Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Puebla

Implementación de robótica inteligente (Gpo 501)

Actividad 1.4 (Trayectorias en lazo abierto)

Alumno

José Diego Tomé Guardado A01733345

Pamela Hernández Montero A01736368

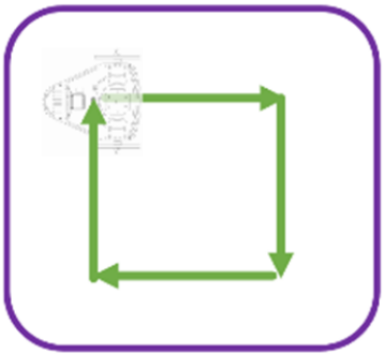
Victor Manuel Vázquez Morales A01736352

Fernando Estrada Silva A01736094

Fecha de entrega

Miércoles 17 de Abril de 2024

1. Código implementado para trayectoria de un cuadrado



Como podemos observar primeramente vamos a necesitar hacer una trayectoria con la figura de un cuadrado que tenga la finalidad de poder avanzar e ir girando dependiendo de la longitud que queramos para el cuadrado. Para ello vamos a ver necesario sobre todo el poder manipular correctamente la velocidad lineal y velocidad angular como lo explicaremos a continuación en el código.

Implementación en matlab

Primeramente se tienen las instrucciones para poder borrar todas las variables del espacio de trabajo y también para cerrar las figuras abiertas y finalmente limpiar la ventana de comandos. Posteriormente tenemos el tiempo de simulación qué es muy importante debido a que va a expresar los cambios que haremos en el vector de la velocidad angular, la relación es que el número de elementos en el vector `w` debe ser igual al número de muestras qué se toman durante el tiempo de simulación. Cada uno de los elementos `w` va a corresponder a la velocidad angular de referencia en un momento en específico de la simulacion determinado por el vector de tiempo `t`. **`w = [0, pi/2, 0, pi/2, 0,pi/2, 0, pi/2, 0, pi/2];`**

También tenemos el tiempo de muestreo en segundos y el vector de tiempo qué va desde 0 a tf con incrementos de ts.

```
clear
close all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% TIEMPO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tf = 9;              % Tiempo de simulacion en segundos (s)
ts = 1;             % Tiempo de muestreo en segundos (s)
t = 0: ts: tf;        % Vector de tiempo
N = length(t);        % Muestras
```

Tenemos las posiciones del robot definidas en x1,y1 donde también tendremos la orientación, es decir phi que será el ángulo en que se esté efectuando. Teniendo en 0 todo para indicar que empezará desde el origen.

```
%%%%%%%%%%%%%%%%%%%%%%%% CONDICIONES INICIALES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x1 = zeros (1,N+1);  % Posición en el centro del eje que une las ruedas
(eje x) en metros (m)
y1 = zeros (1,N+1);  % Posición en el centro del eje que une las ruedas
(eje y) en metros (m)
phi = zeros(1, N+1); % Orientacion del robot en radianes (rad)
x1(1) = 0;     % Posicion inicial eje x
```

```
y1(1) = 0;   % Posicion inicial eje y
phi(1) = 0;   % Orientacion inicial del robot
```

Aqui comenzaremos por establecer las variables de las posiciones del punto de control del robot en el tiempo inicial qué va a coincidir con la posicion inicial del robot. Se van a inicializar como vectores de 0.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% PUNTO DE CONTROL %%%%%%%%%%%%%%%%%%%%%%%%%%%%
hx = zeros(1, N+1);  % Posicion en el punto de control (eje x) en metros
(m)
hy = zeros(1, N+1);  % Posicion en el punto de control (eje y) en metros
(m)
hx(1) = x1(1); % Posicion en el punto de control del robot en el eje x
hy(1) = y1(1); % Posicion en el punto de control del robot en el eje y
```

- **Velocidades lineales y angulares**

```
%%%%%%%%%%%%%%%%%%%%%%% VELOCIDADES DE REFERENCIA %%%%%%%%%%%%%%%%%%%%%%%%%
u = 1.5*ones(1,N); % Velocidad lineal de referencia (m/s)
w = [0, pi/2, 0, pi/2, 0,pi/2, 0, pi/2, 0, pi/2]; % Velocidad angular de
referencia (rad/s)x
```

Tenemos como primero la velocidad lineal en la que vamos a establecer un 1.5 m/s para todos los instantes de tiempo que también se refleja como los metros que avanzará en la simulación en 3D del puzzlebot. La función de ones crea el vector con longitud N que se multiplica por 1.5 para obtener el vector con valores de velocidades lineales y pueda efectuar los movimientos.

La secuencia de `[0, pi/2, 0, pi/2, 0,pi/2, 0, pi/2, 0, pi/2];` está diseñado para que pueda hacer los giros de 90 grados en cada paso de tiempo, esencialmente el robot sigue un recorrido de un cuadrado. Primero avanza en línea recta durante un segundo `0 rad/s` y luego hace un giro 90 grados en sentido horario `pi/2 rad/s` luego sigue avanzando y hace un giro hasta que al final efectua un giro para quedar finalmente en la posición correcta.

```
%%%%%%%%%%%%%%%%%%%%%%%% BUCLE DE SIMULACION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k=1:N

   phi(k+1)=phi(k)+w(k)*ts; % Integral numérica (método de Euler)

   %%%%%%%%%%%%%%%%%%%% MODELO CINEMATICO %%%%%%%%%%%%%%%%%%%%%%%%

   xp1=u(k)*cos(phi(k+1));
   yp1=u(k)*sin(phi(k+1));
   x1(k+1)=x1(k) + xp1*ts ; % Integral numérica (método de Euler)
```

```
    y1(k+1)=y1(k) + yp1*ts ; % Integral numérica (método de Euler)


    % Posicion del robot con respecto al punto de control
    hx(k+1)=x1(k+1);
    hy(k+1)=y1(k+1);
end
```

En este bucle de simulacion simplemente utilizaremos de un ciclo for para poder hacer correctamente la simulación del movimiento del robot, se actualizan las posiciones y también las orientaciones que tendra el robot con un modelo cinematico y también el método de integración de Euler. Calculando la posición del punto de control en cada paso de tiempo.

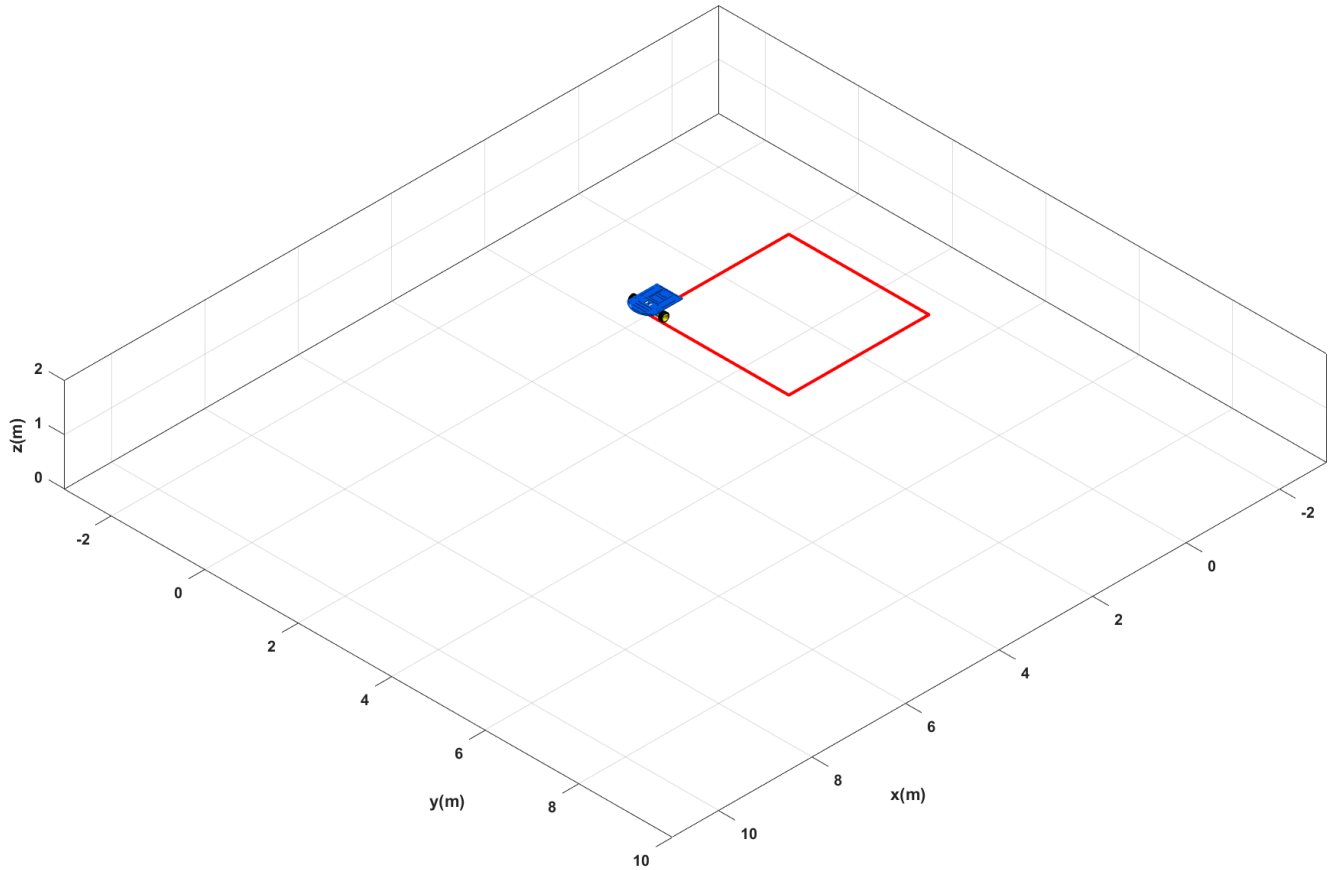
- **Simulación en 3D**

Ahora tendremos la simulación en 3d donde basicamente tendremos toda la configuracion de la escena y también del dibujo del robot. Graficando en su posicion inicial y trazando la trayectoria del punto de control, así como también trazar las trayectorias con una línea roja y un bucle ciclo for para poder efectuar su movimiento.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% SIMULACION VIRTUAL 3D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% a) Configuracion de escena

scene=figure;  % Crear figura (Escena)
set(scene,'Color','white'); % Color del fondo de la escena
set(gca,'FontWeight','bold') ;% Negrilla en los ejes y etiquetas
sizeScreen=get(0,'ScreenSize'); % Retorna el tamaño de la pantalla del
computador
set(scene,'position',sizeScreen); % Congigurar tamaño de la figura
camlight('headlight'); % Luz para la escena
axis equal; % Establece la relación de aspecto para que las unidades de
datos sean las mismas en todas las direcciones.
grid on; % Mostrar líneas de cuadrícula en los ejes
box on; % Mostrar contorno de ejes
xlabel('x(m)'); ylabel('y(m)'); zlabel('z(m)'); % Etiqueta de los eje

view([135 35]); % Orientacion de la figura
axis([-3 11 -3 10 0 2]); % Ingresar limites minimos y maximos en los ejes x
y z [minX maxX minY maxY minZ maxZ]

% b) Graficar robots en la posicion inicial
scale = 4;
MobileRobot_5;
H1=MobilePlot_4(x1(1),y1(1),phi(1),scale);hold on;

% c) Graficar Trayectorias
H2=plot3(hx(1),hy(1),0,'r','lineWidth',2);
```

```
% d) Bucle de simulacion de movimiento del robot
step=1; % pasos para simulacion
for k=1:step:N
   delete(H1);
   delete(H2);

   H1=MobilePlot_4(x1(k),y1(k),phi(k),scale);
   H2=plot3(hx(1:k),hy(1:k),zeros(1,k),'r','lineWidth',2);

   pause(ts);
end
```

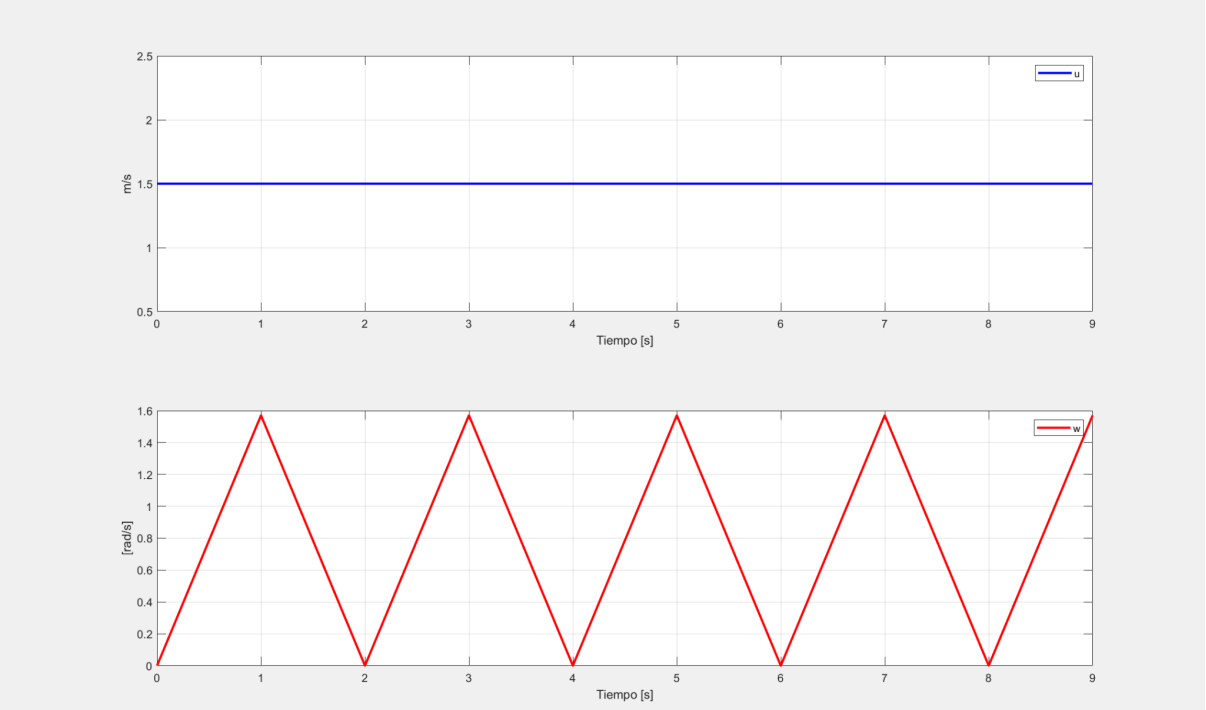
En este caso vamos a observar cómo se hace la simulación en 3D donde vemos como se hizo el movimiento del robot, en este caso pusimos como la velocidad lineal 1.5 m/s que basicamente es la velocidad que va a tener y además se ve reflejado con la distancia que recorre en cada lado del cuadrado. Como la velocidad angular vamos a tener lo siguiente que ya indicamos arriba como funciona `w = [0, pi/2, 0, pi/2, 0,pi/2, 0, pi/2, 0, pi/2];` qué primero avanza, luego gira 90 grados y así respectivamente, teniendo 5 avances en línea recta y efectuando 5 giros de 90 grados.



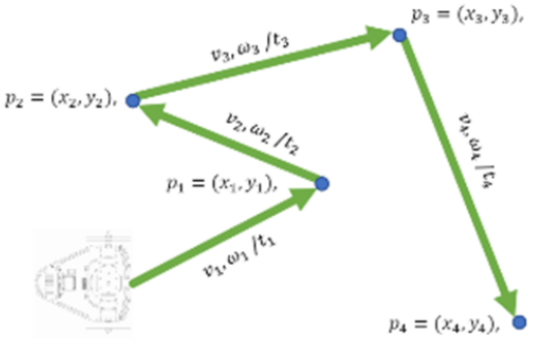
- **Gráficas de velocidades**

Podemos ver primeramente la gráfica de u que es la velocidad lineal donde observamos que tenemos una línea recta en 1.5 en el eje y que va a indicar la velocidad lineal que es constante en 1.5 m/s durante todo el tiempo de la simulación lo que hace que concuerde con la definición de u como un vector constante de 1.5 en cada muestra de tiempo.

De otra forma para la gráfica w que es la velocidad angular podemos ver que tenemos reflejados unos picos que van a representar este cambio de `0` a `pi/2 rad/s` que indica que el robot girará 90 grados en sentido horario. Esto es correcto debido a que en el vector que declaramos en w se alternan estos valores de 0 a pi/2 para poder lograr la trayectoria correcta del cuadrado. Finalmente las caídas van a indicar que la velocidad angular vuelve a 0 lo que concuerda con que el robot deje de girar y avance en línea recta.



2. Código implementado para trayectoria con puntos



Para esta segunda trayectoria se utilizará el mismo método implementado anteriormente. Sin embargo, para este caso se trata de trayectoria irregular la cual, en términos generales consiste en modificar la consistencia de los vectores proporcionados, considerando el tiempo de muestreo asignado al principio del programa.

En esta trayectoria se utiliza básicamente el mismo programa implementado anteriormente. Sin embargo, cuenta con ciertas variaciones.
La lógica consiste en el mismo método, tratándose de una serie de coordenadas en forma (v,w), las cuales representan la velocidad lineal y angular respectivamente. Se trata de 2 arreglos distintos en los cuales se almacenan ambas velocidades y determinan si el robot debe girar o desplazarse para cada segmento de la trayectoria.

En esta primera sección se definen los arreglos que representan el tiempo de simulación y muestreo. Es importante que para trayectorias como estas el tiempo de muestreo se mantenga en 1 para hacer énfasis en líneas rectas, además de ahorrar carga computacional.

```
tf = 8;              % Tiempo de simulacion en segundos (s)
ts = 1;             % Tiempo de muestreo en segundos (s)
t = 0: ts: tf;       % Vector de tiempo
N = length(t);       % Muestras
```

Dadas las condiciones iniciales, se mantiene el robot en (0,0) orientación 0

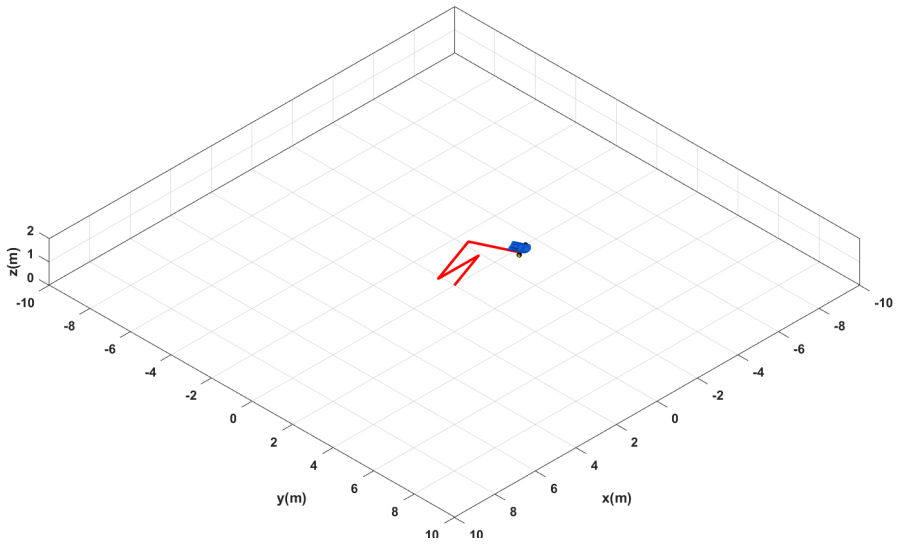
```
phi = zeros(1, N+1); % Orientacion del robot en radianes (rad)
x1(1) = 0;     % Posicion inicial eje x
y1(1) = 0;    % Posicion inicial eje y
phi(1) = 0;    % Orientacion inicial del robot
```

Los siguientes arreglos corresponden a la cinemática del robot. Como bien se ha mencionado, se trata de 2 arreglos independientes que prácticamente describen una coordenada para el robot.

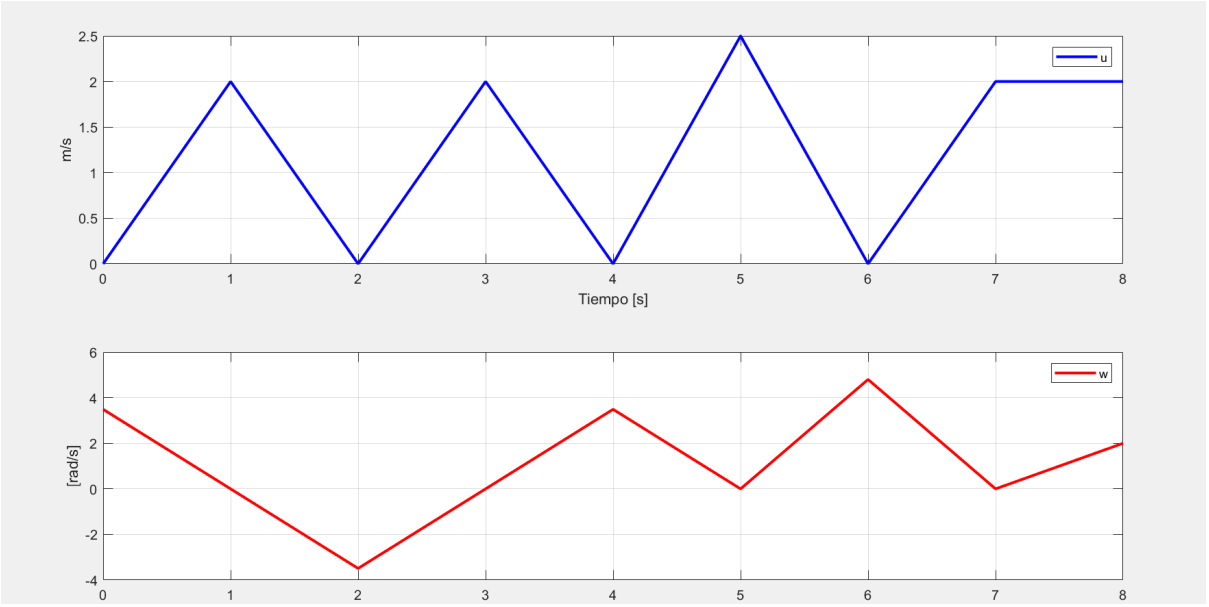
Por ejemplo, considerando que el primer movimiento del robot es un giro sobre su eje z, la velocidad angular tiene asignado un valor de 3.48, mientras que la lineal se mantiene en 0. Esto significa que el robot sólo debe girar sin avanzar, para el siguiente paso se invierte, solo avanza pero no hay velocidad angular. Lo mismo sucede para cada segmento de la trayectoria.

```
u = [0,    2,  0,    2, 0,    2.5,   0,   2, 2];
w = [3.49, 0, -3.49, 0, 3.49, 0,  4.8, 0, 2]; % Velocidad angular de
referencia (rad/s)x
```

traye A continuación se muestra el resultado obtenido con la simulación. Como se demostró, se grafican 4 trayectorias independientes que se construyen a partir de los arreglos v y w respectivamente.



Por otro lado, las velocidades se comportan de manera similar. Para la lineal, se tienen picos que representan la desaceleración del robot, variando entre una velocidad alcanzada y 0, por ello se obtiene una señal triangular. Lo mismo sucede con la angular, la magnitud del ángulo depende de que tan cerrado es el giro y tiene estados en los que es 0 que es cuando el robot avanza para posteriormente cambiar de dirección.


u
m/s
Tiempo [s]
w
[rad/s]