Álvaro Calvo Matos

Federico Vaz Fernández

Raúl Zahínos Marín

GIERM

27 de diciembre de 2019

Control y Programación de Robots

Robótica Móvil

**Índice**

[1 Introducción. 2](#_Toc28364199)

[2 Análisis cinemático. 3](#_Toc28364200)

[2.1 Modelo cinemático en su forma Jacobiana. 3](#_Toc28364201)

[2.2 Trayectorias de lazo abierto. 3](#_Toc28364202)

[2.3 Trayectoria parabólica genérica. 3](#_Toc28364203)

[3 Control del robot. 4](#_Toc28364204)

[3.1 Dinámica básica de los actuadores. 4](#_Toc28364205)

[3.2 Control punto a punto. 4](#_Toc28364206)

[3.3 Control en línea recta. 4](#_Toc28364207)

[3.4 Control en postura. 4](#_Toc28364208)

[3.5 Control mediante algoritmo de persecución pura para el seguimiento de una trayectoria senoidal. 4](#_Toc28364209)

[4 Conclusiones generales. 4](#_Toc28364210)

# Introducción.

El objetivo que se persigue con la realización de este trabajo es el de poner en práctica los conocimientos adquiridos en esta parte de la asignatura.

Para lograr controlar robots móviles se requieren solventar algunos problemas técnicos, comenzando por conocer la constitución del propio robot a controlar. Tipos de ruedas, disposición, grados de libertad, restricciones, actuadores…

El primer paso será el de elaborar un modelo cinemático que describa el movimiento del robot en función de los actuadores. Con él podemos además obtener el modelo cinemático inverso, y así conocer las referencias necesarias para alcanzar el estado o trayectoria que se desee.

Antes de controlar debemos tener en cuenta las dinámicas del robot móvil, aunque en el simulador que se construirá para el control solo se tendrán en cuenta las dinámicas de los actuadores. Es importante no perder de vista que cualquier control diseñado y probado en un simulador basado en modelos simplificados, se comportará, por lo general, peor en la realidad.

Para controlar necesitaremos primero una trayectoria que seguir. Generar una trayectoria supone conocer el terreno por el que nos vamos a mover. Para este ejercicio, supondremos que no existen obstáculos, y que disponemos de todo el plano XY libre para el movimiento de nuestro robot. Solo importará pues la propia trayectoria a seguir, la cual dependerá del movimiento que queramos conseguir.

Por último, necesitaremos sensores con los que poder medir y así cerrar el lazo de control. La naturaleza de estos sensores será crucial para el control del robot móvil, ya que de ellos depende la localización y cálculo de la orientación del robot bajo el sistema de referencia escogido.

El conocimiento de la posición de un robot a partir de los sensores que incorpore es uno de los grandes problemas del control de robots móviles. Aquí se supondrán sensores perfectos, sin ruidos ni errores, de forma que conocemos totalmente el estado actual del robot.

Los controles que se diseñarán serán principalmente controles proporcionales al error presente entre la posición actual del robot y la posición deseada, teniendo en cuenta además el ángulo entre el robot y la posición deseada.

# Análisis cinemático.

El robot con el que se trabajará presenta una configuración diferencial, cuyas ruedas izquierda y derecha, paralelas entre si y con ejes fijos, están actuadas y tienen la capacidad de girar en ambos sentidos.

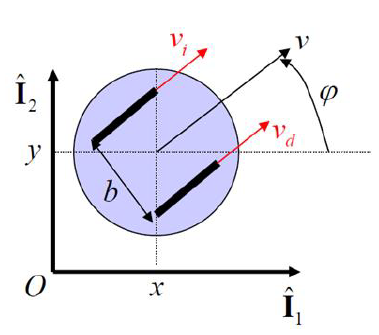


Figura 1. Configuración Diferencial.

Se trata de un robot subactuado, ya que dispone de dos únicos actuadores para el movimiento y la orientación en el plano XY.

Sus parámetros característicos son: b = 0.8 m y R = 0.4 m.

Posteriormente se decidirán el resto de características del robot en consecuencia a estos parámetros.

## Modelo cinemático en su forma Jacobiana.

El modelo cinemático del robot diferencial en forma jacobiana es el siguiente:

De las cuales:

* Variables generalizadas:
* Variables de actuación:

Estas ecuaciones forman el modelo cinemático directo del robot diferencial, y se han escrito dentro de una función de Matlab para disponer de un simulador de la cinemática de nuestro robot.

El código finalmente utilizado sería:

#### function [out] = Modelo\_Cinematico(in)

#### % Variables de actuación - Velocidad angular de las ruedas

#### w\_i=in(1);

#### w\_d=in(2);

#### 

#### % Variables generalizadas

#### x=in(3);

#### y=in(4);

#### phi=in(5);

#### 

#### % Datos geométricos

#### b = 0.8 ; % Distancia entre las ruedas

#### R = 0.4 ; % Radio de las ruedas

#### 

#### % Modelo cinemático directo - Forma Jacobiana

#### J\_q=[R/2\*cos(phi) R/2\*cos(phi);

#### R/2\*sin(phi) R/2\*sin(phi);

#### -R/b R/b ];

#### 

#### p\_d = [w\_i w\_d]';

#### 

#### q\_d =J\_q \* p\_d ;

#### 

#### out = q\_d; %[x\_d y\_d phi\_d];

#### end

## Trayectorias de lazo abierto.

En el siguiente apartado se nos pide alimentar al robot con unas actuaciones en velocidad sobre las ruedas constante y una actuación oscilatoria senoidal sobre la variable de dirección, para el caso del robot diferencial esto se traduce en una velocidad extra, senoidal y desfasada para cada rueda.

Como resultado de la simulación se obtienen las siguientes graficas:



Figura 2. Velocidades de actuación en lazo abierto.



Figura 3. Trayectoria resultante sobre el plano XY de la actuación en lazo abierto.

## Trayectoria parabólica genérica.

En este apartado se pide calcular las referencias que deberían de pasarse al robot diferencial para que siga una trayectoria parabólica.

Para ello se nos exige el uso de la jacobiana inversa como modelo de cinemática inversa. Este se ha calculado y expresado en la siguiente función de Matlab:

#### function [out] = Modelo\_Cinematico\_Inverso(in)

#### v = in(1);

#### phi\_d = in(2);

#### 

#### % Datos geométricos

#### b = 0.8 ; % Distancia entre las ruedas

#### R = 0.4 ; % Radio de las ruedas

#### 

#### w = [1/R -b/(2\*R); 1/R b/(2\*R)]\*[v phi\_d]';

#### 

#### w\_i = w(1);

#### w\_d = w(2);

#### 

#### out = [w\_i w\_d];

#### end

El modelo cinemático inverso calcula las velocidades angulares de cada rueda en función del vector velocidad v y de la derivada del ángulo . Para calcularlos se ha procedido a parametrizar la ecuación de una parábola genérica respecto al tiempo. Una vez hecho, se han derivado x e y, obteniendo las componentes horizontal y vertical del vector velocidad v. Derivando de forma similar se obtiene

Estos cálculos se han introducido dentro de una función única entrada es el tiempo.

#### function [out] = Generador\_de\_trayectorias(in)

#### t = in(1);

#### 

#### x\_d = 0.25;

#### A = 2;

#### x = x\_d \* t;

#### y\_d = (-2\*x + A)\*x\_d;

#### 

#### 

#### v = sqrt(x\_d^2 + y\_d^2);

#### phi\_d = -(2\*x\_d)/((A - 2\*x)^2 + 1);

#### 

#### w = Modelo\_Cinematico\_Inverso([v phi\_d]);

#### 

#### w\_i = w(1);

#### w\_d = w(2);

#### 

#### out = [w\_i w\_d];

#### end

El resultado de la simulación resulta bastante bueno como podemos comprobar en la siguiente figura:



Figura 4. Trayectoria parabólica en lazo abierto sobre el plano XY.



Figura 5. Referencia en velocidad obtenidas para el seguimiento de la trayectoria parabólica.

# Control del robot.

Probado el funcionamiento del simulador cinemático del robot es hora de comenzar a diseñar y probar controles. Se agregarán bloques al modelo de Simulink creado hasta ahora con el objetivo de mejorar un poco más el modelo y de implementar los controles necesarios para el cumplimiento de las diferentes especificaciones.

## Dinámica básica de los actuadores.

Con el objetivo de ser un poco más fieles a la realidad, se agregará la dinámica de los actuadores al simulador, de forma que se vean diferencias entre las referencias que se marcan y las velocidades que toma realmente el robot.

Se empleará para ello una función de transferencia de segundo orden con dos polos en el mismo lugar.

Se agregan además saturaciones en velocidad y aceleración para simular otros límites físicos de los actuadores. Se obtiene el siguiente esquema de Simulink que se añadirá a partir de ahora al simulador:

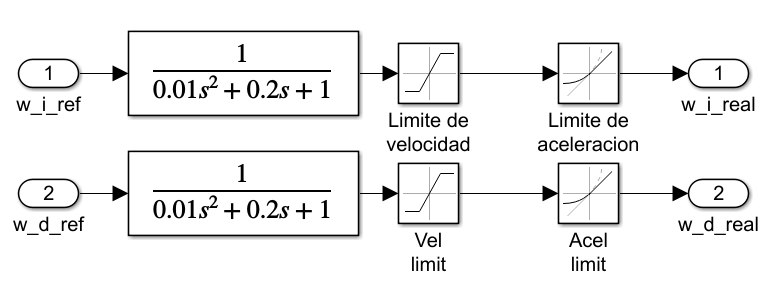


Figura 6. Simulación de la dinámica básica de los actuadores.

Destacar que tanto los valores de los polos como los limites para las saturaciones se han colocado a conveniencia, siguiendo un poco también la lógica conocidas las dimensiones físicas del robot.

## Control punto a punto.

Para llevar a cabo el control de punto a punto se miden la distancia desde el punto actual hasta el punto de destino y el ángulo hacia el cual está el punto de destino. Tratamos esta distancia hasta el destino como un error en distancia y a la diferencia del ángulo actual y el que nos dirige hacia el destino lo tratamos como un error en ángulo.

Disponemos así de dos errores a los cuales podemos aplicar las estrategias clásicas de control PID que se han visto en la asignatura.

La velocidad será calculada para ser proporcional a la distancia hasta la meta, o lo que es lo mismo, proporcional al error en distancia.

Para el control en dirección se usará también otro control proporcional.

#### function [out] = Control\_punto(in)

#### x = in(1);

#### y = in(2);

#### phi = in(3);

#### 

#### x\_ref=3;

#### y\_ref=3;

#### 

#### Kv=0.5;

#### Kp=4;

#### 

#### v = Kv \* sqrt((x\_ref - x)^2+(y\_ref - y)^2);

#### 

#### phi\_ref = atan2( (y\_ref - y),(x\_ref - x) );

#### phi\_d = Kp \* angdiff(phi\_ref, phi);

#### 

#### out = Modelo\_Cinematico\_Inverso([v phi\_d]);

#### 

#### end

Tras ajustar los valores de las ganancias y ejecutar la simulación correspondiente se obtiene el siguiente resultado:



Figura 7. Control punto a punto.



Figura 8. Velocidades de referencia para el control punto a punto.

## Control en línea recta.

El seguimiento de una línea recta () requiere de dos controles para ajustar la dirección. Un controlador dirige el robot de forma que se minimice la distancia normal hacia la recta de acorde a la ecuación:

El primer control proporcional gira el robot hacia esta línea. El segundo control ajusta el ángulo de la cabeza del robot de forma que sea paralela a la línea.

Existirá una ley de control combinada que sea resultado de la suma de los dos controles proporcionales.

#### function [out] = Control\_linea(in)

#### x = in(1);

#### y = in(2);

#### phi = in(3);

#### 

#### % Recta de referencia

#### % ax + by + c = 0

#### a=0.5;

#### b=-1;

#### c=0.5;

#### 

#### % Parámetros del controlador

#### Kd=0.7; % Proporcional a la distancia

#### Kh=1.5; % Proporcional a la orientación

#### 

#### % Distancia a la recta

#### d=(a\*x +b\*y +c)/sqrt(a^2 + b^2);

#### 

#### % Orientación de referencia

#### phi\_ref = atan2(a,-b);

#### 

#### % Señales de control

#### v = 1;

#### phi\_d = Kd\*d + Kh \* angdiff(phi\_ref, phi);

#### 

#### out = Modelo\_Cinematico\_Inverso([v phi\_d]);

#### out=[out d];

#### end

Como prueba del funcionamiento de este algoritmo de control se muestran a continuación los resultados de la simulación.



Figura 9. Trayectoria seguida sobre el plano XY durante el control en línea recta.



Figura 10. Velocidades necesarias para el seguimiento de la recta.

En la Figura 10 podemos comprobar las dinámicas de los actuadores, que se manifiestan en un tiempo no nulo de respuesta, así como en las notables saturaciones tanto de velocidad como de aceleración.

## Control en postura.

## Control mediante algoritmo de persecución pura para el seguimiento de una trayectoria senoidal.

# Conclusiones generales.