Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Puebla

Implementación de robótica inteligente (Gpo 501)

Actividad 1.8 (Seguimiento de Trayectorias)

Alumno

José Diego Tomé Guardado A01733345
Victor Manuel Vázquez Morales A01736352
Pamela Hernández Montero A01736368
Fernando Estrada Silva A01736094

Fecha de entrega

Sábado 27 de Abril de 2024

Implementación Código de matlab

Primeramente vamos a tener la limpieza de la pantalla y también del command window para poder mostrar de mejor forma el funcionamiento, por eso se hace la limpieza del entorno antes de ejecutar el código por eso se hace esto. Después definimos los parámetros para el tiempo, se establece el de simulación en segundos para poder seguir cada una de las trayectorias, el tiempo de muestreo que es pequeño para que pueda seguir las trayectorias de la mejor forma sobre todo cada cuando se va a ir tomando la muestra.

```
%Limpieza de pantalla
clear all
close all
clc
%1 TIEMPO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tf=40;              % Tiempo de simulación en segundos (s)
ts=0.05;             % Tiempo de muestreo en segundos (s)
t=0:ts:tf;          % Vector de tiempo
```

También vamos a definir el tiempo para el hexagono debido a que se necesita normalizar el intervalo de tiempo ya que al normalizar podemos de esta forma tener un trazado del hexagono ya que si no lo hacemos se hace un trazado de un círculo. La normalización del tiempo es importante para poder formar el hexagono ya que controla cómo se van a distribuir los puntos en el plano xy a lo largo de un intervalo de tiempo. Con este incremento de 1 podemos ver como los puntos generados van a estar más cerca entre sí a lo largo de toda la trayectoria significando que la figura pueda tener más lados y se considere más compleja como en este caso un hexágono

```
%%%% PARA PODER DIBUJAR EL HEXAGONO %%%%%%%%%%%
%Se define el parámetro "t" para la proyección de trayectoria
tiempo=[1:1:13];
%Se normaliza el intervalo de tiempo al intervalo de variación del ángulo
t2= normalize(tiempo,"range",[pi,5*pi]);

%CAMBIAR POR t2 CUANDO SEA EL HEXAGONO
N= length(t);       % Muestras
```

Para las condiciones iniciales tenemos los valores de las posiciones iniciales del eje para el robot y también la orientación, es necesario igualar el punto de control con las proyecciones para seguir la trayectoria deseada y porque al inicio del movimiento el punto de control y el robot coinciden para calcular los errores de seguimiento.

```
%2 CONDICIONES INICIALES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Damos valores a nuestro punto inicial de posición y orientación
x1(1)=0;  %Posición inicial eje x
y1(1)=0;  %Posición inicial eje y
phi(1)=0; %Orientación inicial del robot
%Igualamos el punto de control con las proyecciones X1 y Y1 por su
%coincidencia
```

```
hx(1)= x1(1);        % Posición del punto de control en el eje (X) metros
(m)
hy(1)= y1(1);        % Posición del punto de control en el eje (Y) metros
(m)
```

Tenemos el seguimiento de las trayectorias con las ecuaciones paramétricas y se puede ver que también tenemos las velocidades de la trayectoria deseada donde tenemos la derivada de las ecuaciones de hxd y hyd para poder tener el seguimiento de las coordenadas tanto de x como de y porque se calcula la derivada de la función de la trayectoria deseada respecto al tiempo.

1. **Seguimiento de las trayectorias**

Para el seguimiento de las trayectorias definimos los siguientes parámetros para ajustar de forma correcta cada una de las trayectorias.

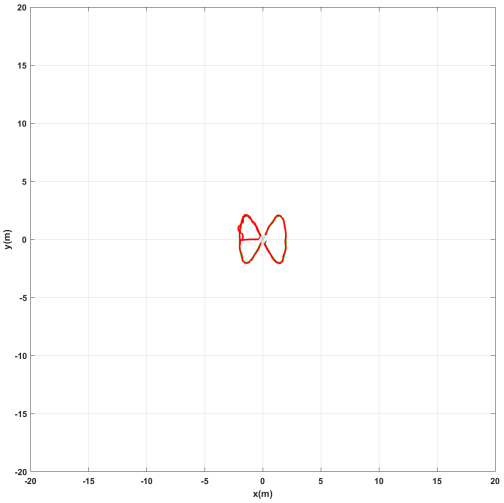
```
tf=47;              % Tiempo de simulación en segundos (s)
ts=0.04;             % Tiempo de muestreo en segundos (s)
t=0:ts:tf;          % Vector de tiempo

   %c)Matriz de Ganancias
   K=[18 0;...
      0 18];
```

Para cada una de las siguientes ecuaciones

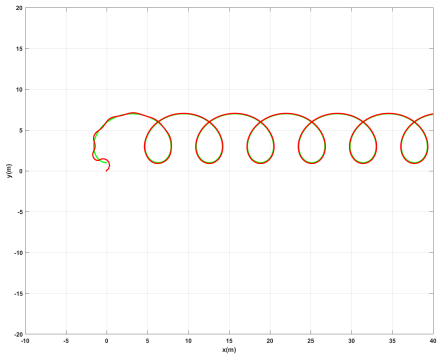
a) x =2cos(0.2*t), y = 2sen(0.4*t),

```
%Inciso a
hxd=-2*cos(0.2*t);
hyd=2*sin(0.4*t);
%Velocidades de la trayectoria deseada
hxdp=0.2*2*cos(0.2*t);
hydp=0.4*2*cos(0.4*t);
```



Análisis: Para este caso, observemos que la matriz de ganancias implementada permite al robot seguir la trayectoría de manera casi precisa. Notemos, de igual forma, que el robot parece oscilar considerablemente al seguir la trayectoria, por lo que para una futura mejora podríamos considerar disminuir ligeramente la ganancia de la velocidad angular o, en dado caso, saturar su valor para evitar este tipo de oscilaciones.

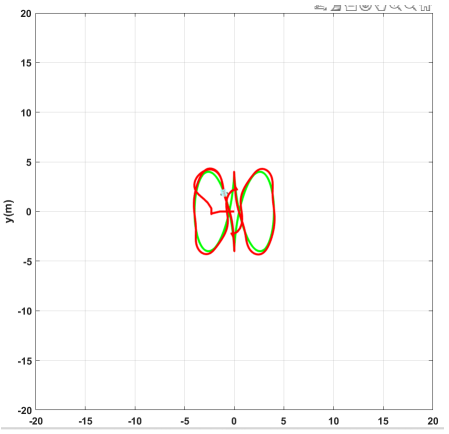
b) x =t−3sen(t), y = 4−3cos(t)



```
%Inciso b
hxd = t-3*sin(t)
hyd = 4-3*cos(t)
%Velocidades de la trayectoria deseada
hxdp = 1 - 3*cos(t)
hydp = 3*sin(t)
```

Análisis: Para esta trayectoria observemos que la matriz de ganancias implementada permite seguir al robot la trayectoria de manera precisa, presentando oscilaciones únicamente al inicio de la trayectoria.

c) x =3cos(t)−cos(3t), y = 4sin(3t),



```
%Inciso c
hxd = -3*cos(t)-cos(3*t);
hyd = 4*sin(3*t);
%Velocidades de la trayectoria deseada
hxdp =  3*sin(t)+3*sin(3*t);
hydp = 3*4*cos(3*t);
```

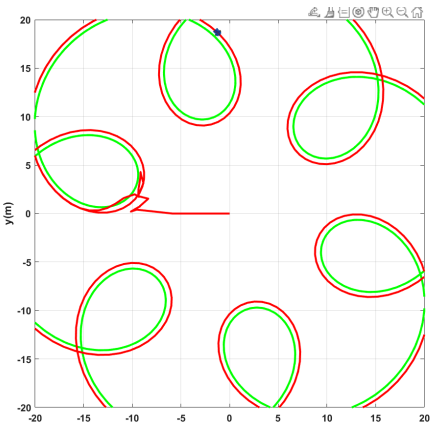
Análisis: Para este caso, observemos que la matriz de ganancias implementada permite al robot seguir la trayectoria de manera precisa siempre y cuando el movimiento o el cambio entre puntos no sea tan abrupto. Cuando los cambios son abruptos el robot presenta dificultades para seguir la trayectoria, por lo que podríamos aumentar o modificar la ganancia para la velocidad angular.

d) x = cos(t) + 1/2cos(7t) + 1/3sen(17t), y = sen(t) + 1/2sen(7t) + 1/3cos(17t)

```
%Inciso d
hxd =  cos(t) + 1/2*cos(7*t) + 1/3*sin(17*t);
hyd = sin(t) + 1/2*sin(7*t) + 1/3*cos(17*t);
%Velocidades de la trayectoria deseada
hxdp = -sin(t) - 7*1/2*sin(7*t) + 17*1/3*cos(17*t);
hydp = cos(t) - 7*1/2*cos(7*t) - 17*1/3*sin(17*t);
```

e) x =17cos(t)+7cos(17+7t), y = 17sen(t) −7sen(17+7t),

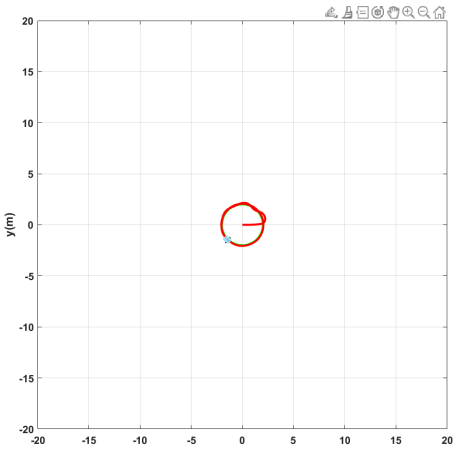
```
%Inciso e
hxd=-17*cos(t) + 7*cos(17+7*t);
hyd=17*sin(t) - 7*sin(17+7*t);
%Velocidades de la trayectoria deseada
hxdp=17*sin(t) - 49*sin(17+7*t);
hydp=17*cos(t) - 49*cos(17+7*t);
```



Análisis: Para este caso, notemos que el controlador funciona de muy buena manera, pues el robot sigue la trayectoria de manera casi precisa, presentando únicamente error al inicio de la trayectoria. Por otra parte, es importante mencionar que el robot parece presentar un pequeño desfase con respecto a la trayectoria, por lo que podría considerarse aumentar un poco la matriz de ganancias o indicarle de otra forma al robot que debe realizar esa corrección.

f) x =2cos(t), y = 2sen(t)

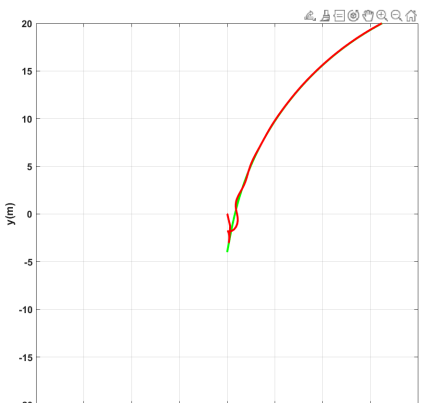
```
%Inciso f
hxd=2*cos(t)
hyd=2*sin(t)
%Velocidades de la trayectoria deseada
hxdp= -2*sin(t)
hydp= 2*cos(t)
```



Análisis: En esta trayectoria circular, podemos observar que el controlador presenta pequeños errores al inicio de la trayectoria, pero posteriormente logra llegar a un estado estacionario de poco error, lo que se puede ver representado en cómo el robot sigue muy bien la trayectoria circular.

g) x =5t−4sen(t), y = 5t−4cos(t)

```
%Inciso g
hxd=5*t-4*sin(t)
```

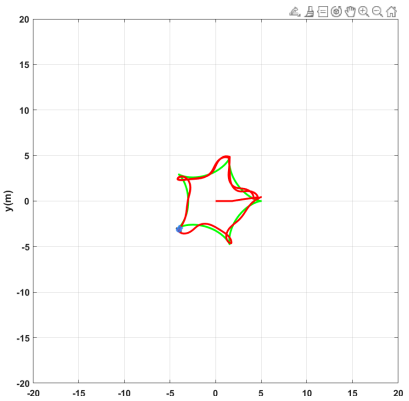


```
hyd=5*t-4*cos(t)
%Velocidades de la trayectoria deseada
hxdp=5-4*cos(t)
hydp=5+4*sin(t)
```

Análisis: En este caso, podemos ver que el robot en un inicio oscila intentando corregir su trayectoria. No obstante, en este caso observamos que el robot sigue la ruta de manera precisa.

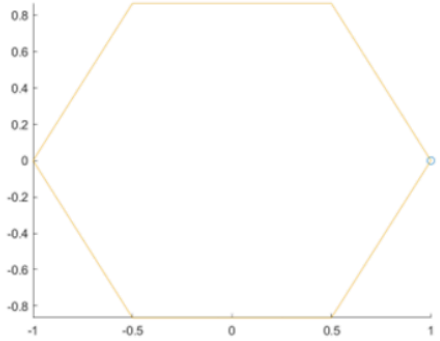
h) x =4cos(t)+cos(4t), y = 4sen(t) −sen(4t)

```
%Inciso h
hxd=4*cos(t)+cos(4*t)
hyd=4*sin(t)-sin(4*t)
%Velocidades de la trayectoria deseada
hxdp=-4*sin(t)-4*sin(4*t)
hydp= 4*cos(t)+4*cos(4*t)
```

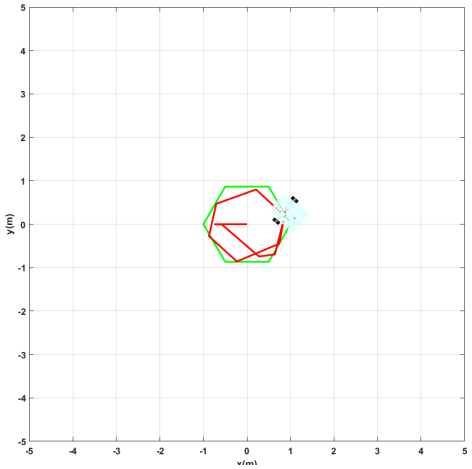


Análisis: En esta trayectoria en forma de estrella podemos identificar que al controlador le cuesta trabajo seguir la trayectoria o setpoint. Esto puede ser debido a que la figura presenta cambios muy abruptos ocasionados por el tamaño pequeño de la figura.

2. Trayectorias en específico

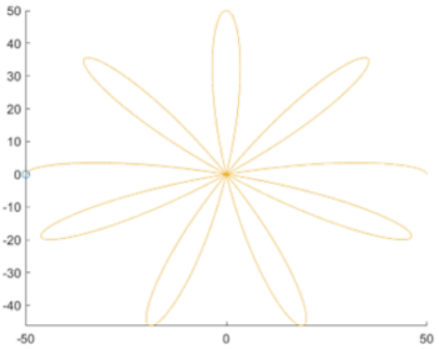


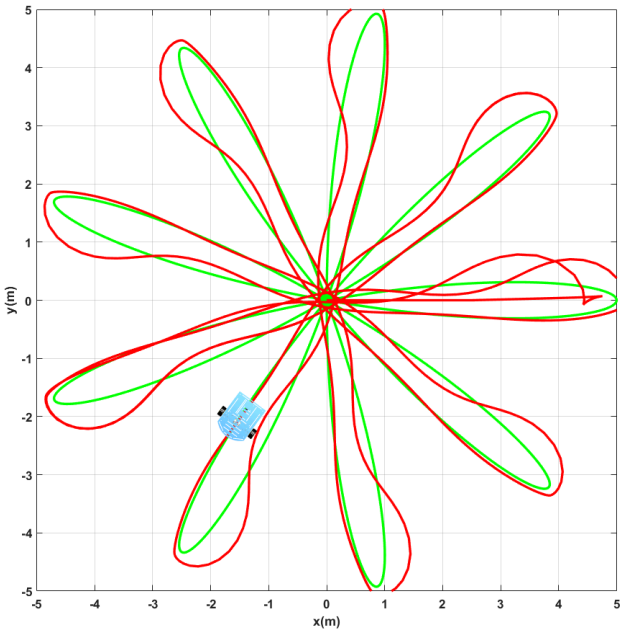

```
%FIGURA HEXAGONO
hxd = cos(t2)
hyd = sin(t2)
%Velocidades de la trayectoria deseada
hxdp= sin(t2)
hydp= cos(t2)
```



Análisis: En este caso, dado que la figura se obtuvo normalizando el vector de tiempo, la cantidad de muestras (periodo de muestreo) es muy pequeña lo que dificulta la simulación y por tanto el seguimiento de la trayectoria. Para corregir esto, podríamos realizarlo por medio de funciones parciales para así tener más muestras o puntos en el espacio.

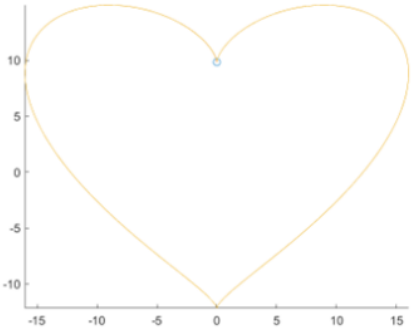
```
%FIGURA FLOR
hxd = 5*cos(9*t).*cos(t)
hyd = 5*cos(9*t).*sin(t)
%Velocidades de la trayectoria deseada
hxdp=-45*sin(9*t).*cos(t)-5*cos(9*t).*sin(t)
hydp=-45*sin(9*t).*sin(t)+5*cos(9*t).*cos(t)
```



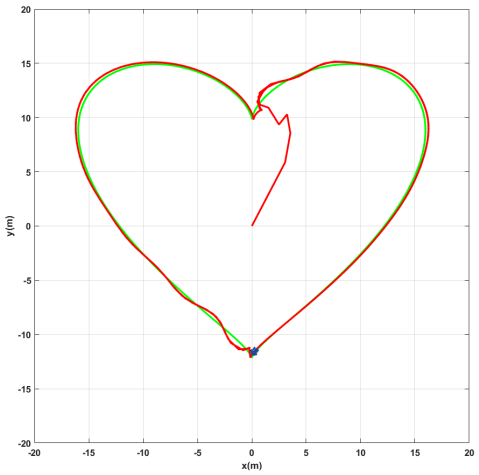


Análisis: En el caso de esta figura, el robot presenta dificultades para realizar el seguimiento de trayectoria. Esto puede deberse al tamaño de la figura y a los cambios abruptos de la misma. Para corregir esto, podríamos

aumentar el tamaño del vector de tiempo disminuyendo el periodo de muestreo, etc.




```
%FIGURA CORAZON
hxd=12*sin(t)-4*sin(3*t)
hyd=13*cos(t)-5*cos(2*t)-2*cos(3*t)-4*cos(16)
%Velocidades de la trayectoria deseada
hxdp= 12*cos(t)-12*cos(3*t)
hydp= -13*sin(t)+10*sin(2*t)+6*sin(3*t)
```



Análisis: Para el caso de esta figura, observamos un controlador considerablemente bueno, pues el robot sigue la trayectoria de una forma eficiente, exceptuando claro las complicaciones iniciales al intentar corregir la trayectoria.

…

Aquí es donde realizamos un control para el bucle de la simulación del control de robot, en cada iteración del bucle se están calculando los errores de control y se determina la matriz jacobiana donde se establecen las ganancias de control. Se calculan las velocidades deseadas y también aplicamos la ley de control para poder obtener las velocidades que son de entrada del robot, después se actualizan las posiciones y la orientación del robot con el modelo cinemático.

```
%4 CONTROL, BUCLE DE SIMULACION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k=1:N
```

```
    %a)Errores de control (Aqui la posición deseada ya no es constante,
    % varia con el tiempo)
    hxe(k)=hxd(k)-hx(k);
    hye(k)=hyd(k)-hy(k);

    %Matriz de error
    he= [hxe(k);hye(k)];
    %Magnitud del error de posición
    Error(k)= sqrt(hxe(k)^2 +hye(k)^2);
    %b)Matriz Jacobiana
    J=[cos(phi(k)) -sin(phi(k));... %Matriz de rotación en 2D
       sin(phi(k)) cos(phi(k))];
    %c)Matriz de Ganancias
    K=[15 0;...
       0 15];

    %d)Velocidades deseadas
    hdp=[hxdp(k);hydp(k)];
    %e)Ley de Control:Agregamos las velocidades deseadas
    qpRef= pinv(J)*(hdp + K*he);
    v(k)= qpRef(1);    %Velocidad lineal de entrada al robot
    w(k)= qpRef(2);    %Velocidad angular de entrada al robot
```

Se van a ir calculando las componentes de la velocidad lineal del robot en las direcciones que se tiene en x e y, ademas de la velocidad lineal del robot y la orientación en ese momento. Se integran las velocidades lineales para obtener las coordenadas y posición, con el metodo Euler para la integración númerica donde x1(k) y y1(k) son las coordenadas de posición en el tiempo de k. Finalmente se van a ir actualizando las coordenadas del punto de control con las coordenadas de posición del robot.

```
%5 APLICACIÓN DE CONTROL AL ROBOT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %Aplico la integral a la velocidad angular para obtener el angulo "phi"
de la orientación
   phi(k+1)=phi(k)+w(k)*ts; % Integral numérica (método de Euler)

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

   xp1=v(k)*cos(phi(k));
   yp1=v(k)*sin(phi(k));
   %Aplico la integral a la velocidad lineal para obtener las cordenadas
   %"x1" y "y1" de la posición
   x1(k+1)=x1(k)+ ts*xp1; % Integral numérica (método de Euler)
   y1(k+1)=y1(k)+ ts*yp1; % 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
```