CONTROL Y PROGRAMACIÓN DE ROBOTS

Proyecto de robótica móvil

Grado en Ingeniería Electrónica, Mecatrónica y Robótica

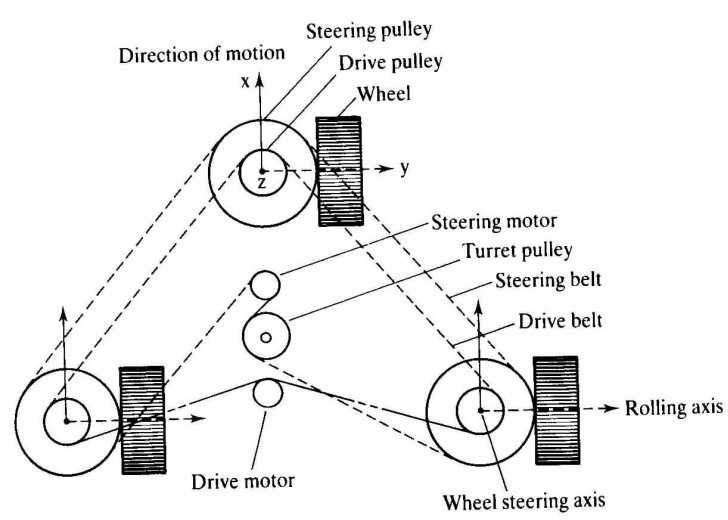
Autores: Montes Grova, Marco Antonio
Lozano Romero, Daniel
Mérida Floriano, Javier

Índice

1. Introducción al proyecto

En el proyecto que sigue a continuación se desarrollará el modelado y control de un robot móvil tipo síncrono. La principal característica destacable de este tipo de robots recae en el mecanismo mecánico interno que posee mediante el cuál se podrán mover 3 ruedas empleando únicamente 2 motores.
Con uno de los motores se desplazará en línea recta y con el otro se le dará el ángulo de giro deseado sobre sí mismo.

El modelo síncrono se basa en 3 ruedas idénticas que se desplazan y giran al unísono, haciendo posible que el robot pueda moverse en cualquier dirección y orientación en el plano del suelo. Estas ruedas están dispuestas de forma triangular en la base del robot, el cual se va a representar como un objeto cilíndrico, siendo la base uno de los lados circulares. Par un mayor entendimiento, se va a representar un esquema del mecanismo utilizado en este tipo de robot móvil:



Este modelo tiene como ventajas la separación de motores entre traslación y rotación que facilita el control, la garantía de control en trayectorias rectilíneas debido a la mecánica del mismo y la facilidad que incluyen las restricciones homólogas que posee. Como desventaja encontramos el complejo diseño mecánico que posee para poder transmitir la rotación y traslación a las tres ruedas simultáneamente y, por ende, su difícil implementación en la realidad.

2. Análisis cinemático

2.1. Introducción teórica y simplificación del modelo

Gracias a la mecánica del modelo síncrono, los cálculos para la obtención del modelo cinemático del mismo van a ser sencillos, ya que se trata de un modelo con restricciones holónomas, es decir, el comportamiento del vehículo se puede representar a través de sus variables generalizadas:

- Las variables cartesianas X y Y como coordenadas del centro del robot en el plano del suelo.
- La variable angular φ como la rotación producida frente al estado inicial, es decir, cuando éste vale cero.

las cuales son integrables en el tiempo de tal modo que resulta sencillo la obtención de un modelo cinemático de posiciones y velocidades.

La postura del robot, se podría obtener a partir de la aplicación de una matriz de rotación en torno al eje perpendicular al plano de suelo, es decir, respecto al eje Z:

$$R(\varphi) = \begin{bmatrix} cos\varphi & sin\varphi & 0 \\ -sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Además de ello, cabe destacar que, para simplificar el modelado del robot, se han asumido una serie de restricciones cinemáticas en las ruedas cómo son:

- Movimiento en un plano horizontal.
- Las ruedas poseen un único punto de contacto.
- Las ruedas no son deformables.
- v_c en el punto de contacto será nulo.
- La rueda no resbala, sino que produce un movimiento de arraste o deslizamiento.
- No habrá fricción en el punto de contecto.
- Los ejes de dirección serán en todo momento ortogonales a la superficie.
- Las ruedas están conectadas al bastidor de giro.

2.2. Obtención del modelo cinemático directo y su Jacobiano

Para el modelo cinemático directo en cuestión, sólo se necesita como dato el valor del radio de las ruedas, R, para poder realizar la conversión entre velocidad angular y velocidad lineal. En este trabajo, siendo el grupo de trabajo número 11, se tiene cómo radio de las ruedas el valor de $R = 0{,}4m$. Con esto ya se puede hallar el modelo cinemático del robot:

Para comenzar se deben definir las variables generalizadas y de actuación del sistema, respectivamente:

$$q = \begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} \qquad\qquad p = \begin{bmatrix} \dot{\theta} \\ \omega \end{bmatrix}$$

Como ya vio en las clases de la asignatura, las variables generalizadas son las que dan información sobre el estado del robot, ya que se encuentran en ellas las coordenadas cartesianas X e Y del plano del centro del robot y el ángulo de giro φ con respecto al eje X, como se ha decidido definir en este proyecto. Así mismo, las variables de actuación son, como dice su nombre, las variables que reflejarán el valor de movimiento de los motores del robot, es decir, del motor encargado del desplazamiento (velocidad angular de desplazamiento, $\dot{\theta}$) y del encargado de la rotación (velocidad angular de rotación, ω).

Una vez detectadas las variables del sistema, se deberá definir la relación entre ellas y, de ese modo,obtener la expresión del Jacobiano, es decir, la relación entre la derivada de las variables articulares de salida y de entrada.
En primer lugar, se definirá v como la componente global de velocidad de desplazamiento, por tanto, se calculará como:

$$v^2 = \dot{x}^2 + \dot{y}^2 \text{ o como } v = R\dot{\theta}$$

y, a su vez, la componentes cartesianas de la velocidad se definirán como:

$$\dot{x} = vcos(\varphi)$$
$$\dot{y} = vsin(\varphi)$$

Sabiendo además que la velocidad de rotación $\dot{\varphi}$ concuerda con la variable de actuación ω, se puede montar la matriz Jacobiana del sistema del mismo que se dijo anteriormmente, es decir:

$$J = \begin{bmatrix} Rcos(\varphi) & 0 \\ Rsin(\varphi) & 0 \\ 0 & 1 \end{bmatrix}$$

La matriz Jacobiana describirá la cinemática directa del sistema relacionando las variables de actuación con las derivadas de las variables generalizadas y, como se sabe que serán integrables debido a la existencia de restricciones holónomas, se podrán integrar para poder hallar los valores de posición y orientación.

Dicho esto, se puede expresar el modelo cinemático como función de entradas (variables de actuación) y salidas (variables generalizadas):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} Rcos(\varphi) & 0 \\ Rsin(\varphi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \omega \end{bmatrix} \tag{2}$$

e integrando respecto del tiempo éstas matrice,se obtendrá:

$$\begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \varphi_0 \end{bmatrix} + \begin{bmatrix} \int_0^t R\dot{\theta}cos(\varphi)dt \\ \int_0^t R\dot{\theta}sin(\varphi)dt \\ \int_0^t \omega dt \end{bmatrix} \tag{3}$$

dónde x_0, y_0 y φ_0 serán las condiciones iniciales de posición y orientación.

2.2.1. Esquema de simulación del modelo directo obtenido

Una vez obtenido el modelo cinemático directo se implementará en una función en Matlab, la cuál servida para modelar el comportamiento del movimiento del robot. Por tanto, el modelo cinematico directo se implementará del siguiente modo:

Listing 1: Implementación del modelo cinemático directo

```
%%MODELO CINEMATICO DIRECTO ROBOT MOVIL SINCRONO
function gener_out=MCD_movil(in)
     R=0.4; %-> Radio de la rueda [m]

    tetha_d=in(1);  % Velocidad de desplazamiento
    omega=in(2);    % Velocidad de rotacion
    phi=in(3);      % Angulo del robot

    % Jacobiano de velocidades
    jac=[R*cos(phi) 0;
         R*sin(phi) 0;
            0       1];

    % Vector de variables actuadoras de entrada
    act=[tetha_d;
         omega];

     % Definicion de salidas
     x_d=jac(1,:)*act;
     y_d=jac(2,:)*act;
     phi_d=jac(3,:)*act;

     gener_out=[x_d;y_d;phi_d];
end
```

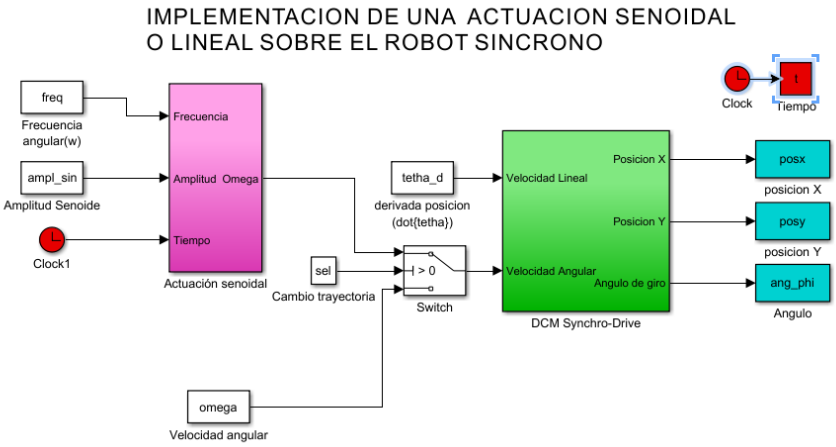


Figura 1: Esquema global

A continuación, se mostrará el montaje completo en Simulink que ha sido utilizado para obtener los resultados de la simulación.

En éste montaje existen dos subsistemas, los cuales se mostrarán a continuación. El primero de ellos, será el encargado de generar una actuación senoidal sobre el sobor, de tal modo que siga una trayectoria de éste tipo.

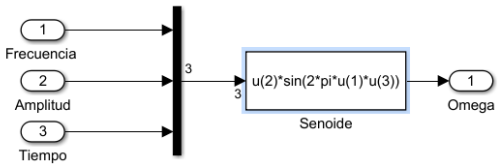


Figura 2: Esquema del bloque *Actuación Senoidal*

El segundo de ellos, será la implmentación del modelo cinemático directo del robot en un bloque, en el cuál se relamientará la variable angular, la cual es una variable de entrada y de salida del modelo y, además de ello, se saturarán las actuaciones.

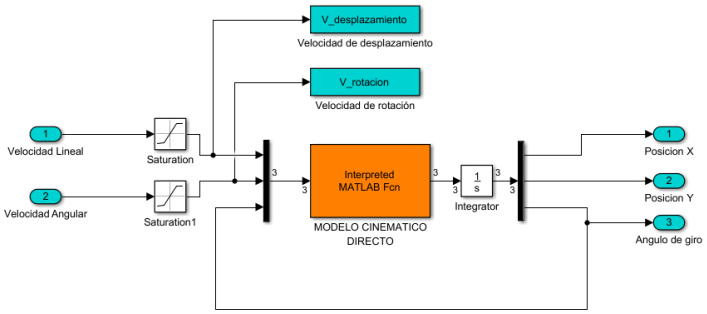


Figura 3: Esquema del bloque *DCM Synchro-Drive*

Cabe destacar cómo se saturarán las variables del modelo; la velocidad angular, se limitará a aproximadamente $\pm$ 15 grados por segundo, ya que es el angulo máximo aproximado al que teóricamente se puede girar un volante. En cuando a la velocidad lineal, se limitará aproximadamente a 30 centímetros por segundo, que correspondería aproximadamente a 1 km/h, lo cuál se considera una velocidad suficiente para un robot móvil.

Antes de introducir los experimentos realizados, cabe destacar que en un anexo se encontrarán los códigos de programación implementados para realizar dichos experimentos.

2.2.2. Experimentos realizados al modelo para su verificación

Con este código se pueden realizar varios experimentos. Para todos ellos las características comunes serán que partirán de las mismas condiciones iniciales para las variables generalizadas ($x = 0, y = 0, \varphi = 0$) y tendrán las mismas saturaciones, las cuales hemos supuesto como realistas para este tipo de robot.

- **Primer experimento**
  El primer experimento que se va a mostrar es el de una velocidad de desplazamiento máxima y velocidad de rotación nula. Como la simulación dura 30 segundos, el robot deberá recorrer una distancia de 9 metros:

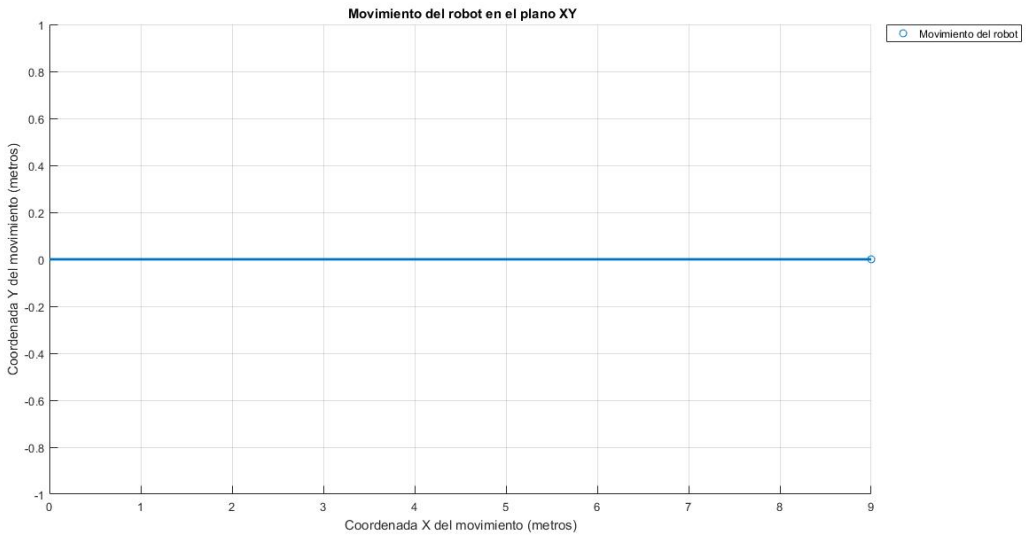


Figura 4: Movimiento en el plano con actuación de desplazamiento constante y rotativa nula

- **Segundo experimento**
  El siguiente experimento combinará ambas variables de actuación, estableciendo el valor máximo en ambas, por lo que el robot deberá dar vueltas en círculo. Como la velocidad de rotación máxima es de 15 grados/segundo y la simulación dura 30 segundos, el robot deberá dar 1 vuelta y 1/4 de vuelta más. En este y el siguiente experimento, al no constar únicamente de una trayectoria rectilínea, se ha aprovechado para representar, a través de flechas, el vector velocidad de desplazamiento:

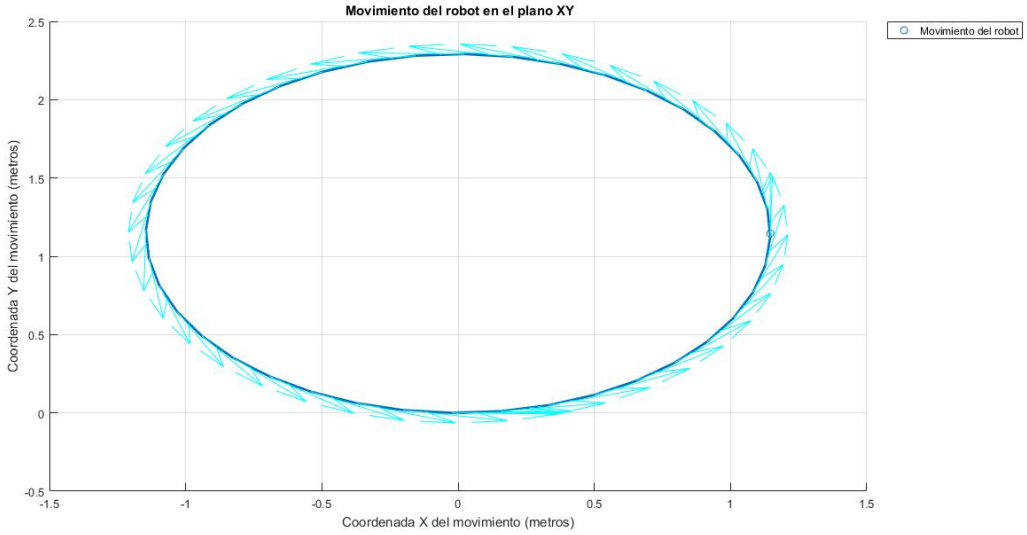


Figura 5: Movimiento en el plano con actuación de desplazamiento y rotativa constantes

- **Tercer experimento**
  Ahora se va a realizar el experimento correspondiente a la prueba solicitada para el trabajo donde se ha de administrar una velocidad de desplazamiento constante y una velocidad de rotación con carácter senoidal. Para ello, se ha introducido una velocidad de desplazamiento de 0.5 radianes/segundo y una senoide de 0.05 Hz de frecuencia y 0.2 radianes/segundo de amplitud:

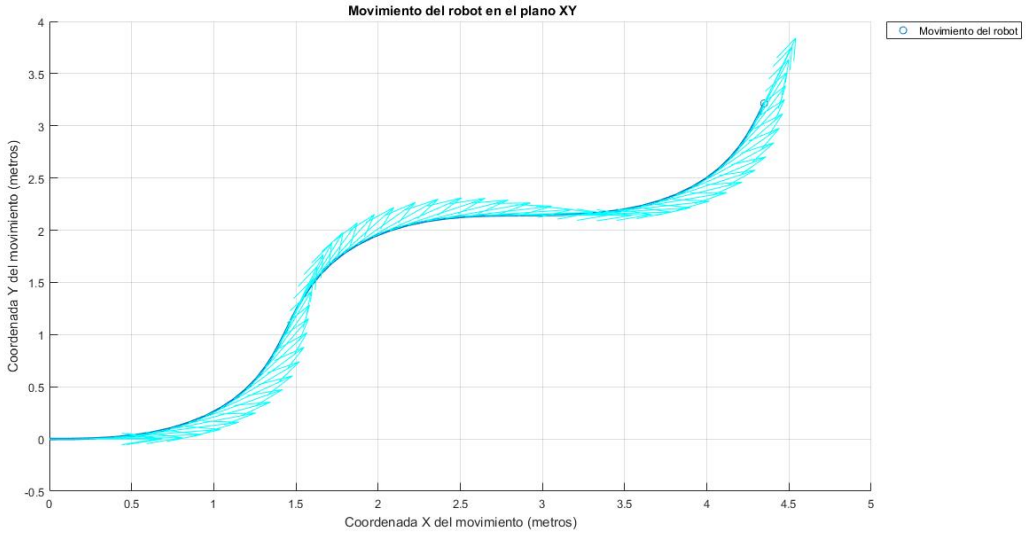


Figura 6: Movimiento en el plano con actuación rotativa senoidal

En este último experimento se puede observar que la variable de actuación ω resulta la senoide esperada y que, como resultado, se obtiene también una trayectoria senoidal, cómo se observa en la figura número 6.

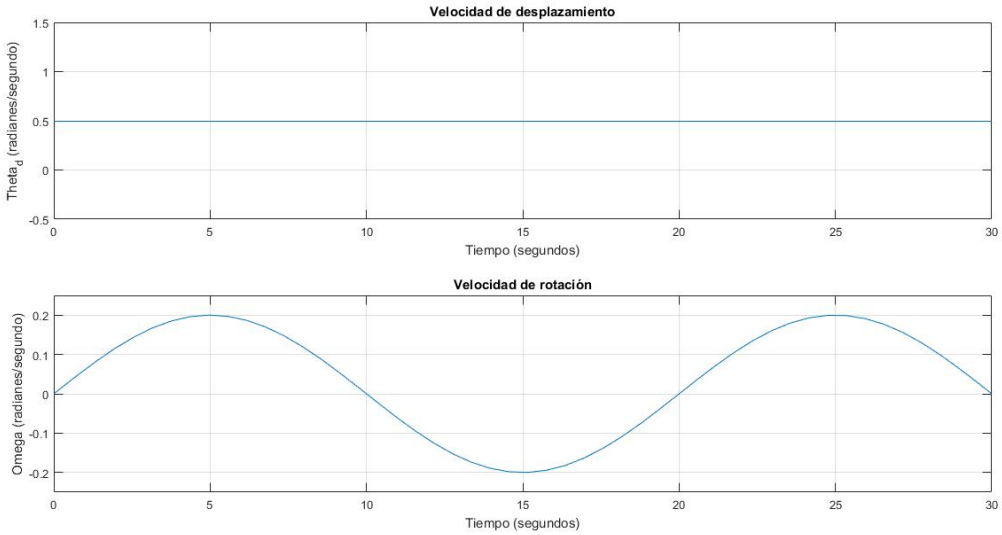


Figura 7: Variables de actuación tras saturación (arriba $\dot{\theta}$ y abajo ω)

2.3. Obtención del modelo cinemático inverso

Una vez comentado todo lo que concierne a la cinemática directa, se puede pasar a obtener el modelo cinemático inverso, el cual será de utilidad para la creación de trayectorias con las variables generalizadas y su posterior transformación a variables de actuación que harán que el robot siga dicha trayectoria.

Dicho esto, el modelo cinemático inverso se basa principalmente en la Jacobiana inversa, ya que ahora se desea relacionar las variables generalizadas como entradas con las variables de actuación como salida. Por ello únicamente hay que coger la ecuación del modelo cinemático directo y pasar el Jacobiano al otro lado, es decir, poner su inversa.

En este caso se da que el Jacobiano del modelo directo no es una matriz cuadrada, por lo que no puede tener inversa, así que habrá que realizar la pseudoinversa del mismo. Para ello, no es necesario realizar los cálculos paso por paso, ya que MATLAB posee una función que calcula la pseudoinversa por el método Moore-Penrose llamada *pinv*. Dicho esto, una vez utilizada la función y habiendo simplificado, el resultado es:

$$J^{-1} = \begin{bmatrix} \frac{cos(\varphi)}{R} & \frac{sin(\varphi)}{R} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Una vez obtenida la expresión del Jacobiano inverso, simplemente bastaría con introducirle los datos de entrada necesarios ($\dot{x}$, $\dot{y}$ y $\dot{\varphi}$) para obtener las salidas esperadas ($\dot{\theta}$ y ω).

Para garantizar el funcionamiento del modelo inverso, se va a realizar el tercer apartado del proyecto, donde se pide obtener las señales de control necesarias para que el robot realice una trayectoria parabólica, la cual va a venir dada por la expresión:

$$y = -\frac{x(x-A)}{D}$$

donde, X e Y son las coordenadas cartesianas de la posición del centro del robot, A una constante que determinará el valor máximo en x que alcanzará la parábola y D otra constante que determinará el valor máximo en y que alcanzará la parábola.

2.3.1. Esquema de simulación del modelo inverso obtenido

Para explicar como se ha llevado a cabo el proceso de obtención de los resultados, se va a hacer uso del montaje en Simulink utilizado en esta parte:

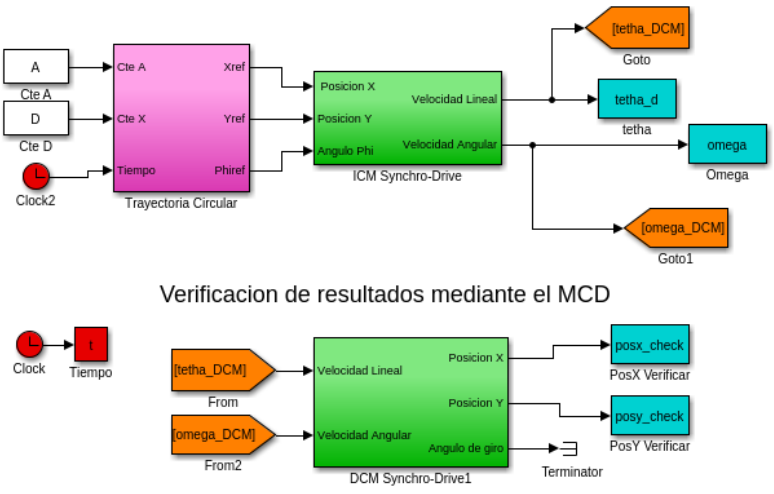


Figura 8: Esquema global

Como se pudo observar, las entradas a este bloque son las 2 constantes (A y D) de la parábola y el tiempo de simulación. Las dos primeras son utilizadas para la formulación de la parábola en sí y la tercera para la definición de X que crecerá linealmente pero a una escala 5 veces más pequeña que el tiempo de simulación.

Para trazar la trayectoria se ha implementado la siguiente función en matlab, la cuál se implementará en Simulink cómo de costumbre. La salidas de dicha función serán las que se ven en el esquema de Simulink mostrado anteriormente.

Listing 2: Implementación trayectoria parabolica

```
%%FUNCION PARA IMPLEMENTAR UNA PARABOLA
function [out]=tray_parab(in)
   % Defincion de entradas
    A=in(1);     %pto de corte final con la X
    D=in(2);     % altura parabola
    x=in(3);     %tiempo

    %Se escala la variable X
    x=x/5;
    % Definicion de la variable y
    y=-(1/D)*(x*(x-A));

    % definicion de las variables de salida
    phi_ref=atan2(y,x);

    out=[x;y;phi_ref];
end
```

Una vez se han definido los puntos de la trayectoria que ha de seguir el robot y con qué orientación ha de hacerlo, simplemente hay que pasar dichos datos al bloque donde se encuentra el modelo cinemático inverso:

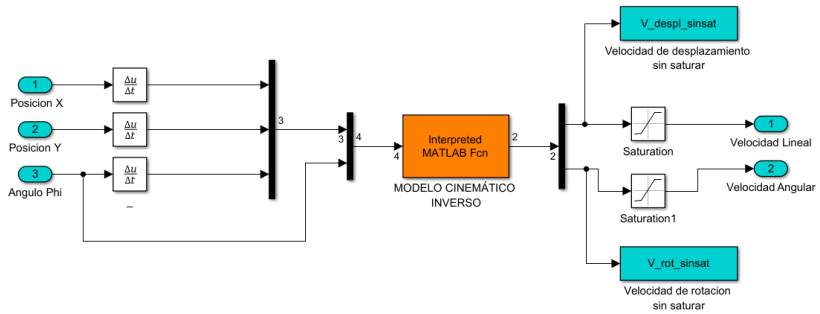


Figura 9: Bloque dónde se implementa la cinemática inversa del robot

Como se puede observar, lo primero que se debe hacer es derivar cada una de las entradas respecto al tiempo, puesto que el Jacobiano inverso, como antes lo hacía el directo, considera relaciones entre velocidades. Cabe destacar que, derivar computacionalmente no es una buena técnica, debido a que añade una componente elevada de error, sin embargo, en éste caso, dicha componente es asumible. Si en alguna otra ocasión fuese posible, sería mucho mas conveniente integrar una velocidad para conocer la posición que derivar una posición para conocer la velocidad.

Una vez hecho eso, se introducen los valores en el bloque que contiene a la función que ejecuta la cinemática inversa, cuyo código se muestra a continuación:

Listing 3: Modelo cinemático inverso del robot sincrono

```
%%MODELO CINMEATICO INVERSO ROBOT MOVIL SINCRONO
function gener_out=MCI_movil(in)
R=0.4; %-> Radio de la rueda [m]

x_d=in(1);      % Velocidad cartesiana X
y_d=in(2);      % Velocidad cartesiana Y
phi_d=in(3);    % Velocidad angular phi
phi=in(4);      % Angulo phi

% Jacobiano inverso de velocidades
jac_inv=[cos(phi)/R, sin(phi)/R, 0;
                  0,          0, 1];

%Vector de parametros generalizados
gen=[x_d y_d phi_d]';

% Definicion de salidas
tetha_d=jac_inv(1,:)*gen;
omega=jac_inv(2,:)*gen;

gener_out=[tetha_d;omega];
end
```

Se observa que simplemente se realiza la multiplicación matricial para obtener las variables de actuación.

Por último, antes de volver al esquema global, se saturan los valores de estas salidas con las mismas saturaciones que se habían determinado realistas en la cinemática directa.
Para terminar, en el esquema global, se coloca de nuevo el bloque de la cinemática directa para comprobar, con las variables de actuación obtenidas con el modelo inverso, se obtiene la misma trayectoria que se ha diseñado.

2.3.2. Experimentos realizados al modelo inverso para su verificación

Con el montaje en Simulink ya creado, lo único que falta es lanzar un script que asigne valores a las incógnitas de la simulación (A y D) y represente los datos:

Se debe tener en cuenta siempre que en la posición inicial hay que determinar la orientación inicial que debe tener el robot, lo cual se podrá obtener fácilmente como $atan(\frac{A}{D})$. Esto debe ser así porque, en caso contrario, al estar el robot en reposo con φ nulo, al principio de la trayectoria el valor de velocidad de rotación saturará intentando conseguir dicha orientación y la trayectoria no será la misma porque el sistema, con dicha saturación, no será capaz de realizarla en un tiempo tan pequeño. Así, para unos valores de $A = 3$ y $D = 10$, obtenemos:

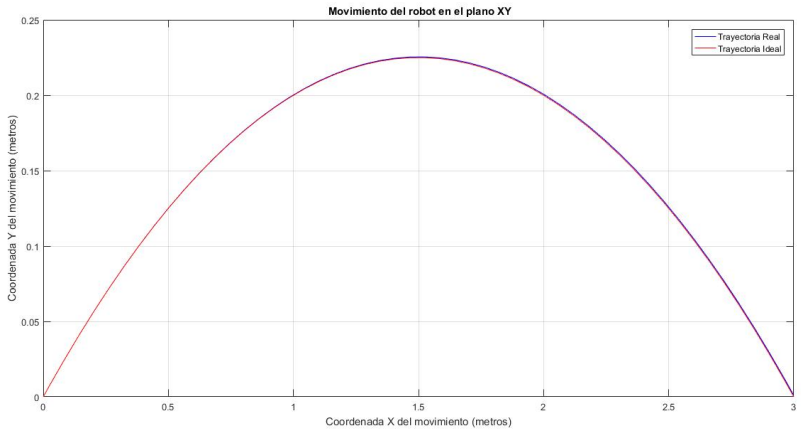


Figura 10: Comparativa entre la trayectoria pedida y la obtenida

Como se puede observar, en la gráfica comparativa anterior entre la trayectoria creada por funciones y la obtenida a partir de los valores de las variables de actuación sacadas del modelo cinemático inverso, los resultados son prácticamente iguales, por lo que se puede decir que el modelo inverso es aceptable.

Aparte, se van a mostrar las gráficas de los valores de las variables de actuación obtenidas con la cinemática inversa, así como el valor de la variable φ:

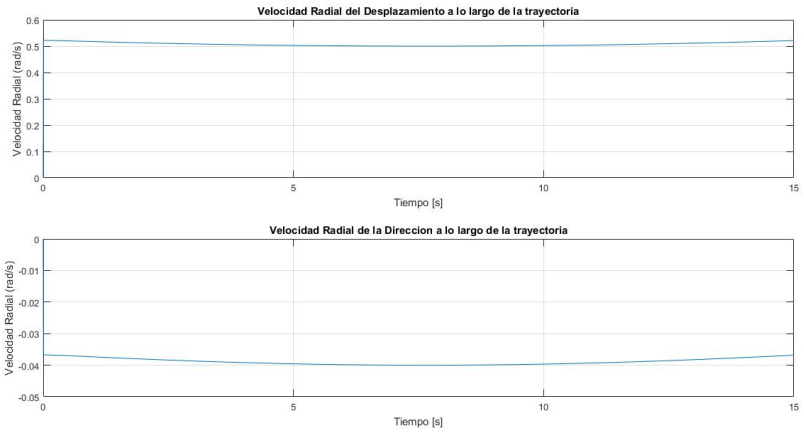


Figura 11: Valores de las variables de actuación

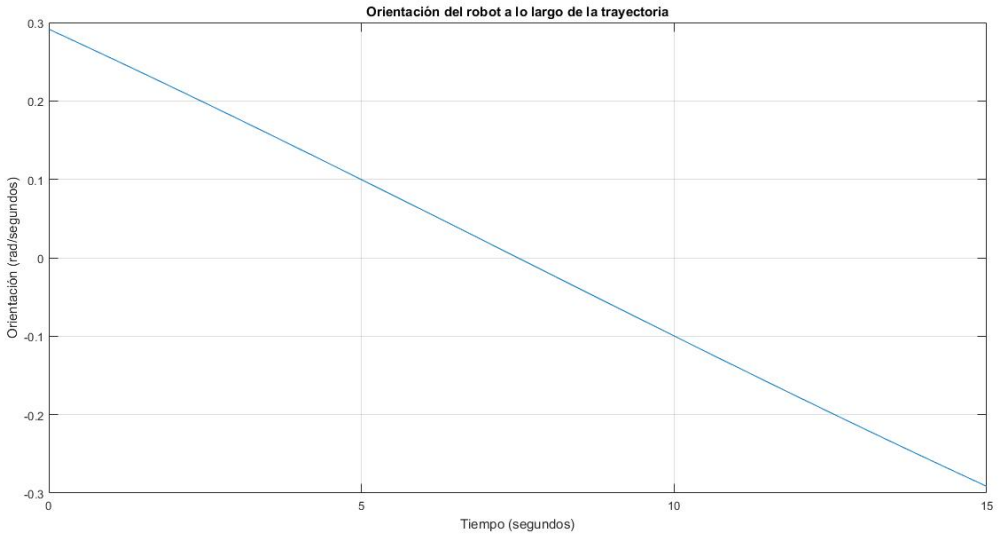


Figura 12: Valor de la orientación a lo largo de la trayectoria

Como se observa, en ambas velocidades no se produce saturación alguna, ya que están por debajo y por encima de sus respectivos valores de saturación. Además, con la gráfica de la orientación, se puede observar como justo a la mitad de la trayectoria, la misma es 0, ya que el robot se encuentra en paralelo al eje X en dicho momento.

Cabe añadir que es lo que pasaría si las variables de actuación saturasen, ya que no podrían realizar la misma trayectoria. Para ello se van a mostrar dos experimentos distintos. El primero de ellos se basa en bajar la saturación de $\dot{\theta}$ a la mitad, es decir, a ± 0.375:

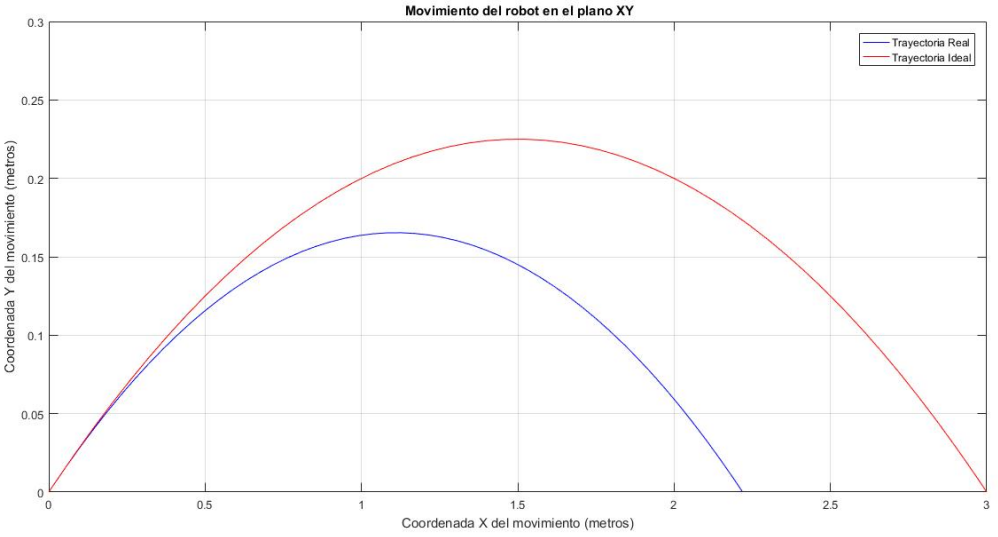


Figura 13: Comparativa de trayectorias con $\dot{\theta}$ saturada

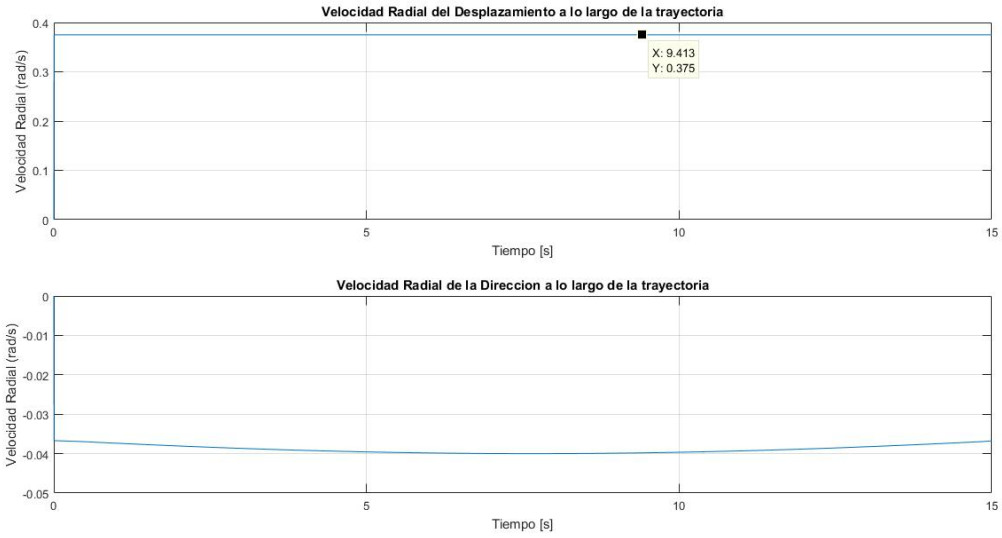


Figura 14: Valores de variables de actuación con saturación en $\dot{\theta}$

Como se puede observar, la trayectoria real que alcanza el robot se reduce tanto en X como en Y. Esto es lógico desde el punto de vista de la cinemática, puesto que al no poder alcanzar la velocidad necesaria, no llegará a ninguno de los valores objetivos en las coordenadas cartesianas. En la figura anterior se puede ver como $\dot{\theta}$ satura mientras que ω no.

Ahora se probará a saturar la velocidad de rotación, reduciendo el límite original a una décima parte, obteniendo una nueva saturación de ± 0.02618:

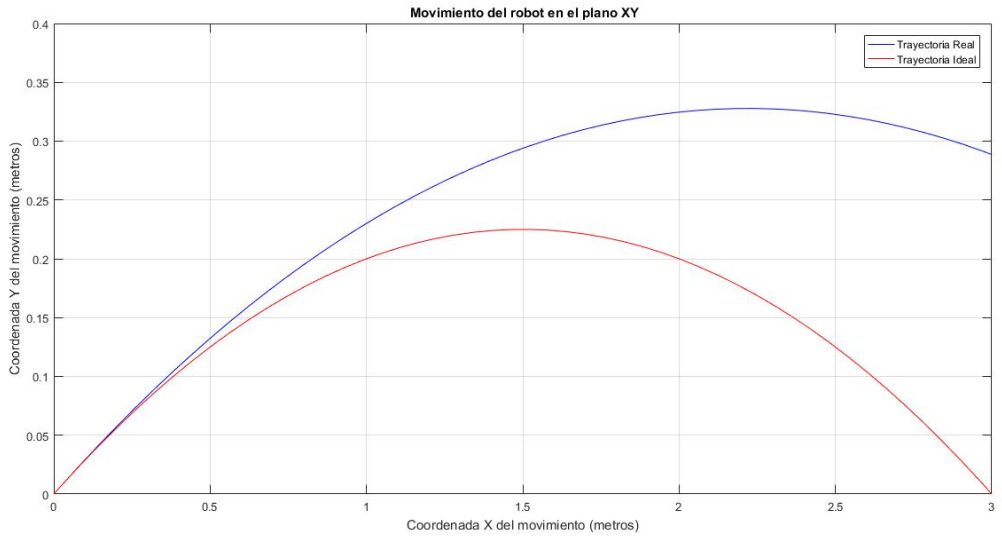


Figura 15: Comparativa de trayectorias con ω saturada

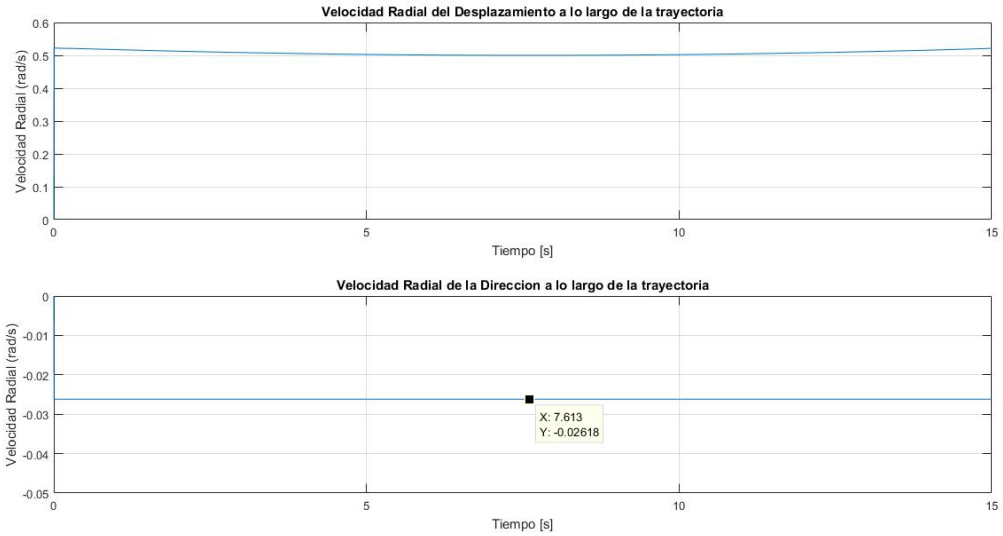


Figura 16: Valores de variables de actuación con saturación en ω

Aquí, como también era de esperar, el robot no puede seguir la trayectoria especificada, aunque esta vez lo hace de forma distinta. Esto se debe a que tiene la velocidad de desplazamiento necesaria para alcanzar los puntos objetivos pero no la capacidad de rotar lo suficientemente rápido como para seguirlos y por ello, se sobrepasa de sus objetivos. En la figura 17 se observa la saturación en ω.
El efecto que se produce al haber saturaciones en ambas actuaciones es una combinación de ambos resultados, es decir, el "no poder girar a tiempoz el "no avanzar lo suficientemente rápido":

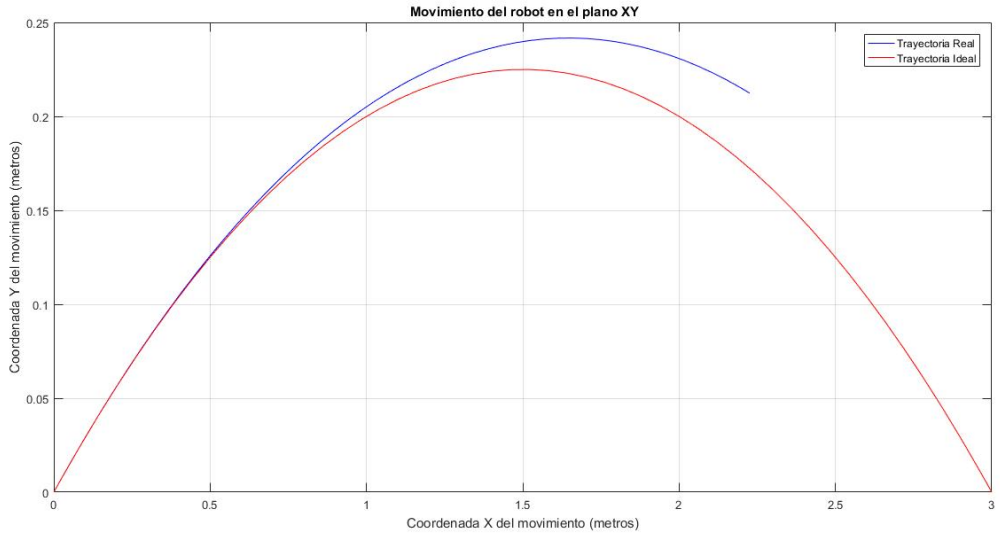


Figura 17: Comparativa de trayectorias con ω y $\dot{\theta}$ saturadas

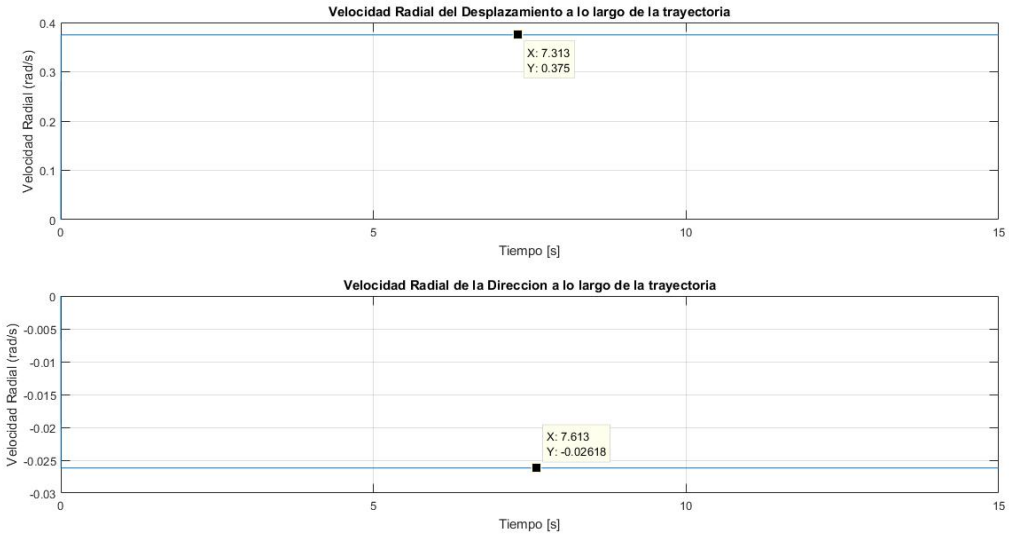


Figura 18: Valores de variables de actuación con saturación en ω y $\dot{\theta}$

3. Control dinámico

En lo que a la implementación de un algoritmo de control dinámico sobre el robot concierne, en primer lugar será necesario compltear el modelo añadiendo la dinámica de los actuadores.
Los actuadores del robot serán 3 motores de corriente continua, cuya función de transferencia que relaciona la entrada de tensión del motor con la salida en velocidad del mismo se define del siguiente modo:

$$G(s) = \frac{v(s)}{u(s)} = \frac{K}{\tau s + 1} \tag{5}$$

Será necesario definir el valor de la ganancia de la función de transferencia y la constante de tiempo, los cuales se obtendrán experimentalmente y unificarán toda la dinámica del motor, es decir,momento de inercia del motor, coeficiente de fricción viscosa, constante de fuerza contra-electromotriz, etcétera.

3.1. Implementación de diversos algoritmos de control

3.1.1. Control a un punto

En este apartado se va a realizar el control de movimiento a un punto, interesando únicamente que el robot consiga alcanzar dicho objetivo, sin importar orientación y sin restricciones temporales.

Para comenzar se hablará de la estrategia de control, la cual se basa en la teoría dada en clase con el uso de la ley de control proporcional:

$$v^* = K_v\sqrt{((x^* - x)^2 + (y^* - y)^2)} \tag{6}$$

$$\omega = K_h(\varphi^* - \varphi), \tag{7}$$

donde v es la velocidad lineal de desplazamiento en control, K_v la constante de proporción que se deberá ajustar, (x^*, y^*) el punto objetivo, (x, y) el punto actual, ω el valor de velocidad angular de orientación en control, K_h la constante de proporción para la velocidad de rotación que deberá ser siempre mayor que 1, φ^* el ángulo objetivo en cada momento de la trayectoria para llegar al punto objetivo que se calcula como $\varphi^* = atan(\frac{y^* - y}{x^* - x})$ y φ el ángulo de rotación actual.

Dicha ley de control proporcional se ha implementado en una función de MATLAB a la que se le introducen las referencias de posición así como la realimentación de posiciones y orientación actuales, y se le saca las velocidades de desplazamiento y rotación que han de ser aplicadas al robot.

Para la comprobación de este controlador se ha realizado un script de pruebas donde se pueden introducir los valores de (x^*, y^*) y decidir si el robot, en la posición inicial que parte de (0,0), va a estar orientado hacia el punto objetivo o no, en cuyo caso la variable φ valdrá también 0.
Se mostrarán entonces, a continuación, los resultados obtenidos con la introducción de distintos puntos objetivos:

- **Punto objetivo (0,5) sin preorientación**

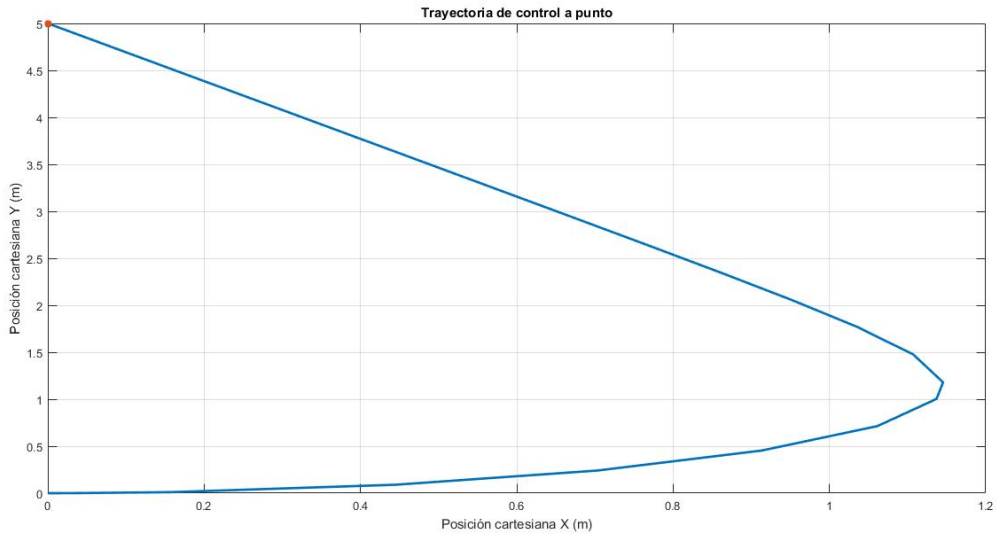


Figura 19: Trayectoria realizada en control de un punto

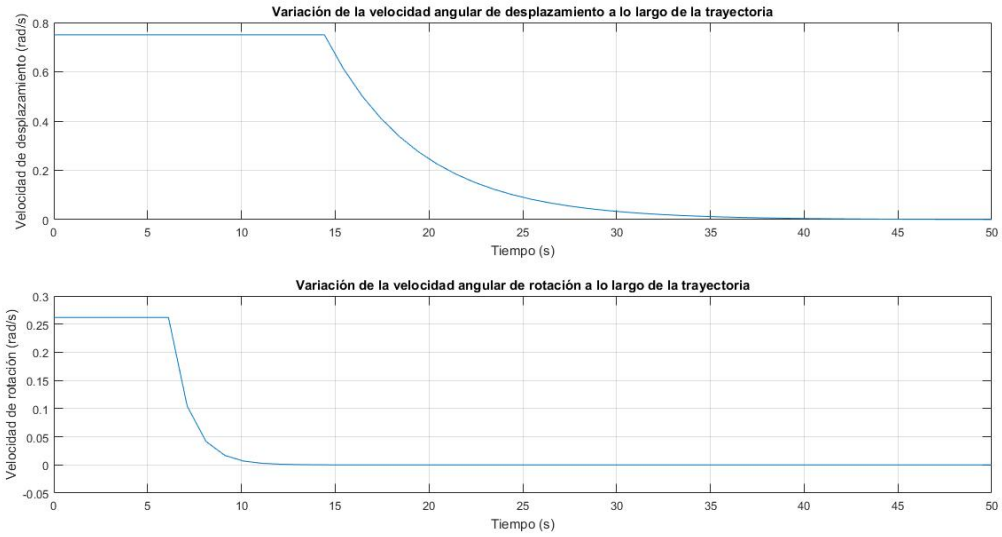


Figura 20: Velocidades a lo largo del tiempo de la trayectoria (arriba $\dot{\theta}$ y abajo ω)

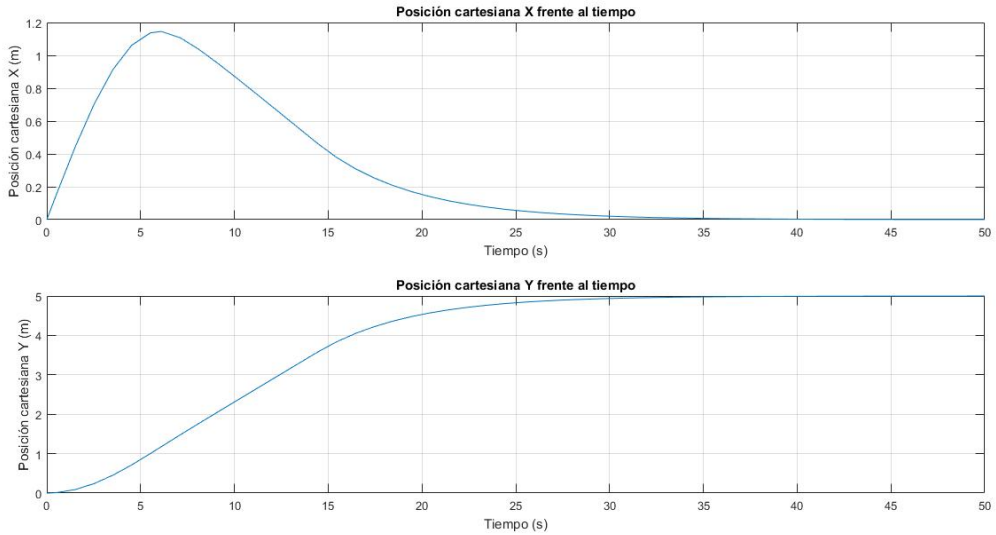


Figura 21: Posiciones del robot a lo largo de la trayectoria respecto al tiempo

Como se puede observar, al no encontrarse el robot orientado hacia el punto objetivo en la posición inicial, para llegar al destino deberá realizar un giro aproximadamente 180 grados.
Se puede observar también como al principio no es capaz de girar rápidamente hacia el objetivo, y esto se debe a la saturación que se ha establecido en la misma pues, como se muestra en la gráfica de velocidades, los primeros 6 segundos, se encuentra girando al máximo de velocidad. Esto se encuentra también en la gráfica de la posición X frente al tiempo, donde en el segundo 6 el robot empieza a volver al valor de x^* solicitado una vez el robot ya ha conseguido orientarse y no hace falta que le aplique la máxima velocidad de rotación.
También se puede concluir que el robot se queda en el punto objetivo una vez alcanzado puesto que en las gráficas se puede observar que llega a éste antes de que termine la simulación.

- **Punto objetivo (0,5) con preorientación**

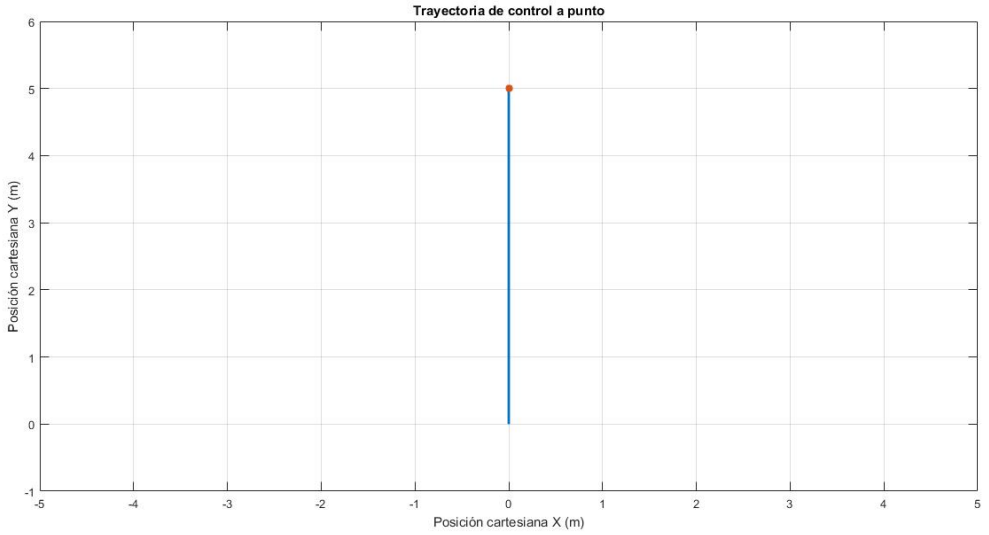


Figura 22: Trayectoria realizada en control de un punto

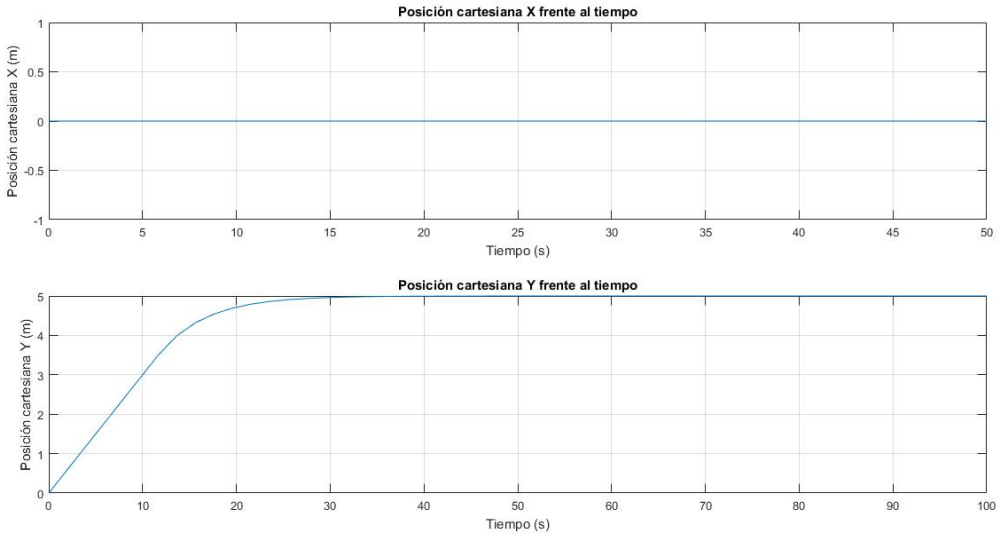


Figura 23: Posiciones del robot a lo largo de la trayectoria respecto al tiempo

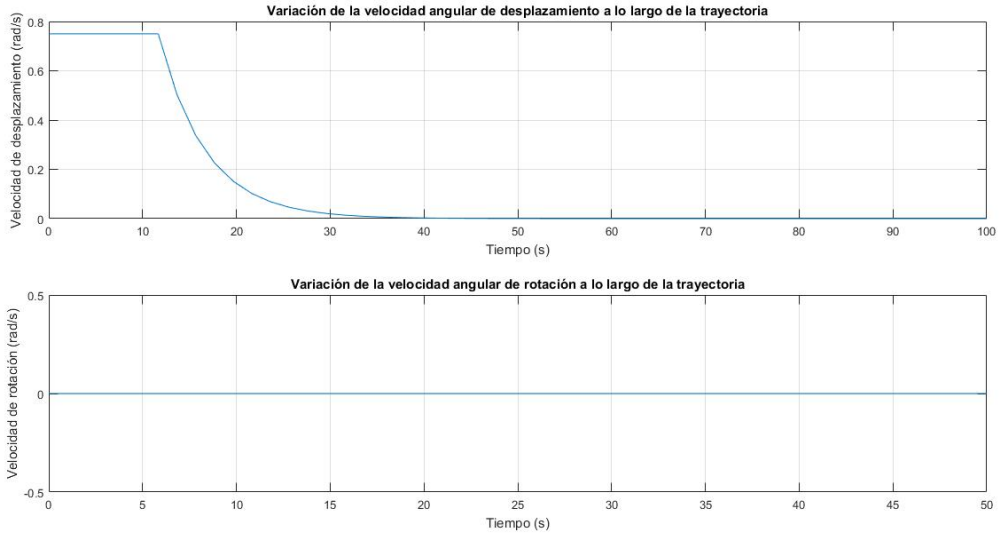


Figura 24: Velocidades a lo largo del tiempo de la trayectoria (arriba $\dot{\theta}$ y abajo ω)

Este caso es igual que el anterior pero habiendo realizado preorientación en la posición inicial hacia el punto objetivo. Es por esto que el robot únicamente deberá seguir una línea recta, puesto que ya se encuentra orientado y únicamente hará falta que actúe la variable de velocidad de desplazamiento $\dot{\theta}$.

Se pueden realizar más experimentos pero éstos no dan más información, puesto lo único que hay que tener en cuenta al darle dicho punto es que se ha dejado un tiempo de simulación suficiente para que el robot pueda llegar al objetivo y que con las velocidades máximas implementadas es capaz de hacerlo en dicho tiempo.

3.1.2. Control a una línea

En este apartado se va a estudiar el control de seguimiento de una línea por parte del robot. Para ello debemos conocer en primer lugar la recta que se desea seguir, la cual vendrá descrita como $ax+by+c=0$ Utilizando entonces la ley de control proporcional para el ángulo de orientación, se deberá implementar el control descrito a continuación:

$$\omega = -K_d d + K_h(\varphi^* - \varphi), \tag{8}$$

donde K_d es la constante de proporción para el control de la distancia a la que se encuentra el robot de la línea a seguir, K_h la constante de proporción para el control de la orientación respecto a la línea, φ^* y φ los ángulos de orientación de referencia, calculada como $\varphi^* = atan(\frac{-a}{b})$, y actual, y d la distancia del robot a la línea definida como $d = \frac{(a,b,c)\cdot(x,y,1)}{\sqrt{a^2+b^2}}$.
Como se puede apreciar en este tipo de control no se controla la velocidad de desplazamiento pues no es necesario, por lo que se le asigna un valor constante.

Una vez descrito el funcionamiento del control, implementado en una función de MATLAB donde se han puesto la restricciones de que las líneas siempre avanzarán en X positivo y que no podrán ser verticales para facilitarla implementación del mismo.
Una vez tenidas en cuenta dichas restricciones, se puede pasar a realizar experimentos que determinen la fiabilidad del control:

- **Recta a seguir con A=2, B=5 y C=3**

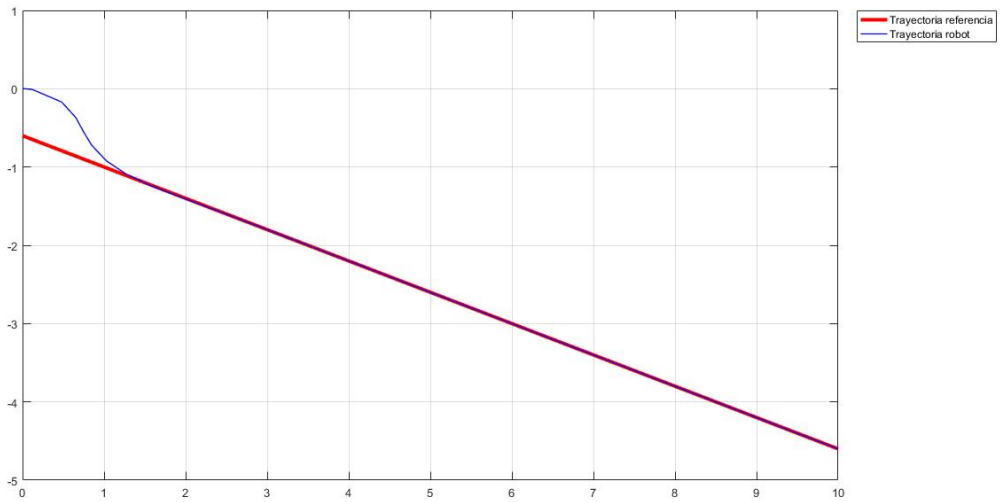


Figura 25: Trayectoria realizada en control de seguimiento de línea

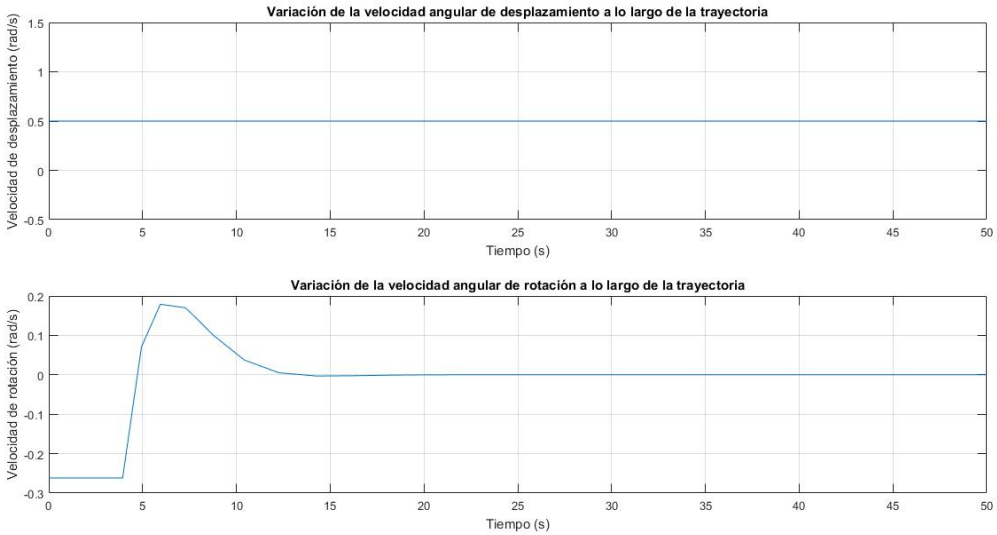


Figura 26: Velocidades de actuación a lo largo del tiempo de la trayectoria (arriba $\dot{\theta}$ y abajo ω)

3.1.3. Control a una trayectoria

3.1.4. Control a una postura

3.2. Ley de control *Persecución pura*

4. Anexos y conclusiones

4.1. Codigos de programacion

4.1.1. Pruebas del modelo cinematico directo

Listing 4: caption caption caption

```
%%SCRIPT EMPLEADO PARA REALIZAR PRUEBAS DE LA SIMULACION DEL MCD
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% En el script que sigue se implementara una trayectoria que siga las
% siguientes premisas:
% velocidad lineal -> tetha_d (sera contante)
% velocidad angular -> omega (constante o A*sin(w*t))
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

selection='Seleccione el tipo de trayectoria a implementar:\n 0.Lineal/Curva. \n 1.
    Senoidal.\n';
sel=input(selection);
while (sel >1)
    disp('Error. Parametro no valido\n')
    selection='Seleccione el tipo de trayectoria a implementar:\n 0.Lineal/Curva. \n 1.
        Senoidal.\n';
    sel=input(selection);
end

% %%%%%%%Posicion inicial del robot %%%%%%%
pos_init=[0;0;0];

```

```
% %%%%%%%Tiempo de simulacion %%%%%%%%
% Para simular el seno empelar un tiempo grande
% (aprox. 30 segundos, aunque depende de la frecuencia que se le meta)
t_sim=30;

% %%%%%%Saturacion en velocidades angulares y lineales %%%%%%
% No se gira un volante a mas de 10-15 deg/sec, por tanto, ahi estara la saturacion del
    movimiento
omega_sat=[-0.2618 0.2618];     % 15 grados/segundo
tetha_d_sat=[-0.75 0.75];       % Velocidad lineal de 30 cm/seg

% %%%%%%%SELECCION DEL TIPO DE TRAYECTORIA DESEADA %%%%%%%%%%
switch(sel)
    % SI SE DESEA QUE EL ROBOT SIGA UNA TRAYECTORIA LINEAL
    case 0
        selection='Asigne velocidad de giro del robot.\nEl rango posible es [-0.2618
            0.2618]\nVelocidad de giro introducida: ';
        omega=input(selection);
        if omega>omega_sat(2)
            omega=omega_sat(2);
            disp('Valor por encima de la saturacion superior. Se asignara el valor maximo
                posible')
        elseif omega<omega_sat(1)
            omega=omega_sat(1);
            disp('Valor por debajo de la saturacion inferior. Se asignara el valor minimo
                posible')
        end

        selection='Asigne velocidad de rotacion de las ruedas(velocidad lineal).\nEl
            rango posible es [-0.75 0.75]\nLa velocidad lineal introducida es: ';
        tetha_d=input(selection);
        if tetha_d>tetha_d_sat(2)
            tetha_d=tetha_d_sat(2);
            disp('Valor por encima de la saturacion superior. Se asignara el valor maximo
                posible')
        elseif tetha_d<tetha_d_sat(1)
            tetha_d=tetha_d_sat(1);
            disp('Valor por debajo de la saturacion inferior. Se asignara el valor minimo
                posible')
        end
        % Para evitar errores, se inicializan a cero el resto de variables
        freq=0; ampl_sin=0;

    % SI SE DESEA QUE EL ROBOT SIGA UNA TRAYECTORIA SENOIDAL
    case 1
        % Parametros senoides
        selection='Asigne frecuencia de la senoide: ';
        freq=input(selection); %Frecuencia de la senoide en la direccion
        selection='Asigne amplitud de la senoide: ';
        ampl_sin=input(selection); %Amplitud de la senoide en la direccion

         selection='Asigne velocidad de rotacion de las ruedas(velocidad lineal).\nEl
             rango posible es [-0.75 0.75]\nLa velocidad lineal introducida es: ';
        tetha_d=input(selection);
        if tetha_d>tetha_d_sat(2)
```

```
67              tetha_d=tetha_d_sat(2);
68              disp('Valor por encima de la saturacion superior. Se asignara el valor maximo
                    posible')
69          elseif tetha_d<tetha_d_sat(1)
70              tetha_d=tetha_d_sat(1);
71              disp('Valor por debajo de la saturacion inferior. Se asignara el valor minimo
                    posible')
72          end
73          % Para evitar errores, se inicializan a cero el resto de variables
74          omega=0;
75  end
76
77
78  % Se lanza la simulacion
79  sim('sl_MCD_sincrono');
80
81  % Se grafica el resultado obtenido
82  figure();hold on;...
83      comet(posx,posy);grid; title(' Movimiento del robot en el plano XY');...
84      xlabel('Coordenada X del movimiento (metros)'); ylabel('Coordenada Y del movimiento (
            metros)');...
85      legend('Movimiento del robot','Location','BestOutside');
86  % Graficamos el vector velocidad en cada pto de la trayectoria
87  plot(posx,posy,'LineWidth',2);
88  u=cos(ang_phi);
89  v=sin(ang_phi);
90  quiver(posx,posy,u,v,'c'); %Ploteo del vector de la velocidad Lineal.
91  hold off;
```

4.1.2. Pruebas del modelo cinematico inverso

Listing 5: caption caption caption

```
%%PRUEBAS DE SIMULACION DE LA TRAYECTORIA PARABOLICA A PARTIR DEL MCI
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% En el script que sigue se implementara una trayectoria a partir del
% jabociano del robot, es decir, del modelo de velocidades del mismo.
% Se buscara implementar una trayectoria parabolica
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;%clear all;

% %%%DEFINICION DEL TIEMPO DE SIMUACION %%%%%
% Para la simulacion de la actuacion senoidal introducir un tiempo de simulacion grande
% (aprox. 30 segundos, aunque depende de la frecuencia que se le meta)
t_sim=30;

% %%%%%%Posicion inicial del robot %%%%%%%
pos_init=[0;0;0.2915];%El valor de phi sera el valor del angulo sacado por la arcotangente de la
                      %derivada de la parabola en t=0

% %%%%%Saturacion en velocidades angulares y lineales %%%%%%
% No se gira un volante a mas de 10-15 deg/sec, por tanto, ahi estara la saturacion del movimiento
omega_sat=[-0.2618/10 0.2618/10];     % 15 grados/segundo
tetha_d_sat=[-0.75/2 0.75/2];       % Velocidad lineal de 30 cm/seg


% Descripcion de la entrada parabolica.
A=3;
D=10;

% Se lanza la simulacion
sim('sl_MCI_sincrono');

% Se grafica el resultado obtenido
figure();
subplot(2,1,1);
plot(t,tetha_d);grid; title('Velocidad Radial del Desplazamiento a lo largo de la trayectoria');
xlabel('Tiempo [s]'); ylabel('Velocidad Radial (rad/s)');
subplot(2,1,2);
plot(t,omega);grid; title('Velocidad Radial de la Direccion a lo largo de la trayectoria');
xlabel('Tiempo [s]'); ylabel('Velocidad Radial (rad/s)');

%Comprobamos Resultado
figure();
plot(posx_check,posy_check,'b',trayec_x,trayec_y,'r');grid; title(' Movimiento del robot en el plano XY');...
    xlabel('Coordenada X del movimiento (metros)'); ylabel('Coordenada Y del movimiento (metros)');...
    legend('Movimiento del robot','Location','BestOutside');
```