Robótica Móvil
Control de Modelos Cinemáticos de Robots

Iñaki Rañó

Notas

En esta práctica vamos a controlar dos modelos cinemáticos de robots móviles, el modelo integrador y el modelo uniciclo. El objetivo es diseñar un controlador que dirija el robot hacia el origen de un sistema de coordenadas suponiendo que el robot tiene conocimiento exacto de su posición respecto a dicho sistema.

1 Introducción y Recordatorio de Teoría de Control

La forma más general de un sistema de control con variables de estado $\mathbf{x}$, entrada $\mathbf{u}$ y salida $\mathbf{y}$ viene dada por dos ecuaciones, la ecuación de transición de estado y la de salida[1]:

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u})$$
$$\mathbf{y} = \mathbf{G}(\mathbf{x}, \mathbf{u})$$

En teoría de control lineal $\mathbf{F}(\cdot, \cdot)$ y $\mathbf{G}(\cdot, \cdot)$ son funciones lineales[2] (i.e. matrices que multiplican a $\mathbf{x}$ y $\mathbf{u}$), pero esto, en general, es la excepción. Si además la entrada y la salida ($\mathbf{u}$ y $\mathbf{y}$) son variables escalares, el sistema se puede caracterizar por su función de transferencia. El objetivo de la ingeniería de control es encontrar entradas $\mathbf{u}$ que consigan que la salida $\mathbf{y}$ se comporte de una forma determinada, por ejemplo que $\mathbf{y}$ tome cierto valor pasado un tiempo. En general todas las varibles, $\mathbf{u}$, $\mathbf{x}$ y $\mathbf{y}$ son en realidad funciones del tiempo, lo que da lugar a dos posibilidades en la búsqueda de $\mathbf{u}$. Se puede buscar 1) una función del tiempo (i.e. secuencia de comandos) $\mathbf{u}(t)$ que consiga que $\mathbf{y}$ se comporte de la forma deseada (control en bucle abierto). Sin embargo, lo que se busca generalmente es 2) encontrar una función $\mathbf{u}(\mathbf{x})$ (o $\mathbf{u}(\mathbf{y})$) que consiga que $\mathbf{y}$ se comporte de la forma deseada (control en bucle cerrado). Si encontramos una función así las ecuaciones se convierten en:

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}(\mathbf{x}))$$
$$\mathbf{y} = \mathbf{G}(\mathbf{x}, \mathbf{u})$$

con lo que la primera ecuación es una ecuación diferencial que solo depende de $\mathbf{x}$. Para sistemas lineales (con controladores lineales) la ecuación de transición de estado es una ecuación diferencial lineal (vectorial) cuya solución son funciones exponenciales (reales o

[1] Estamos suponiendo que el sistema es independiente del tiempo.

[2] Sistemas LTI, lineales invariantes con el tiempo: $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$, $\mathbf{y} = C\mathbf{x} + D\mathbf{u}$.

complejas, i.e. sinusoidales). El controlador en el caso lineal no es más que una matriz de ganancias K que multiplica al vector de estado $\mathbf{x}$, es decir $\mathbf{u} = -K\mathbf{x}$.

En esta práctica vamos a trabajar con dos modelos cinemáticos de robots, el modelo integrador y el modelo unciclo. El término 'cinemático' significa que la ecuación de transición de estado relaciona las velocidades con las posiciones (sin considerar aceleraciones que es lo que realmente hace que se muevan las cosas). Cada modelo representa la forma en que se mueve un robot con distintas configuraciones de ruedas. Por ejemplo, el modelo integrador podría usarse para modelar el movimiento de robots con ruedas suecas, mientras que el uniciclo es el que se usa par robots con direccionamiento dual (dos ruedas controladas por dos motores independientes).

Modelo Integrador: Este es un modelo bidimensional (en el plano) en el que se pueden controlar las velocidades en direcciones perpendiculares (p.e. en el sistema de referencia del suelo). Las ecuaciones de movimiento de un robot así son:

$$\begin{aligned}\dot{x} &= v_x \\ \dot{y} &= v_y\end{aligned}$$

donde por analogía con el sistema de control visto antes tenemos que $\mathbf{x} = [x, y]$, $\mathbf{u} = [v_x, v_y]$, y podemos suponer que la ecuación de salida es $\mathbf{y} = \mathbf{x}$, por lo que $A = 0$, $B = I$, $C = I$ y $D = 0$ (donde I es la matriz identidad 2×2 y 0 es una matriz de ceros 2×2), i.e.

$$\begin{aligned}\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \\ \mathbf{y} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}\end{aligned}$$

Modelo Uniciclo: El modelo uniciclo corresponde con un robot que también se mueve en el plano pero tiene una orientación preferente[3], por lo que el vector de estado es tridimensional $\mathbf{x} = [x, y, \theta]$, sin embargo solo se puede controlar la velocidad de avance (v) y la de giro (ω), por lo que $\mathbf{u} = [v, \omega]$. El robot avanza en la dirección dada por el ángulo θ, pero instantáneamente no puede moverse en cualquier dirección[4]. La ecuaciones de transición de estado para el robot uniciclo son:

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega\end{aligned}$$

y al igual que antes podemos considerar $\mathbf{y} = \mathbf{x}$. Como se puede ver en las ecuaciones, este es un modelo de control no-lineal, i.e. no es posible identificar matrices A, B, C y D para escribir estas ecuaciones como $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$. Es más, una estrategia de control lineal como $\mathbf{u} = -K\mathbf{x}$, no funciona para dirigir al robot al origen ($[x, y, \theta] = [0, 0, 0]$]).

[3]Un robot que sigue el modelo integrador también puede tener orientación, lo que implicaría una tercera ecuación de estado estilo $\dot{\theta} = w$

[4]Solo en la dirección de θ.

2 Simulación a través de integración

A partir de esta sección obviaremos la ecuación de salida, ya que supondremos que queremos controlar el estado $\mathbf{x}$ en vez de la salida $\mathbf{y}$ (o lo que es lo mismo controlamos $\mathbf{y} = \mathbf{x}$). Como hemos visto, una vez diseñado un controlador para nuestro sistema la ecuación de transición de estado se convierte en una ecuación diferencial ordinaria con variables vectoriales). Por ejemplo, en el caso del modelo integrador con un controlador lineal tenemos tenemos $\dot{\mathbf{x}} = -K\mathbf{x}$, o en forma matricial:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = - \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Si la posición inicial del robot es $\mathbf{x}_0$, la trayectoria como función del tiempo, solución del problema de valores iniciales, es $\mathbf{x}(t) = \mathbf{x}_0 e^{-Kt}$ (función exponencial de una matriz), y en el caso en que $k_{12} = k_{21} = 0$ la solución se simplifica de forma que $x(t) = x_0 e^{-k_{11}t}$ y $y(t) = y_0 e^{-k_{22}t}$, que tienden a $[x, y] = [0, 0]$ si $k_{11} > 0$ y $k_{22} > 0$.

En general nos interesa obtener la trayectoria del robot y ver si al pasar el tiempo se aproxima al origen, pero encontrar expresiones analíticas (*closed-form solution*) no suele ser posible (especialmente en sistemas no lineales, e.g. uniciclo). En lugar de ello normalemente (incluso cuando hay solución analítica) se usan métodos de integración numérica (p.e. el método de Euler) por los que se obtiene una secuencia de puntos que represetan la trayectoria del robot (las variables de estado en distintos instantes de tiempo). Como integrar ecuaciones diferenciales a partir de condiciones iniciales es un problema que aparece a menudo en ciencias e ingeniería, existen librerías especiales para resolver este tipo de problemas. En `python` por ejemplo la libreria `scipy` contiene un paquete para integrar ecuaciones diferenciales. Concretamente la funcion `scipy.integrate.odeint()` puede resolver problemas de valores iniciales, y su API es:

```
scipy.integrate.odeint(func, x0, T,...)
```

donde:

- `func` es una función o *callable* con el siguiente formato `dx/dt=func(x,t)`, donde `x` es el vector de estado, `t` es el instante de integración (relevante para sistemas dependientes del tiempo) y `dx/dt` es la derivada temporal de `x`, i.e. $\dot{\mathbf{x}}$.
- `x0` es el valor inicial del estado, i.e. $\mathbf{x}_0$
- `T` es un vector de instantes temporales en los que se quiere obtener el estado, es decir, si `T`$= [t_0, t_1, \ldots, t_n]$, la función `odeint()` generará los puntos $\mathbf{x}_{t_0}$, $\mathbf{x}_{t_1}$,... $\mathbf{x}_{t_n}$ y los retornará como una matriz de tamaño $n \times d$, donde d es la dimensión de $\mathbf{x}$.

Esta es la función que se usará para simular los modelos cinemáticos de la práctica, como se puede ver en el fichero `simulator.py` adjunto. Para poder ejecuar este *script* de `python` es necesario completar las líneas de código al principio del fichero, donde se define la posición inicial del robot `p0` y un objeto de tipo `IntegratorRobot` o `UnicycleRobot` (medio-definidos en el fichero `robot.py`). La función `odeint()` es totalmente agnóstica del problema a resolver, y en este caso, con los parámetros adecuados puede simular ambos modelos.

3 Control del Modelo Integrador

Dado que el modelo integrador corresponde a un sistema lineal se utilizará un controlador lineal en variables de estado, es decir una matriz K 2×2 de ganancias con entradas k_{11}, k_{12},

k_{21} y k_{22}. El fichero `robot.py` contiene la definición de la clase `IntegratorRobot` que incluye un método `controler()` que es necesario implementar para controlar el robot integrador de forma que se dirija hacia el origen.

Comienza definiendo una matriz de ganancias diagonal con componentes idénticas en la diagonal y simula el sistema en bucle cerrado. Para visualizar los resultados se define la clase `GraphicsRobot` con tres métodos diferentes de visualización que terminan al pulsar una tecla sobre la ventana:

1. `PlotTrajectory()` representa la trayectoria del robot (incluyendo posiciones inicial y final) en el espacio Cartesiano.
2. `PlotVariables()` muestra en diferentes gráficas la evolución temporal de cada una de las variables de estado
3. `AnimateTrajectory()` muestra una animación de la trajectoria resultante del robot.

Realiza distintas simulaciones con posiciones iniciales del robot distintas para cierta matriz K **Q1. ¿Tienen las trajectorias generadas para distintas posiciones iniciales características comunes? ¿Cuáles?**. Cambia la matriz de ganancias para que las constates k_{11} y k_{22} tengan valores diferentes **Q2. ¿Se observa algún cambio en las trayectorias? ¿Cuáles y por qué? Q3. ¿Es posible hacer que el sistema sea inestable, i.e. que el robot no se acerque al origen? ¿Cómo? Q4. ¿Es posible hacer que el robot se mueva solamente hacia el eje x o el y? ¿Cómo?**

Cambia otra vez tu matriz de ganancias para que no sea diagonal y simula el control del robot modelo integrador con esa ganancia. **Q5. ¿Hay ganancias para los que el comportamiento es estable o inestable? Pon ejemplos. Q6. ¿Qué aspecto tienen las trayectorias para distintas matrices de ganancia? ¿Puedes relacionar las trayectorias con los valores y vectores propios de la matriz K?**[5].

4 Control del Modelo Uniciclo

El modelo uniciclo no puede controlarse a través de un controlador lineal en las variables de estado $\mathbf{x} = [x, y, \theta]$, pero puede demostrarse que a través de un cambio de variables el sistema en bucle cerrado es estable si se usa el siguiente controlador:

$$v = k_\rho \rho$$
$$\omega = k_\beta \beta + k_\alpha \alpha$$

donde k_ρ, k_β y k_α son ganancias (constantes) y las nuevas variables son ρ, β y α, que se definen de la forma siguiente:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$
$$\alpha = \arctan \frac{\Delta y}{\Delta x} - \theta$$
$$\beta = -\theta - \alpha$$

donde Δx y Δy son los incrementos en la posición Cartesiana con respecto a la referencia, es decir $\Delta x = x_r - x$ y $\Delta y = y_r - y$ (en este caso la posición de referencia es $[x_r, y_r] = [0, 0]$). Respecto al cálculo de $\arctan(x)$, dado que la imágen, i.e. los valores que

[5] Puedes calcularlos con la función `numpy.linalg.eig()` de `numpy`

retorna, está en el rango $(-\frac{\pi}{2}, \frac{\pi}{2})$, no es posible obtener ángulos mayores que ese rango[6], por lo que en lugar de usar la expresión $\arctan \frac{\Delta y}{\Delta x}$ utilizaremos la función $arctan2(y, x)$[7] que convierte el ángulo al rango $(-\pi, \pi]$ teniendo en cuenta los signos de x e y. Finalmente, las operaciones con ángulos, p.e. al calcular α o β darán resultados fuera del rango, por lo que se proporciona un método para volver a poner el ángulo en el rango apropiado. Este método está implementado en la clase `UnicycleRobot` del fichero `robot.py`, y al igual que en el caso del robot integrador hay un método `controler()` que es necesario implementar para controlar el robot uniciclo de forma que se dirija hacia el origen. Como se puede ver en el fichero, el modelo uniciclo se implementa en el método `__call__()` de la clase.

Implementa el controlador propuesto, ajusta valores para las ganancias y simula el movimiento del robot para distintos valores iniciales de la pose, i.e. $\mathbf{x}_0 = [x_0, y_0, \theta_0]$. **Q7. Comprueba que el controlador será estable para valores de ganancias que cumplan $k_\rho > 0$, $k_\alpha > k_\rho$ y $k_\beta < 0$. Ilustra tu respuesta con simulaciones. Q8. ¿Que pasa en los límites de estabilidad? Muestra alguna simulación en estos casos Q9. ¿Alcanza el robot la posición objetivo, i.e. $x = 0$, $y = 0$ y $\theta = 0$? ¿Por qué?**

[6] Por ejemplo si $\Delta x = -1$ y $\Delta y = -1$, que corresponde al ángulo $-\frac{3\pi}{4}$, el cálculo de $\arctan \frac{-1}{-1} = \arctan(1) = \frac{\pi}{4}$, que no es el resultado que deseamos ya que queremos que los ángulos estén en el rango $(-\pi, \pi]$.

[7] `numpy.arctan2(y,x)`