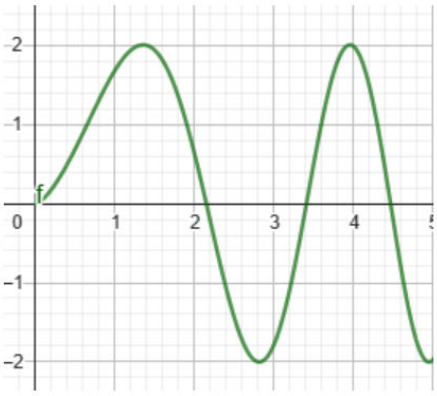
Actividad 1.5: Evaluación

José Diego Tomé Guardado A01733345

- **Primera trayectoria**

X = [0 a 5]

F(x) = 2*sen (x^(1.5))



Para la primera parte tendremos esta onda senoidal por cuestiones de tiempo y sobre todo de que necesitamos tener mucho muestreo debido a que se buscan generar las ondas senoidales, tendremos que crear unas funciones y moverle al bucle de simulacion para que se pueda formar correctamente por lo que explicaremos a continuacion como lo logramos de una forma para poder tener una corrección, sobre todo tomar en cuenta la función propuesta para que fuera más sencillo graficarla.

- **Primera trayectoria**

Variables para el tiempo de simulación y muestreo, vector de tiempo y muestras

Primeramente tenemos el tiempo de simulacion en segundos que lo definimos en 7 ya que podemos ver que con este valor se ajusta correctamente para el tiempo en que se termina la grafica, además de poder darle un tiempo de muestreo de 0.05 que fue el que mejor se ajusto para poder formar las ondas correctamente y de esta forma cada muestreo se tuviera no tan abrupto, debido a que si lo subia se tenian las ondas pero las bajadas y las subidas de la grafica eran muy abruptas y no se tenia un buen trazado de estas.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% TIEMPO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tf = 7;             % Tiempo de simulación en segundos (s)
ts = 0.05;           % Tiempo de muestreo en segundos (s)
t = 0: ts: tf;      % Vector de tiempo
N = length(t);      % Muestras
```

Función para la onda

Tenemos que definir el rango de x que ya se nos habia dado antes con un vector creado que va a tener los valores desde 0 hasta a con un total de N puntos que son las muestras ya definidas al principio, siguiendo se calcularan los valores de la funcion senoidal para cada valor del rango de x y se define la funcion senoidal que nos propusieron: F(x) = 2*sen (x^(1.5)) y se almacenan los valores en un vector para poder hacer el trazado continuamente. Finalmente se calcula la derivada de la funcion senoidal para poder generar una estimacion de la pendiente de la funcion en cada punto que se tenga del rango de x y se almacenan en un vector el de dF_x.

```
%%%%%%%%%%%%%%%%%%%%%%%% DEFINICION DE LA FUNCIÓN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Definir el rango de x
a = 5;
x_range = linspace(0, a, N);

% Calcular los valores de la función senoidal
F_x = 2 * sin(x_range.^1.5);

% Calcular la derivada
dF_x = gradient(F_x, x_range);
```

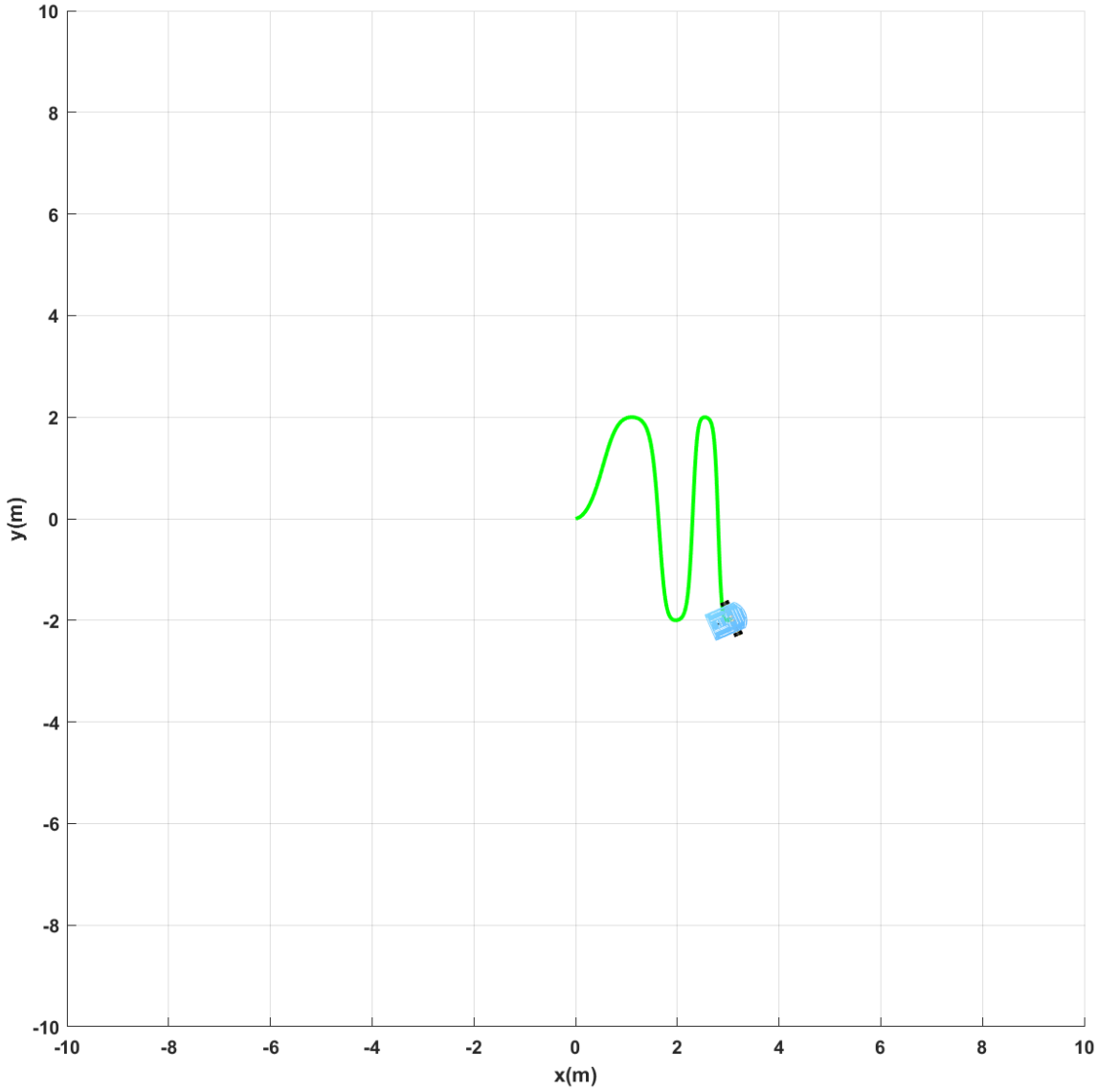
Posición inicial robot un bucle para ajustar la grafica para las subidas y bajadas

Primeramente vamos a tener definidas las posiciones iniciales del robot para el eje de x,y donde observamos que asi debe estar conforme a la grafica propuesta, ademas de comenzar con un angulo de pi/4 para que este en la orientación correcta del robot y desde ahi comenzar a trazar la funcion que declararemos mas adelante. En el bucle tenemos una iteracion sobre cada paso de tiempo desde 1 hasta N-1, como ya sabemos N son las muestras en el vector de tiempo, después calculamos la orientación del robot en el paso de tiempo actual que se basa en nuestra funcion senoidal, calculamos los cambios de las posiciones del robot en funcion de la orientacion en el paso de tiempo actual y se actualizan sumando los cambios calculados y se tiene un ajuste final para que siga la forma cuando cambie de subida a bajada.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% BUCLE DE SIMULACION %%%%%%%%%%%%%%%%%%%%%%%%%%%%

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);   % Orientación del robot en radianes (rad)

x1(1) = 0;    % Posición inicial eje x
y1(1) = 0;    % Posición inicial eje y
phi(1) = pi/4; % Orientación inicial del robot

for k = 1:N-1

    % Calculamos la orientación del robot basado en función senoidal en el punto actual
    phi(k+1) = atan(dF_x(k));
```

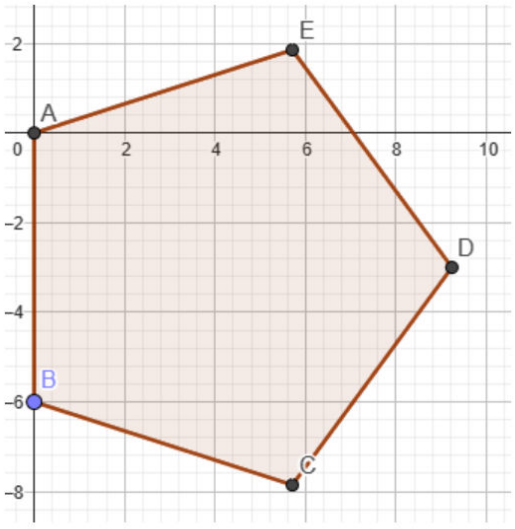
```
    % Calculamos el cambio en x,y
    delta_x = ts * cos(phi(k+1));
    delta_y = ts * sin(phi(k+1));

    % Calculamos las nuevas posiciones del robot
    x1(k+1) = x1(k) + delta_x;

    % Ajustamos la posición y para que siga la forma de la gráfica
    y1(k+1) = F_x(k+1);

end
```

Vemos que correctamente se tuvo el movimiento y el ajuste de la gráfica propuesta así como el rango de Y que era de 2 a -2 para cada una de las ondas, solo que en este caso no tuvimos el ajuste sobre la frecuencia de las ondas por temas de calculo del rango de x debido a que al aumentar puede generar solo pocas ondas pero por cuestiones de tiempo se genero de esta manera teniendo la forma adecuada de la grafica propuesta como una senoidal.



- **Segunda trayectoria**

X = [0 a 9]



Para la segunda trayectoria vamos a observar que tenemos un pentagono el cual tiene un rango en X que va de 0 a 9 para primeramente tomarlo en cuenta, también tenemos que tomar en cuenta el recorrido que se va a ejecutar y la cantidad de unidades que va a avanzar que en este caso es de 6.

Variables para el tiempo de simulación y muestreo, vector de tiempo y muestras

Continuando vamos a proceder a definir los parámetros que se cambiaron en el archivo Main para poder hacer toda la trayectoria, teniendo primero un tiempo de simulación de 9 segundos debido a que sabemos que son la cantidad de movimientos que estara efectuando el robot tanto giros como avances en las velocidades lineales y angulares que explicaremos más adelante. Con un tiempo de muestreo de 1 para que se pueda ver de mejor forma y vaya más fluido además de que hará cada movimiento en 1 segundo .

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

Posición inicial robot

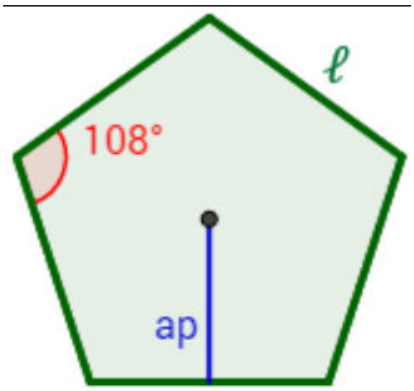
Para la posición inicial tomaremos en cuenta la imagen propuesta donde vemos que la figura se situa en las coordenadas (0,0) para el punto A y de ahi empieza el recorrido por los demás a B,C,D,E y regresa a cerrar la

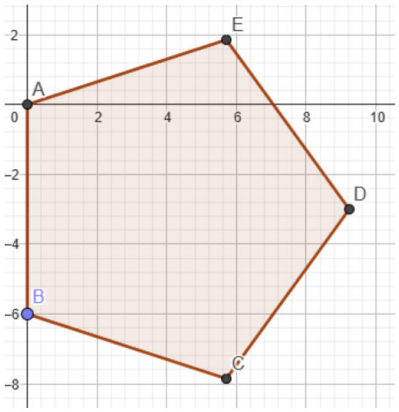
figura en A, entonces comenzando en A ponemos las posiciones iniciales del robot en el eje x,y de 0. Para la orientación del robot comence con algunas complicaciones debido a que no comprendia bien como funcionaba hasta que probando valores con radianes pude llegar a un valor ajustado para que comenzara en la posición en que el robot viera hacía el punto B para comenzar a marcar las trayectorias definidas en los vectores de velocidades lineales y angulares.

```
%%%%%%%%%%%%%%%%%%%%%% 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) = 92.69;   % Orientacion inicial del robot
```

Velocidad lineal y angular

Como parte final del código tendremos lo que cambiamos de las velocidades lineales y angulares entonces para primeramente el vector de las velocidades lineales que es cuanto va a avanzar el robot lo definimos con 6 unidades ya que como observamos en la primera imagen tenemos que el primer recorrido que hace es de 0 a 6 por lo que con ello vamos a saber que cada lado será de 6 y por eso lo definimos de esta forma. Ahora para continuar con las velocidades angulares que seran los momentos en los que gire el robot tendremos definido un valor de 1.25664 y esto se debe a que tendremos que considerar que el ángulo que se forma en un pentagóno es de 108 grados:

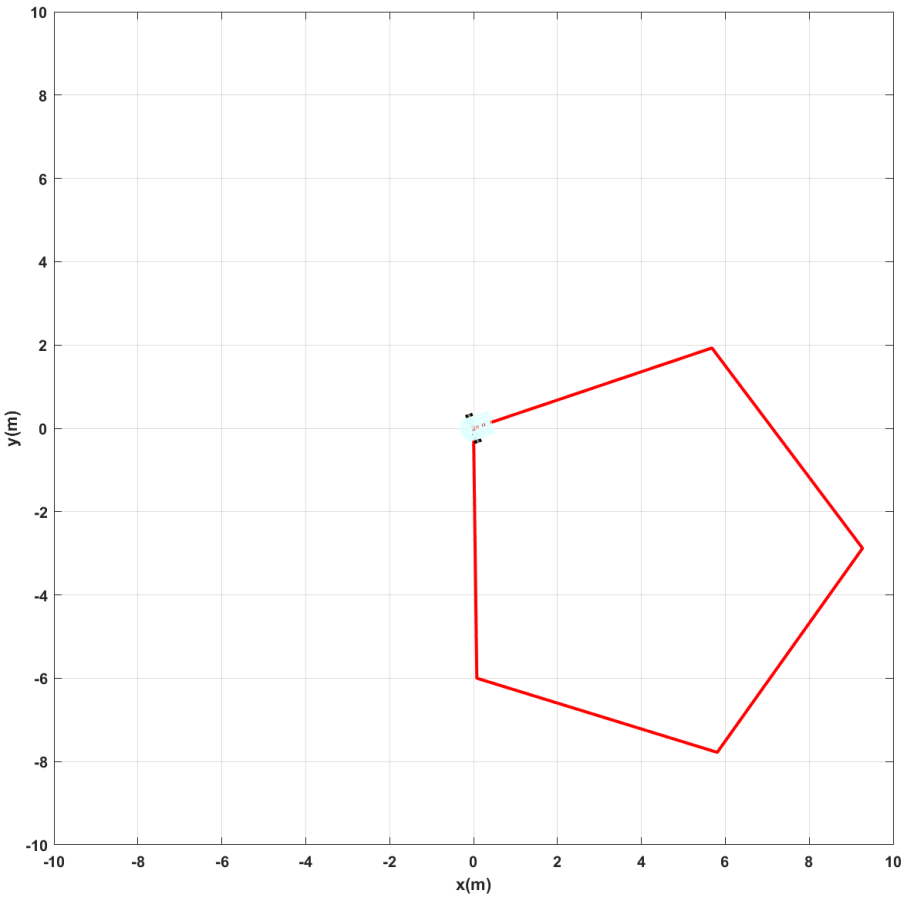




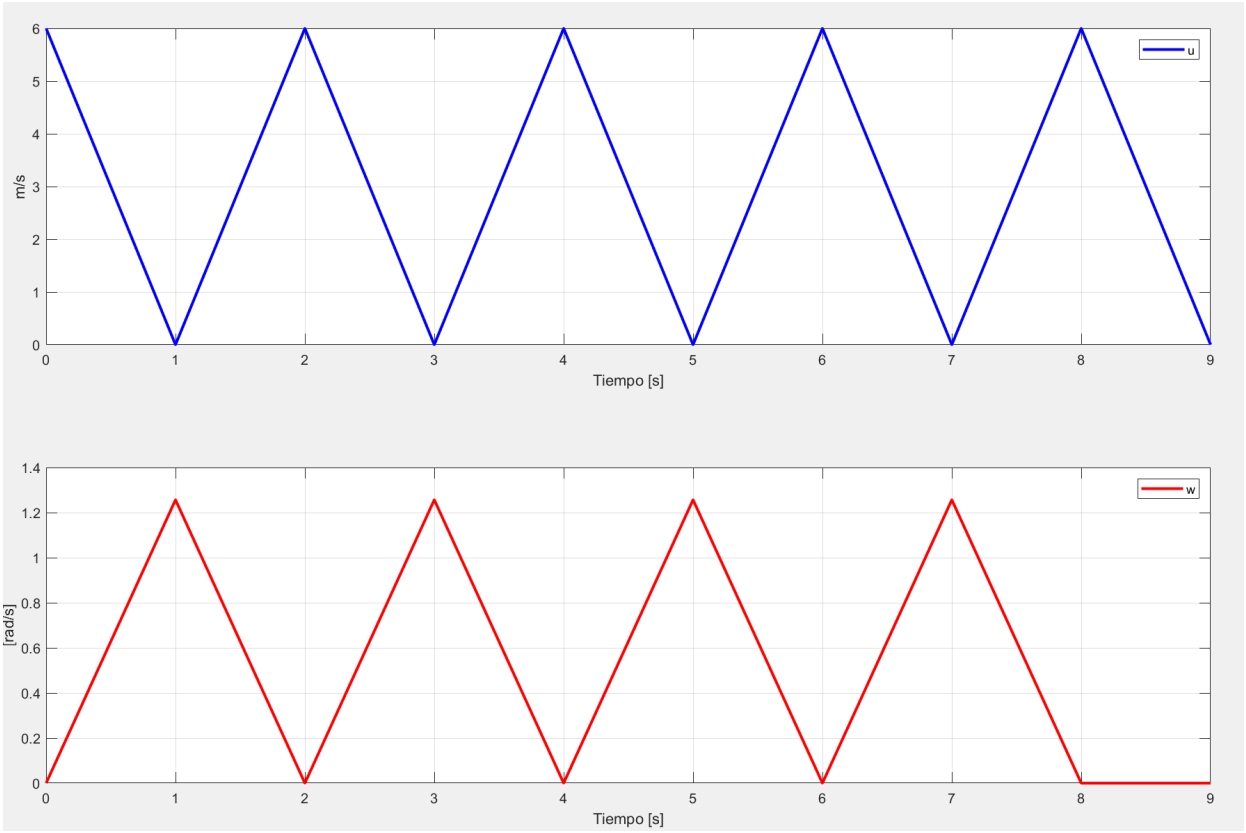
Con esto y conociendo nuestra figura propuesta observar que para el ángulo formado en A se va a tener de 108 grados por lo que ya dijimos y considerando que esta recto al eje y podria formar un ángulo de 180 grados, solo necesitamos hacer la resta de estos dos grados para saber que ángulo se forma: 180 -108 = 72 grados. Entonces tenemos los 72 grados para cada uno de los puntos marcados en la figura y convertimos los grados a radianes: 72 * pi/180 = 1.25664 dandonos el valor exacto para poder ponerlo en el vector de las velocidades angulares.

El recorrido basicamente es el mismo porque son lados iguales y el angulo ya es igual también para formar una figura que este bien posicionada y sea geometricamente correcta, avanzando 6 unidades para cada lado del pentagono y teniendo los giros de 1.25664 radianes para hacer las trayectorias correctas. Tendremos que declarar un 0 en las velocidades lineales y angulares al final debido que si no tenemos declarado un movimiento siguiente no puede efectuar el anterior es por ello que lo ponemos como 0, al final de cuentas esto no va a afectar para el trazado de la trayectoria.

```
%%%%%%%%%%%%%%%%%%%%%% VELOCIDADES DE REFERENCIA %%%%%%%%%%%%%%%%%%%%%%%%%
u = [6,       0,  6,        0,  6,        0,   6,         0,  6,  0];
w = [0, 1.25664,  0,  1.25664,  0,  1.25664,   0,   1.25664,  0,  0];
```



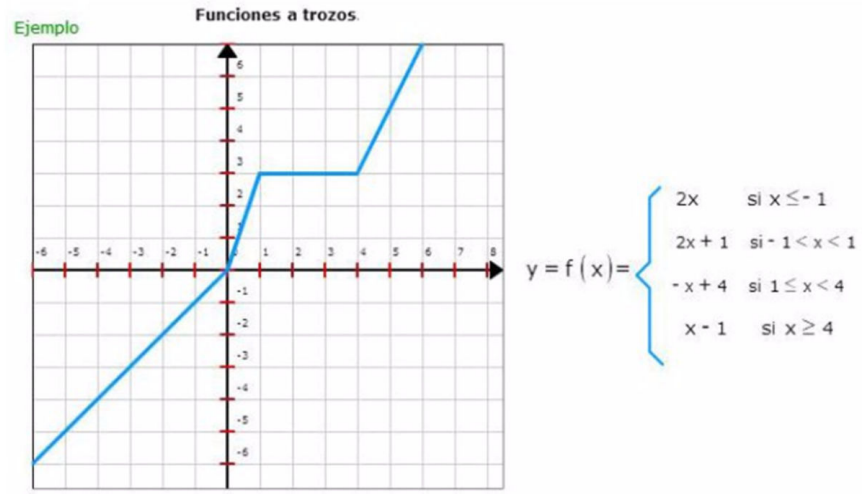
- **Gráficas de las velocidades lineales y angulares**



Las gráficas que se muestran podemos observar que para la velocidad lineal y la velocidad angular tenemos similitudes entre ellas con picos los cuales se pueden explicar primeramente que vemos que el primer pico lo encontramos en 0 con una bajada mientras que el pico de la velocidad angular se encuentra arriba debido a que se esta efectuando el giro y asi consecutivamente. Cuando el pico de la velocidad lineal se encuentra arriba y el de la velocidad angular se ecuentra abajo es que se esta haciendo el movimiento de avanzar sin tener giros y cuando es al reves es que se esta haciendo el giro de la velocidad angular y no hay velocidad lineal porque no esta avanzando el robot en ese movimiento.

- **Tercera trayectoria**

X = [-6 a 6]



Para esta ultima trayectoria vamos a poder observar que va a ser una grafica donde tendremos que definir varios giros y tambien dezplazamiento del robot como las velocidades lineales y angulares por lo que hay que considerar cada movimiento que se haga del robot para el muestreo

Variables para el tiempo de simulación y muestreo, vector de tiempo y muestras

Con lo anteriormente definido podemos decir que en mi caso voy a considerar 7 movimientos para que el robot efectue la trayectoria completa sobre todo porque primeramente estare efectuando los giros en la velocidad angular y posteriormente el avance del robot en la velocidad lineal. Tambien un tiempo de muestreo de 1 segundo que va a indicar que cada parte de la simulación osea los movimientos los tiene que efectuar en 1 segundo sobre todo para tener un buen movimiento y trazar bien la trayectoria de cada parte sin que tenga perturbaciones o desalineaciones.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% TIEMPO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tf = 7;              % 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
```

Posición inicial robot

Comenzando de la parte de la gráfica que podemos ver que empieza desde la parte inferior izquierda tendremos la posicion inicial del robot que declararemos como las coordenadas (-6,6) lo que puede tenerse como que desde ahi empezara su recorrido. Y con un angulo de pi/4 para poder tener ya la rotacion del robot como deberia de estar para poder comenzar a efectuar los movimientos de las velocidades lineales y angulares.

```
%%%%%%%%%%%%%%%%%%%%%%%% 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) = -6;    % Posicion inicial eje x
y1(1) = -6;   % Posicion inicial eje y
phi(1) = pi/4;   % Orientacion inicial del robot
```

Velocidad lineal y angular

Ahora continuaremos con la declaracion de las velocidades lo que vamos a explicar como los movimientos que hara el robot en cada parte:

1. Avanzar 8.5 unidades (velocidad lineal) y no tiene rotación (velocidad angular)

2. No avanza (velocidad lineal) y tiene rotación de 0.65 radianes (velocidad angular)

3. Avanzar 3 unidades (velocidad lineal) y no tiene rotación (velocidad angular)

4. No avanza (velocidad lineal) y tiene rotación negativa de -1.43 radianes (velocidad angular)

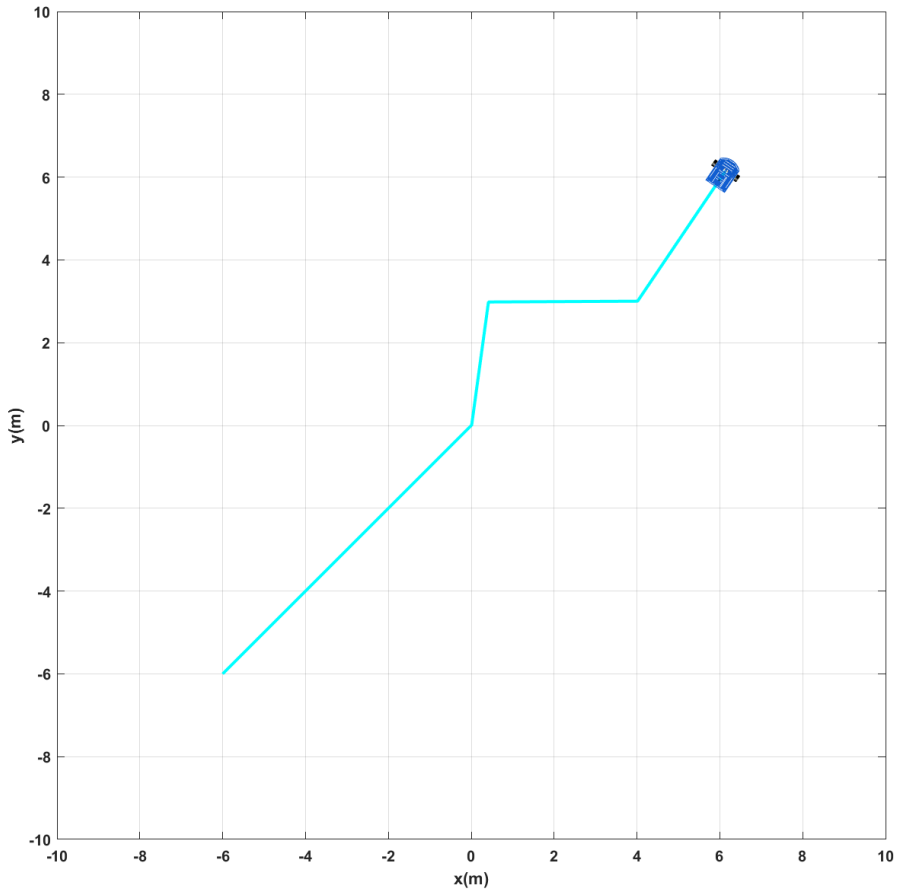
5. Avanzar 3.6 unidades (velocidad lineal) y no tiene rotación (velocidad angular)

6. No avanza (velocidad lineal) y tiene rotación de 0.972 radianes (velocidad angular)

7. Avanza finalemente 3.8 unidades (velocidad lineal) y no tiene rotación debido a que termina el recorrido (velocidad angular)

8. Finalemente tendremos que declarar un 0 en las velocidades lineales y angulares debido que si no tenemos declarado un movimiento siguiente no puede efectuar el anterior es por ello que lo ponemos como 0, al final de cuentas esto no va a afectar para el trazado de la trayectoria.

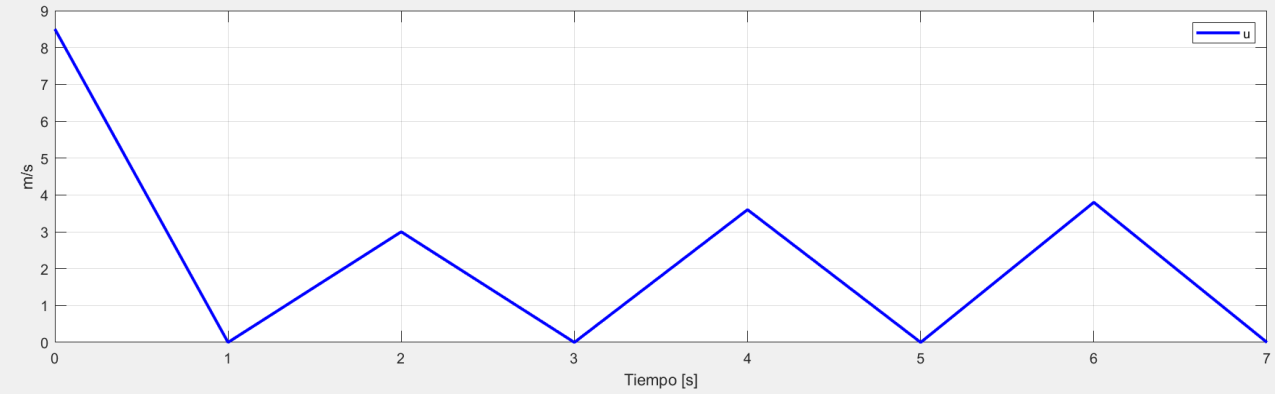
```
%%%%%%%%%%%%%%%%%%%%%% VELOCIDADES DE REFERENCIA %%%%%%%%%%%%%%%%%%%%%%%%%

u = [8.5,    0,  3,      0,  3.6,      0,  3.8,  0];
w = [0,   0.65,  0,  -1.43,    0,  0.972,    0,  0];
```

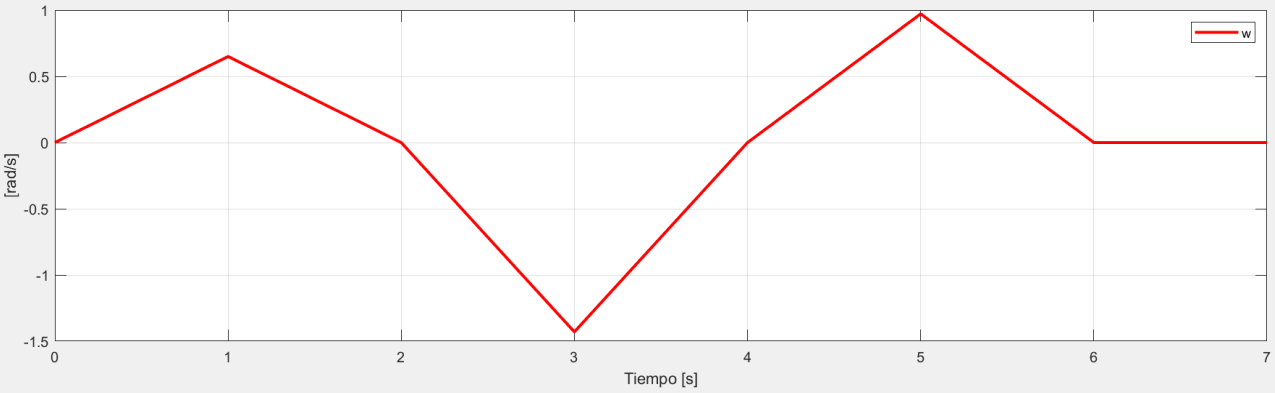


- **Gráficas de las velocidades lineales y angulares**

Ahora para las gráficas tendremos que para la lineal se pueden observar los picos que va a ser la desaceleración del robot por lo que tendremos una velocidad que va a alcanzar y bajando a 0 debido a que en esos instantes de tiempo se tiene un 0 para la lineal pero si representara que a continuación vuelva a tener una velocidad nuevamente y asi consecutivamente dependiendo de la cantidad de movimientos declarados.

Para la grafica de la velocidad angular tendremos lo mismo expresado pero ahora con el angulo dependiendo de como se ve el angulo y que tan grande es la rotacion que hace o que tan pequeño es el angulo para poder hacer la rotacion correspondiente, en la parte de los 0 es cuando tenemos el avance del robot para despues efectuar nuevamente un giro que se representa como esa caida y luego esa subida de la magnitud del angulo.


u
m/s
Tiempo [s]
0
1
2
3
4
5
6
7
8
9


w
[rad/s]
Tiempo [s]
-1.5
-1
-0.5
0
0.5
1