Memoria de la asignatura de Robótica 2021/2022

Luis Gutiérrez Jerez

Manuel Ángel Pazos Ruiz

PRIMERAS CLASES

En la primera clase, realizamos un acercamiento al lenguaje de programación Matlab y algunas de sus funciones útiles que serán detalladas a continuación:

- Plot(var1,var2): Crea una representación gráfica (en este caso en 2 dimensiones) de los datos en var2(eje Y) frente a los valores de var1 (eje X).
- Tic: Guarda un identificador temporal para su posterior toc
- Toc(var1): Devuelve el tiempo que ha traspasado tras el tiempo determinado mediante el identificador temporal (que deberá de estar en var1)

La explicación se basó en la importancia de las smooth function (funciones continuas derivables a un punto) en los ámbitos de la róbotica para evitar movimientos bruscos mecánicos.

Hemos realizado el proyecto de mostrar gráficamente la función sen(X) usando una función auxiliar, a continuación el código:

```
1   k=1;
2  for s=0:0.01:2*pi
3
4      t(k) = s;
5      y(k) = signal_v0(t(k));
6      k=k+1;
7
8  end
9
10 %plot(t,y)
11
12 tstart = tic;
13 segundos = 10;
14 t(k) = 0;
15 k = 1;
16 while (t(k) < segundos)
17     k=k+1;
```

```
18      t(k) = toc(tstart);
19     y(k) = signal_v0(t(k));
20
21 end
22
23 plot(t,y)
```

```
1 function salida = signal_v0(t)
2 %UNTITLED2 Summary of this function goes
3 here
4 %   Detailed explanation goes here
5
6 salida = sin(t);
  end
```

Para la siguiente clase también seguimos con el mismo propósito de la anterior.

A partir de la tercera clase se comenzó a trabajar con el objetivo de animar la gráfica sen(x) para simular como se comportaría la cabeza del robot, tras un intento fallido en dicha clase, se consiguió animar en la clase 4 junto a una modificación de la función auxiliar anterior(ahora pasaremos a llamarle signal_vf), adjunto el código fuente:

```
1   clear all
 2  clc
 3
 4  delay = 4;
 5  periodo = 6;
 6  amplitud = 1;
 7
 8  k = 1;
 9  for s=0:0.01:4*pi
10
11      t(k)=s;
```

```
12     y(k)=signal_vf(t(k),delay,periodo,amplitud);
13
14     k = k+1;
15 end
16
17 k = 1;
18 tstart = tic;
19 segundos = 12;
20 tiempo(k) = 0;
21
22 while tiempo(k) < segundos
23
24     y2(k) = signal_vf(tiempo(k),delay,periodo,amplitud);
25     cla
26     plot(t,y)
27     hold on
28
29         plot(tiempo(k),y2(k),'Or')
30     k = k + 1;
31     tiempo(k) = toc(tstart);
32
33     axis([0 12 -1 1])
34
35     drawnow
36
37 end
38
39 plot(t,y)
```

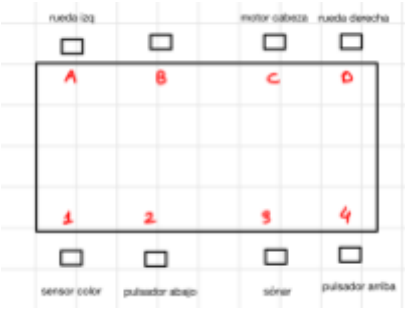
```
1  function salida = signal_vf(t, delay,periodo,amplitud)
2  %UNTITLED Summary of this function goes here
3  %   Detailed explanation goes here
4
5  salida = amplitud * sin((2*pi/periodo)* (t-delay));
6
```

```
 7  if t < delay
 8          salida = 0;
 9  else if t > delay+periodo
10          salida = 0
11  end
12
13  end
```

En la clase 6 se nos dio un acercamiento a las funciones de Matlab que interactúan directamente con el robot, con el objetivo de realizar movimientos en las distintas partes de este Algunas de estas funciones son:

- Legoev3('USB'): Permite conectar el robot y sincronizarlo con el script de Matlab.
- touchSensor(NombreConexion,NPuerto): Devuelve un objeto del sensor de toque conectado en el puerto NPuerto del robot NombreConexion (Tenemos que tener en cuenta que puede haber distintos robots en un mismo sistema, por esto es necesario tener una identificación de cada uno de ellos).
- sonicSensor(NombreRobot): Devuelve un objeto del sensor de proximidad situado en la cabeza del robot, notar que, no es necesario identificar el nº del puerto debido a que al solo haber un sensor de este tipo conectado en el robot, este lo identifica automáticamente.
- Motor(NombreConexion,NPuerto): Nos devuelve un objeto con el que podremos accionar los motores correspondiente al NPuerto indicado configurando posteriormente la velocidad.
- objetoMotor.Speed: Nos permite configurar el valor de la velocidad de giro del motor.
- objetoMotor.start(): Nos permite accionar el motor.
- readDistance(objetoSonicSensor): Devuelve la distancia medida por el objeto SonicSensor
- readTouch(objetoTouchSensor): Devuelve un booleano que será true si el pulsador ha sido accionado.

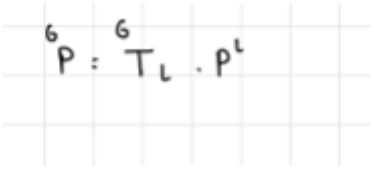
También se nos pidió realizar un esquema de la ubicación de los diferentes puertos del robot el cual muestro a continuación:



A continuación se muestra el código fuente:

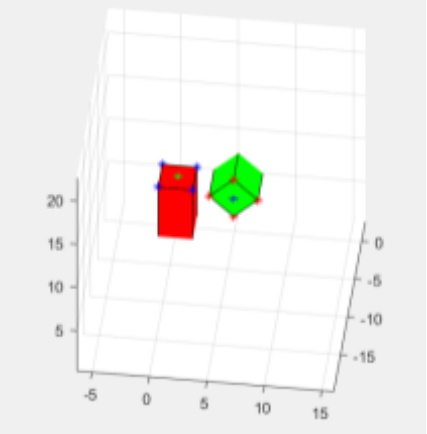
```
WallE=legoev3('USB');

PulsadorArranque =
touchSensor(WallE,4);
Sonar = sonicSensor(WallE);
motorDerecho = motor(WallE,'D');
motorIzquierdo = motor(WallE,'A');

start(motorDerecho);
motorDerecho.Speed = 50;


distancia = readDistance(Sonar);
Pulsacion =
readTouch(PulsadorArranque);
```

En clase vimos como teniendo en cuenta un objecto en un eje de coordenadas este poseía un punto de referencia local y uno global, esto resultaba en la explicación de que si queriamos mover un objeto tenemos que tener en cuenta esto:



P(Global) es una matriz de 4x1 donde la última fila es 1, la matriz de transformación de sistema local a general es 4x4 y P(local) es una matriz de 4x1, la cual también tiene la última fila con un 1.

A partir de esto se nos pidio que hicieramos una transformación en un objeto hecho en clase y que además marcáramos con "*" los vértices y el centro



Realizamos un sistema de representación tridimensional para representar la cabeza del robot, antes de mostrar el código es necesario algunas explicaciones conceptuales y matemáticas:

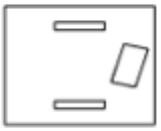
configuración de objetos rígidos: magnitudes que si las conozco me permiten establecer o
%conocer las coordenadas de cada uno de los puntos que componen el cuerpo
%respecto un eje de referencia

DOF : degree of freedom : tanta libertad como variables de configuración
%puedo cambiar

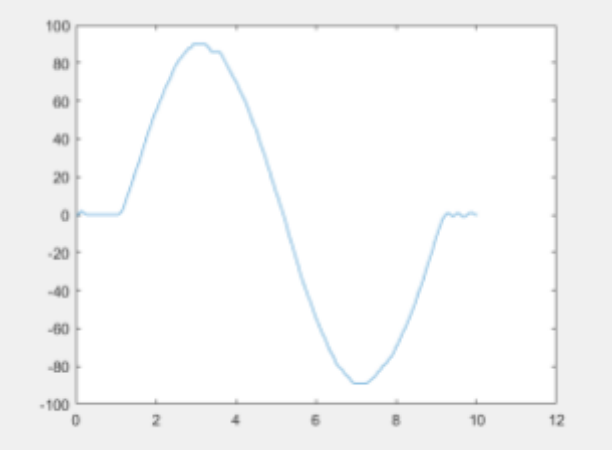
La lógica matemática que vamos a usar van a ser las matrices y las matrices de transformación (o cambio de base)

SESIÓN 27 OCTUBRE

El profesor propuso un ejercicio en la moodle el cual realizamos en clase, este consistía en dibujar el robot y que este creará unos “hijos” mediante parents para que cuando se moviera este todo fuera acoplado, así creamos el robot que muestro a continuación:



Posteriormente en la sesión de práctica hicimos un bucle que se quedara esperando a que pulsáramos el pulsador del robot , cuando este se pulsara y se levantara la pulsación terminaba, si pasan 10 segundos igualmente termina, después hicimos algo parecido, esta vez con el Sonar, cuando se pulsara el botón el programa debía mostrar una gráfica de distancia con respecto al Sónar, después por último se propuso realizarlo con la cabeza y que mostrara los ángulos.

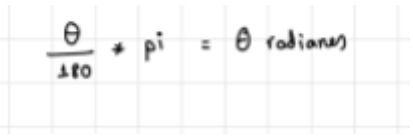



```
while(readTouch(touchSensor(Wall1,4))==0)
    disp 'esperando pulsacion'
end

while(readTouch(touchSensor(Wall1,4))==1)
    disp 'suelta el pulsador'
end
motorCabeza.Speed=100;
start(motorCabeza);
i = 1;
clc;
tstart = tic;
t(1) = toc(tstart);
tiempo_final = 10;
resetRotation(motorCabeza);
while ((t(i) < tiempo_final) && (readTouch(touchSensor(Wall1,4))==0))

    i = i +1; %%incremento del indice
    %%lectura del tiempo
    t(i) = toc(tstart);

    rotacion(i)=readRotation(motorCabeza)


    plot(t,rotacion);
    drawnow

end
motorCabeza.Speed = 0;
motorCabeza.stop;
```

SESIÓN 3 DE NOVIEMBRE

En la siguiente sesión hicimos un programa basado en las dos sesiones anteriores, esta consistía en hacer mover la cabeza al robot y conforme girara ir viendo que el giro en la simulación es el mismo, tuvimos algunos problemas ya que no tuvimos en cuenta el casting a double y al pasar los ángulos a radianes pegaban saltos, lo que se veía en la representación como intermitencias en el giro de la cabeza.

Con esta fórmula convertimos de ángulos a radianes acompañada de un casting de double




```
while(readTouch(touchSensor(WallE,4))==1)
    disp 'suelta el pulsador'
end
motorCabeza.Speed=5;
start(motorCabeza);
i = 1;
clc;
tstart = tic;
t(i) = toc(tstart);
tiempo_final = 10;
resetRotation(motorCabeza);
while ((t(i) < tiempo_final) && (readTouch(touchSensor(WallE,4))==0))

    i = i +1; %%incremento del indice
    %%lectura del tiempo
    t(i) = toc(tstart);

    rotacion(i)=readRotation(motorCabeza)
   % rotacion(i) = double(rotacion(i))*pi/180
   %rotacion(i) = deg2rad(double(rotacion(i))*pi) /180
    %rotacion(i)= (rotacion(i)/180)*pi
     pinta_robot(3,3,0,deg2rad(double(rotacion(i))*pi) /180,SR_robot,SR_cabeza);


   % plot(t,rotacion(i));
    rotacion(i)
    drawnow

end
motorCabeza.Speed = 0;
motorCabeza.stop;

disp 'terminado'
```

SESION 11 DE NOVIEMBRE

En la sesión de hoy hemos hecho un avance teniendo en cuenta todo lo anterior, la diferencia es que en esta clase hemos hecho la función controlador la cual hace que dado un ángulo, la cabeza gire para alcanzar ese ángulo en la mayor velocidad posible y menos oscilación, también limitamos la potencia en 100 ya que si le damos más valor que este, seguirá siendo el máximo (100), a continuación muestro el código de dicha sesión:

```
while ((t(i) < tiempo_final) && (readTouch(touchSensor(WallE,4))==0))

    i = i +1; %%incremento del indice
    %%lectura del tiempo
    t(i) = toc(tstart);

    Giro(i)=readRotation(motorCabeza)
   % rotacion(i) = double(rotacion(i))*pi/180
   %rotacion(i) = deg2rad(double(rotacion(i))*pi) /180
    %rotacion(i)= (rotacion(i)/180)*pi
    referencia(i) = 90;
    error(i) = referencia(i)-Giro(i);

    controlador(i) = k * error(i);

    potencia(i) = controlador(i);

    if potencia(i) > 100
        potencia(i) = 100;
    end

    if potencia(i) < -100
        potencia(i) = -100;
    end

    motorCabeza.Speed=int8(potencia(i));



     pinta_robot(3,3,0,(double(Giro(i))*pi) /180,SR_robot,SR_cabeza);
   %% pinta_robot(3,3,0,,SR_robot,SR_cabeza)


   % plot(t,rotacion(i));
    %Giro(i)
    drawnow

end
```

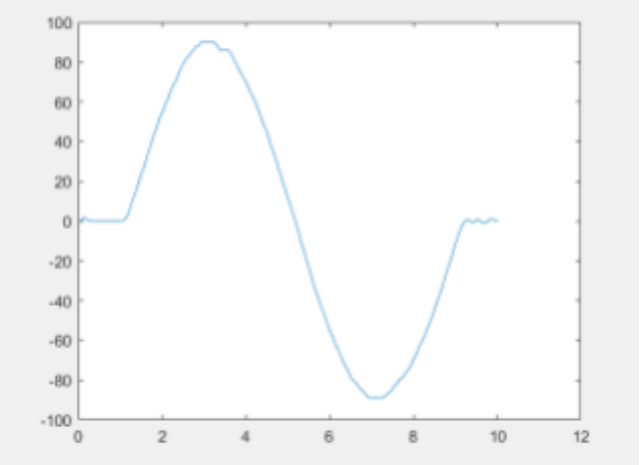
SESIÓN 17 NOVIEMBRE

Como avance a la anterior sesión le metimos la función seno antes realizada en lugar de el encoder de la rueda, de esta manera girara la cabeza sin oscilaciones ya que la función seno es infinitamente derivable y sin saltos bruscos, cuanto más derivable sea una función menos “picos” en la representación gráfica hay, lo que nos viene muy bien ya que así el robot mirara a ambos lados suavemente y al llegar a los extremos irá reduciendo la velocidad .

Como parámetros para esta función le introduciremos como “1” como Delay, “8” como periodo y “90” de amplitud.

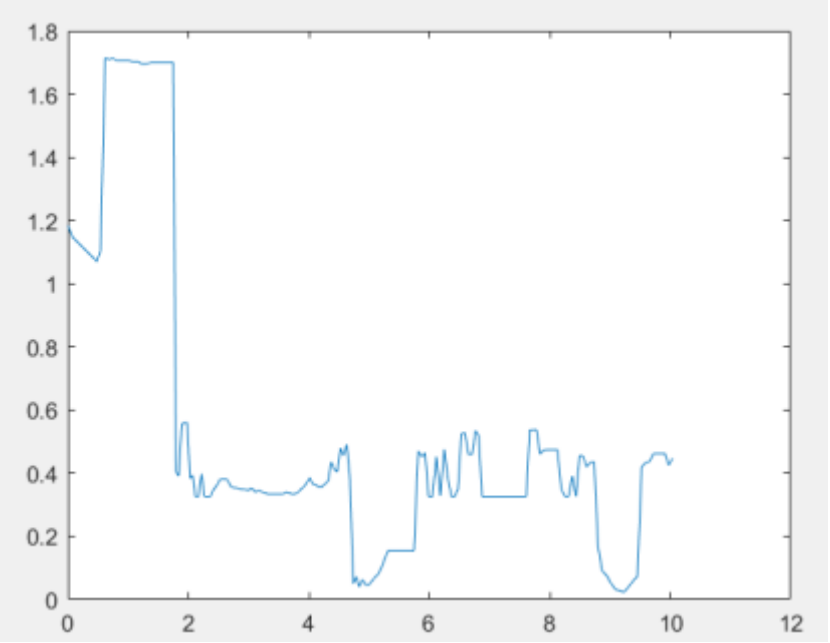
También tuvimos complicaciones con los decimales tras comentárselo al profesor lo solucionamos con un casting de double y la función “Degre2”.

Como podemos ver en la función de abajo esta describe el giro que hace la cabeza como en las clases anteriores, es una función seno para hacer el barrido de la cabeza y identificar obstáculos:



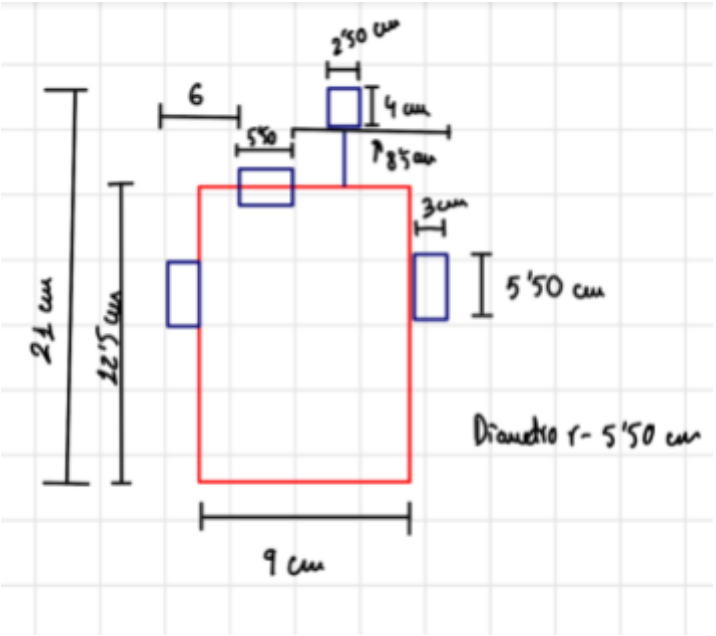
A continuación mostramos dos gráficas con las que investigamos un poco tras conectar la cabeza del robot a una rueda para que esta copiara el mismo

movimiento que hacíamos con la rueda, como vemos en la gráfica de abajo hice movimientos muy dispersos.



SESION 24 DE NOVIEMBRE

En esta clase debíamos tener en cuenta la medida de nuestro robot con la mayor precisión posible para que a la hora de representarlo en la representación sea lo más certera posible, así no nos darán errores del tipo que lo que este pasando en la simulación no se corresponda con la realidad



SESIÓN 1 DE DICIEMBRE

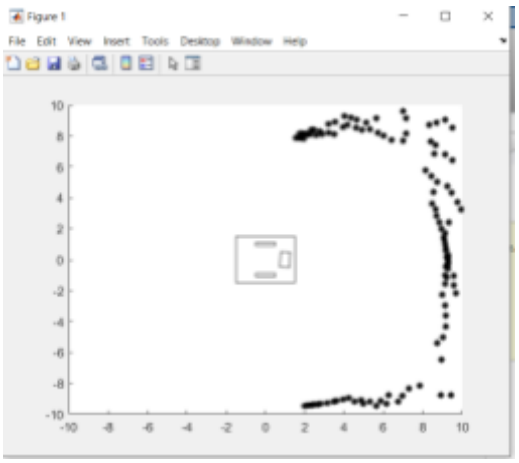
En esta sesión creamos una máquina de estados, dependiendo de que le vaya pasando al robot reacciona de una manera u otra, a continuación explicaremos la máquina de estados:

```
switch estado

    case 1 %andando hacia delante
        %if (readDistance(Sonar)<stop_distance) %si la distancia es menor que 35 para
         if (distancia(i)<stop_distance) %si la distancia es menor que 35 para
            estado=2; %transición de estado de paro
            transicion=i; %indice que marca el inicio del estado 2
        end


    case 2 %parando
        if (vel==0)
            if distancia(i)>stop_distance
                estado=1; %la transición a estado marcha hacia delante
                transicion=i; %indice que marca el inicio del estado 1
            else
                estado=3; %transición a estado girando cabeza
                transicion=i; %indice que marca el inicio del estado 3

            end
        end

    case 3 %girando cabeza

        estado=4; %la transición a estado girando robot
        transicion=i; %indice que marca el inicio del estado 4

    case 4 %girando robot
        estado=5; %la transición a estado marcha atrás
        transicion=i; %indice que marca el inicio del estado 5

   case 5 %marcha atrás
        if (t(i)-t(transicion)>t_marcha_atras)
            estado=2; %transición a estado girando cabeza
            transicion=i; %indice que marca el inicio del estado 2
        end
```

Este switch contiene la máquina de estados que funciona de la siguiente forma, el robot va hacia delante hasta que encuentra un obstáculo, en el momento que la la distancia de detención sea la indicada, se para, posteriormente hace un barrido girando la cabeza y por último da marcha atrás.

Teniendo en cuenta el código anterior, el robot Llega hasta que encuentre un obstáculo a la distancia de detención, posteriormente, el robot da marcha atrás y hace un barrido de puntos

Posteriormente el fichero generaba un mapa con todos los puntos que el sonar iba recogiendo de esta forma se ve recogido en la siguiente imagen:



SESIÓN 15 DICIEMBRE

En esta sesión se nos dieron diferentes objetivos a cumplir, debíamos hacer que el robot hiciera lo siguiente:

1-El robot debía andar hacia delante, hasta encontrar la distancia de detección “STOP_DISTANCE” fuera igual a 25, posteriormente irá hacia atrás

2-El robot irá hacia delante, se detendrá, girará la cabeza con la función seno ya implementada.

3-Haría todo lo anterior además de hacer un mapa de puntos.

4-Después girará sobre sí mismo 90 grados y se moverá hacia delante.

esto último se haría para resolver el problema del laberinto, y después se mejoraría en vez de girando un ángulo fijo de 90 grados, girando el ángulo que estime oportuno para una mayor precisión.

En esta sesión conseguimos el primer paso ya que nuestro robot iba hacia delante y hacia detrás de tal forma que lo siguiente a conseguir es moviera la cabeza para ver los obstáculos a su alrededor.

SESIÓN 22 DICIEMBRE

En esta sesión nos vimos interrumpidos por un problema que nos impedía mover la cabeza de nuestro robot, analizando el código vimos que el problema se localizaba en el case 3, de esta forma intentamos solucionarlo.

Como se muestra en el video el robot sólo nos hacía esto:

https://www.youtube.com/watch?v=wCLrzUah4DA

Describe un movimiento hacia delante y hacia atrás sin mover la cabeza y como si estuviera saltando estados.

SESIÓN 12 ENERO

En esta sesión solucionamos los errores de la anterior sesión alojados en el case 3, el problema estaba en la cabeza del robot que no le añadimos el error y cuando la cabeza giraba no detectaba el giro.

Una vez solucionado esto, nuestro robot hacia todos los cases y de esta forma iba hacia delante se paraba cuando encontraba un obstáculo y posteriormente giraba 90 grados, posteriormente lo probamos en el circuito montado en el laboratorio, en vez de hacerlo girar 90 grados, le metimos la funcionalidad de girar un ángulo determinado adaptándose a sus necesidades

```
case 4 %girando sobre si mismo
    disp 'estoy en case 4'
    error_yaw(i) = (referencia_yaw - yaw(i));
    k_giro = 1.1;
    vel = int8(k_giro*error_yaw(i));
    Power1 = vel;
    Power2 = -vel;
```

En estas imágenes podemos apreciar al robot recorriendo el circuito y buscando el hueco para conseguir llegar al final del recorrido.

https://www.youtube.com/watch?v=pYY9bSnvFQo

En este enlace podemos ver el recorrido del robot, la única complicación que tuvimos fue que había que tenerlo enchufado mediante el cable ya que el pincho wifi estaba ocupado y muchas veces era inevitable que el cable quedara debajo de las ruedas.

Con esto completamos la práctica número dos.

SESIÓN 19 ENERO

En esta sesión empezamos la practica numero tres, haciamos un escenario de tal forma que partimos del punto (0,0) y llegara al punto (0,40), gracias a los videos y scripts que nos fueron proporcionados conseguimos que hiciera este recorrido:

https://www.youtube.com/watch?v=AKb_6MlMIfk

Posteriormente se nos pidió que pasara por un punto intermedio de y se parara en el punto (0,40).

https://www.youtube.com/watch?v=JiwRQjCfGLU

En esta segunda parte mediante el uso de spline y pure pursuit conseguimos completar la practica numero 3

Para conseguir esto hemos hemos modificado el script incluyendo algunas modificaciones.

Primero, hemos añadido la parte del código que nos permite describir el punto a recorrer perteneciente al camino (el cual hemos calculado usando la función mínima distancia) y hemos iniciado la variable punto según si es un punto perteneciente a la trayectoria o si es el punto final.

```
delta= (x(i)-punto(1))*sin(theta(i))-(y(i)-punto(2))*cos(theta(i));

LH=sqrt((x(i)-punto(1))^2+(y(i)-punto(2))^2);

rho=2*delta/LH^2;



%Control proporcional de la velocidad
Kp=1.1;
%final=[camino(end,1) camino(end,2)]; %para converger al final del camino
final=punto; %para converger a un punto
Distance_to_end=sqrt((x(i)-final(1))^2+(y(i)-final(2))^2);
velocidad=Kp*Distance_to_end;
```

En este fragmento de código definimos la delta del recorrido junto al LH y el rho como lo hemos definido en prácticas anteriores.

```
x0 = [0 0 pi/2];
xf = [40 40 pi/2];

pdx=x0(1)+dd*cos(x0(3));
pdy=x0(2)+dd*sin(x0(3));
pax=xf(1)-da*cos(xf(3));
pay=xf(2)-da*sin(xf(3));
 xc=[x0(1) pdx pax xf(1)];
 yc=[x0(2) pdy pay xf(2)];


      xc = [0 pdx 20 pax 40]
      yc = [0 pdy 20 pay 40]


      camino = funcion_spline_cubica_varios_puntos(xc,yc,2)';
```

Para generar el camino se ha usado lo aprendido en la función “ordenminimo” donde hemos descrito los puntos iniciales y finales, obtenido los vectores y realizado las sumas de los subvectores (la descomposición en cada eje de estos) para calcular el camino resultante.