de existir) y limpiar la ventana de comandos para evitar sobrescrituras o errores en caso de existir elementos compartidos con otros códigos.

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

Explicación: Para realizar cálculos simbólicos diferenciales, es necesario emplear variables simbólicas que representan valores numéricos con los que aún no contamos. Declarar estas variables como simbólicas nos permite manipularlas y resolver las ecuaciones.

ETAPA I. Grados de libertad

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

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

```
Cordenadas articulares
```

```
pretty(Q);
```

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

Explicación: La línea 7 define a un vector para representar el tipo de junta que el robot presenta, en este caso ambas juntas son de tipo rotacional. Las líneas de código a continuación nos permiten representar las coordenadas rotacionales (línea 10) del robot de manera legible para el usuario, con el fin de conocer la ubicación actual de las mismas (línea 12).

```
% Creamo el vector de velocidades articulares
```

```
Qp = diff(Q, t); % Utilizo diff para derivadas cuya variable de referencia no depende de otra:
disp('Velocidades articulares')
```

```
Velocidades articulares
```

```
pretty(Qp);
```

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

Análisis de resultados: Estas son las derivadas con respecto al tiempo de los ángulos presentes en el robot, dicho de otra forma, representan las velocidades angulares de la junta, lo que es igual a que tan rápido cambian los ángulos de las juntas con respecto al tiempo.

```
% Número de grados de libertad del robt
GDL = size(RP,2) % Siempre se coloca 2, ya que indica la dimensión de las columnas
```

```
GDL = 2
```

```
GDL_str = num2str(GDL); % Convertimos el valor numérico a una cadena de carácteres tipo string
```

Explicación: Las primeras 3 líneas de esta sección se encargan de calcular la derivada del vector Q con respecto al tiempo, resultando así nuestro vector de velocidades articulares expresado de manera más sencilla mediante pretty. A continuación, a través de la función size en la variable GDL (Grados De Libertad), calculamos el número de grados de libertad del robot, el cual es igual al número de columnas del vector RP. La línea debajo hace la conversión de este valor numérico a uno tipo string.

Articulación 1

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

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

Explicación: P(:,:,1) se encarga de calcular la junta a partir de la longitud del primer enlace del robot (l1) y el ángulo de la primera junta. El cálculo emplea trigonometría y genera un resultado con 2 dimensiones (aunque existe un z con valor 0). En cuanto a R(:,:,1) se trata de una matriz empleada para transformar las coordenadas de un punto de la junta 1 al sistema de coordenadas de la junta 0, en este caso consideramos la rotación en el eje de z.

Articulación 2

```
% Posición de la junta 2 respecto a 1
P(:,:,2) = [l2*cos(th2); %% Solo th2 por que las transformadas de derivación homogéneas locales
```

```
            l2*sin(th2);
                     0]; % Vector de posición indexado por página

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

Explicación: Si bien tanto la matriz de posición como la de rotación presentan el mismo funcionamiento que en la articulación 1, hay algo distintivo particularmente en la matriz de posición. Si bien cada enlace cuenta con un respectivo ángulo de rotación, el segundo, al estar conectado al final del primer enlace provoca que su orientación y posición dependan tanto de su propio ángulo como el del primer enlace, haciendo que la suma de th1 y th2 sea la rotación total de la segunda articulación, la cual influye también en la velocidad final.

ETAPA II. Transformaciones homogéneas

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

Explicación: Este vector es necesario para completar las matrices de transformación homogénea que se presentarán más adelante.

```
% Inicializamos las matrices de transformación homogeneas locales (la de
% cada junta)
A(:,:,GDL) = simplify([R(:,:,GDL) P(:,:,GDL); Vector_zeros 1]);
```

Explicación: Se crea la matriz de transformación homogénea local para la junta del robot, la cual no es otra cosa que la representación de la rotación (R) y traslación que se le aplica a un punto (P).

```
% Inicializamos las matrices de transformación homogeneas globales (con respecto a la referenci
T(:,:,GDL) = simplify([R(:,:,GDL) P(:,:,GDL); Vector_zeros 1]);
```

Explicación: Basada en la anterior pero ahora de manera global con respecto a una referencia, transformando así el sistema de referencias de la junta al sistema global.

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

Explicación: Este vector será requerido para transformar los puntos desde el sistema de coordenadas local de la junta al del sistema de coordenadas global.

```
% Inicializamos los vectores de rotación vistos desde el marco de
% referencia inercial
RO(:,:,GDL) = R(:,:,GDL);
```

Explicación: Similar a la línea anterior, este vector será empleado para informar sobre la orientación de la junta con respecto al sistema de coordenadas global.

```
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); % Obtienes la global de la parte del sistema a partir d
    catch
        T(:,:,i) = A(:,:,i); % Caso específico cuando i = 1 nos marcaría error en try
    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 traslación "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);
    disp('Matriz de rotación')
    pretty(RO(:,:,i));
    disp('Vector de traslación')
    pretty(PO(:,:,i));
end
```

```
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       /
Matriz de rotación
/ cos(th1(t)), -sin(th1(t)), 0 \
|                              |
| sin(th1(t)),  cos(th1(t)), 0 |
|                              |
\      0,            0,      1 /
Vector de traslación
/ l1 cos(th1(t)) \
|                |
```

```
| l1 sin(th1(t)) |
|                |
\        0       /
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))
Matriz de rotación
/ 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 /
Vector de traslación
/ l1 cos(th1(t)) + l2 cos(th1(t) + th2(t)) \
|                                          |
| l1 sin(th1(t)) + l2 sin(th1(t) + th2(t)) |
|                                          |
\                    0                     /
```

Explicación: El anterior ciclo for tiene la tarea de recorrer cada grado de libertad del robot y calcular las matrices y vectores descritos en él. En primer lugar, calcula la matriz de transformación homogénea local para cada junta. En segunda instancia, se calcula la matriz de transformación global a cada una de las juntas, si esta es igual a la primera, entonces esta matriz será igual a la local. Finalmente, extraemos la matriz de rotación y el vector de traslación de la matriz de transformación homogénea global, ambas se imprimen en el entorno.

Análisis de resultados: Las diferencias en estas matrices radican en lo que cada una representa: las matrices A locales describen la posición y orientación de las juntas en relación con la junta previa, mientras que las matrices T globales describen la posición y orientación de las juntas en relación con la base del robot. Además, conforme te desplazas a través del robot (de la junta 1 a la junta 2), las matrices se vuelven más complejas debido a que la posición y orientación de una junta están influenciadas por todas las juntas anteriores.

ETAPA III. Jacobianos (diferencial y analítica)

Jacobiano de forma Diferencial

```
disp('Jacobiano lineal obtenido de forma diferencial')
```

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 matriz del Jacobiano lineal
jv_d = simplify([Jv11, Jv12; ...
                 Jv21, Jv22; ...
                 Jv31, Jv32]);
pretty(jv_d);
```

```
/ - 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            /
```

Explicación: Con respecto al cálculo de jacobianos del robot, existen dos formas de hacerlo, diferencial y analítica, sin embargo, la primera de ellas solo nos permitirá calcular el jacobiano lineal, y para fines de esta actividad necesitaremos del angular también, por lo que esta no será una opción viable, aunque más adelante haremos el cálculo tanto para el lineal como el angular de manera analítica. En cuanto a esta forma de cálculo, tenemos que esta consiste en calcular la derivada de las coordenadas parciales (en x, y, z) de los extremos del robot con respecto a los ángulos. Tras esto, se crea una matriz de 3x2 que no es otra cosa que el Jacobiano lineal del robot.

Análisis de resultados: Las entradas de la matriz son las derivadas parciales de las coordenadas del extremo del robot con respecto a los ángulos de las juntas. Las entradas `0` en la última fila indican que no hay movimiento en la dirección z, lo que sugiere que nuestra articulación del robot se mueve en un plano 2D.

Jacobiano de forma analítica

```
% Calculamos el jacobiano lineal y angular de forma analítica
% Inicializamos jacobianos analíticos (lineal y angular)
Jv_a(:,GDL) = PO(:,:,GDL); % Velocidad lineal
Jw_a(:,GDL) = PO(:,:,GDL); % Velocidad angular
```

Explicación: Como se mencionó arriba, volveremos a calcular el jacobiano debido a la limitante de la forma diferencial, para lo cual declararemos tanto el jacobiano lineal como el angular en el extremo del robot.

```
for k = 1:GDL
    if ((RP(k)==0)|(RP(k)==1)) % Casos: articulación rotacional y prismática

        % 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 % Casos específicos
            Jv_a(:,k) = cross([0,0,1], PO(:,:,GDL));% Matriz de rotación de 0 con respecto a 0
            Jw_a(:,k) = [0,0,1]; % Si no hay matriz de rotación previa se obtiene la matriz ide
        end
    else
        % Para las articulaciones prismáticas
        try
            Jv_a(:,k) = RO(:,3,k-1);
        catch % Casos específicos
            Jv_a(:,K) = [0,0,1]; % Si no hay matriz de rotación previa se obtiene la matriz ide
        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');
```

```
Jacobiano lineal obtenido de forma analítica
```

```
pretty(Jv_a);
```

```
/ - 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           /
```

```
disp('Jacobiano ángular obtenido de forma analítica')
```

```
Jacobiano ángular obtenido de forma analítica
```

```
pretty(Jw_a)
```

```
/ 0, 0 \
|      |
| 0, 0 |
|      |
\ 1, 1 /
```

Explicación: En este ciclo for, se recorrerá cada grado de libertad del robot, no sin antes verificar que tipo de articulación es. Si es una articulación rotacional, se calcularán los jacobianos lineales y angulares empleando la fórmula de producto cruz junto a la columna 3 de la matriz de rotación. En caso de ser prismática, el jacobiano

lineal será equivalente a la tercera columna de la matriz de rotación, y el angular un vector de ceros. Lo anterior finalmente se imprime en la consola de forma simplificada.

Análisis de resultados (Jacobiano lineal): Este jacobiano ha sido previamente calculado mediante el método diferencial. Aunque las entradas parecen diferentes debido a la simplificación y reorganización aplicada por el comando pretty, ambas matrices describen la misma relación entre las velocidades articulares y las velocidades lineales del extremo del robot.

Análisis de resultados (Jacobiano angular): El jacobiano angular nos sugiere que el robot está rotando alrededor del eje z, pero no existe rotación alguna tanto en x como en y, lo cual es consistente con un movimiento en un plano bidimensional. La magnitud de las velocidades angulares en la junta nos indica que está rotando de manera constante.

ETAPA IV. Velocidades

```
disp('Velocidad lineal obtenida mediante el Jacobiano Lineal')
```

```
Velocidad lineal obtenida mediante el Jacobiano Lineal
```

```
V = simplify(Jv_a*Qp');
pretty(V);
```

```
/   _________                                  _________     \
|    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))
```

```
disp('Velocidad angular obtenida mediante el Jacobiano Angular')
```

```
Velocidad angular obtenida mediante el Jacobiano Angular
```

```
W = simplify(Jw_a*Qp');
pretty(W);
```

```
/          0          \
|                     |
|          0          |
|                     |
| _________   _______ |
|  d          d       |
```

```
| -- th1(t) + -- th2(t) |
\ dt          dt        /
```

Explicación: En esta sección nos centraremos fundamentalmente en el cálculo de velocidades angulares y lineales en el extremo a partir de los jacobianos previamente calculados. Para el cálculo de la velocidad lineal multiplicaremos el jacobiano lineal por el vector de velocidades articulares. Para la velocidad angular es exactamente lo mismo, cambiando el jacobiano lineal por el angular.

Análisis de resultados (Velocidad lineal): La primera fila de la matriz representa la velocidad en dirección de la x, la cual depende de las velocidades angulares y de junta y de los ángulos dadas por diferenciales y elementos trigonométricos que relacionan los ángulos y las longitudes de esas partes, particularmente el seno de los angulos junto a las longitudes. La segunda fila de la matriz representa la velocidad en dirección de la y, la cual, al igual que la x, depende de los mismos elementos con la diferencia de que consideramos el coseno en lugar del seno como lo hacía x. No obstante, la tercera fila indica que no hay movimiento en z.

Análisis de resultados (Velocidad angular): Esta matriz indica que el extremo del robot está rotando sobre el eje z a una velocidad dada por la suma de las velocidades angulares de la junta 1 y 2.