Modelado cinemático, desacoplamiento y linealización de un robot no holonómico, movimiento plano (control por punto descentrado)

Presentaciones en vídeo:

http://personales.upv.es/asala/YT/V/robdmod.html , http://personales.upv.es/asala/YT/V/robdfl.html , http://personales.upv.es/asala/YT/V/robdzd.html .

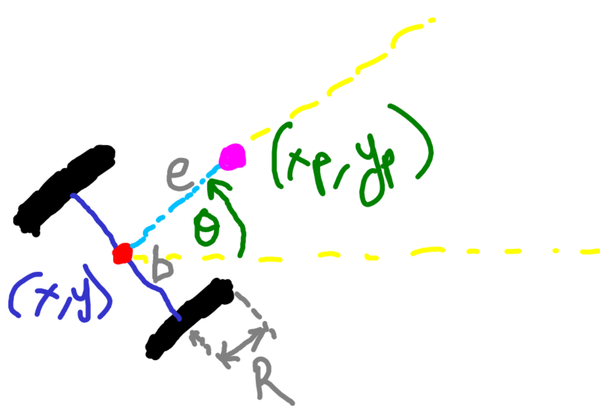
Este código ejecutó sin errores en Matlab `R2020b`

Objetivo: modelar un robot que se mueve girando dos ruedas, y se orienta según la diferencia de ángulo girado entre ambas. Diseñar un control para conseguir una velocidad deseada de un cierto punto que no esté en la linea del eje de las ruedas.

Tabla de Contenidos

Descripción y datos geométricos del robot

```
syms w1 w2 theta x y real
b=0.5; %separación del centro del eje a ruedas.
e=1; %separación del punto de control a centro eje.
R=0.25; %radio rueda
```



*En la parte final de análisis de dinámica del ángulo, llamaremos "*adelante*" la dirección desde el punto rojo (centro de eje de ruedas) hacia el punto descentrado (rosa).

Modelado cinemático

Las ecuaciones de velocidades angulares de ruedas a velocidad lineal y angular del centro del eje entre ruedas son:

```
v=(w1*R+w2*R)/2; %velocidad lineal del centro eje ruedas
w=(w2*R-w1*R)/(2*b); %velocidad angular del cuerpo del robot
```

Las ecuaciones de movimiento del robot son:

```
dxdt=v*cos(theta);
dydt=v*sin(theta);
dthetadt=w;
```

la representación interna normalizada tiene las **ecuaciones de estado**:

```
estado=[x y theta]'; entradas=[w1;w2];
destadodt=[dxdt;dydt;dthetadt]
```

```
destadodt =
```

$$\begin{pmatrix} \cos(\theta)\left(\frac{w_1}{8}+\frac{w_2}{8}\right) \\ \sin(\theta)\left(\frac{w_1}{8}+\frac{w_2}{8}\right) \\ \frac{w_2}{4}-\frac{w_1}{4} \end{pmatrix}$$

No holonómico: si $\theta = 0$, la linealización sale que "y" es un estado NO controlable... si $\theta = \pi/2$, la linealización sale que "x" es un estado NO controlable; un estado no controlable no nos lo vamos a poder quitar de encima (bueno, existen maniobras basadas en el "*corchete de Lie* [Lie bracket]" que formalizan una "maniobra de *aparcamiento*", no objeto de este material), pero podemos hacer que el no controlable sea θ usando el truco del "punto descentrado".

Las **ecuaciones de salida** serán la posición 2D del punto descentrado:

```
xp=x+e*cos(theta);
yp=y+e*sin(theta);
EcSalida=[xp;yp]
```

```
EcSalida =
```

$$\begin{pmatrix} x+\cos(\theta) \\ y+\sin(\theta) \end{pmatrix}$$

```
EcEstadoRobot_num=matlabFunction(destadodt,'Vars',{estado,entradas});
EcSalida_num=matlabFunction(EcSalida,'Vars',{estado}); %para simulación.
```

Desacoplamiento y Linealización por realimentación (no lineal) del estado

En abstracto, con la notación "y=salida", "x=estado", las derivadas temporales de la salida se obtienen con regla de la cadena:

$$\frac{dy(x)}{dt} = \frac{\partial y(x)}{\partial x} \cdot \frac{dx}{dt}$$

Si x es un vector $x = (x_1, \ldots, x_n)$ entonces:

$$\frac{dy(x)}{dt} = \frac{\partial y(x)}{\partial x_1} \cdot \frac{dx_1}{dt} + \ldots + \frac{\partial y(x)}{\partial x_n} \cdot \frac{dx_n}{dt} = \begin{pmatrix} \frac{\partial y(x)}{\partial x_1} & \frac{\partial y(x)}{\partial x_2} & \cdots & \frac{\partial y(x)}{\partial x_n} \end{pmatrix} \cdot \begin{pmatrix} \frac{dx_1}{dt} \\ \vdots \\ \frac{dx_n}{dt} \end{pmatrix}$$

Las derivadas primeras de las salidas respecto al tiempo, pues, son:

```
dsalidadt=jacobian(EcSalida,estado)*destadodt
```

dsalidadt =

$$\begin{pmatrix} \cos(\theta)\left(\frac{w_1}{8} + \frac{w_2}{8}\right) + \sin(\theta)\left(\frac{w_1}{4} - \frac{w_2}{4}\right) \\ \sin(\theta)\left(\frac{w_1}{8} + \frac{w_2}{8}\right) - \cos(\theta)\left(\frac{w_1}{4} - \frac{w_2}{4}\right) \end{pmatrix}$$

Podemso igualarlas a "lo que nosotros queramos" e intentar resolver para despejar w_1 y w_2:

```
syms v_x v_y real
sol=solve(dsalidadt==[v_x;v_y],{w1,w2});
```

Como Matlab no ha dado ningún error ni warning, el "control cinemático" es, por tanto, posible y podemos calcular las velocidades de rueda que dan lugar a las velocidades lineales del punto de control (v_x, v_y) con:

```
simplify(sol.w1)
```

ans = $4\,v_x \cos(\theta) - 2\,v_y \cos(\theta) + 2\,v_x \sin(\theta) + 4\,v_y \sin(\theta)$

```
simplify(sol.w2)
```

ans = $4\,v_x \cos(\theta) + 2\,v_y \cos(\theta) - 2\,v_x \sin(\theta) + 4\,v_y \sin(\theta)$

Por lo tanto con la ley de control $(w1, w2) = sol(\theta, v_x, v_y)$ calculada simbólicamente en la variable "`sol`", convertiríamos el robot a un par de integradores:

$$\frac{dx_p}{dt} = v_x, \qquad \frac{dy_p}{dt} = v_y$$

Con lo cual, tenemos un modelo **linealizado** y **desacoplado**. En transformada de Laplace sería:

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} \frac{1}{s} & 0 \\ 0 & \frac{1}{s} \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

El código numérico que ejecutaría ese desacoplamiento sería:

```
global Desacoplador
Desacoplador=matlabFunction([sol.w1;sol.w2])
```

```
Desacoplador = function_handle with value:
    @(theta,v_x,v_y)[((v_x.*cos(theta).*2.0-v_y.*cos(theta)+v_x.*sin(theta)+v_y.*sin(theta).*2.0).*2.0)./(
```

Devuelve velocidades angulares de rueda dado ángulo y velocidades deseadas v_x, v_y del punto de control.

NOTA: Aunque parece que teniendo en cuenta sólo la salida (punto descentrado), hemos transformado a segundo orden, la física sigue siendo de tercer orden: El modelo de orden 3 con desacoplador cambia las entradas (vel. rueda) a velocidad pto. de control, será

```
destadodt_bc=simplify(subs(destadodt,{w1,w2},{sol.w1,sol.w2}),50)
```

destadodt_bc =

$$\begin{pmatrix} v_x \cos(\theta)^2 + v_y \sin(\theta)\cos(\theta) \\ v_y \sin(\theta)^2 + \frac{v_x \sin(2\,\theta)}{2} \\ v_y \cos(\theta) - v_x \sin(\theta) \end{pmatrix}$$

Dinámica no observable

El sistema original es de orden 3, el sistema transformado es de orden 2 ($1/s$ diagonal). Por tanto, un estado del sistema se ha convertido en *"no observable"* desde las salidas (x_p, y_p) tras el desacoplamiento.

Refinemos el concepto:

-- sigue siendo "físicamente observable" porque lo estamos **midiendo** directamente: es necesario para implementar el desacoplador/linealizador ϕ, en estrategias de "realimentación del estado" no importa observabilidad para nada.

-- No es observable para el controlador "maestro" que controle las posiciones: sólo mirando la posición del punto de control no se puede calcular el ángulo (no holonómico: dependiendo de la trayectoria entre dos posiciones, el ángulo final puede diferir). Las desviaciones de una hipotética "trayectoria angular deseada" no se tienen en cuenta para nada en el cálculo de v_x o de v_y, si sólo se consideran los dos integradores como modelo (por supuesto, no si se considera el tener que hacer un "aparcamiento" donde el ángulo también tenga una referencia).

El modelo de orden 3 con desacoplador, tomando como estados (x_p, y_p, θ) --como dados (x_p, y_p, θ) se puede calcular (x, y, θ) y viceversa, se puede hacer el cambio de variable-- será:

```
destadonewdt_bc=simplify(subs([v_x;v_y;destadodt(3)],{w1,w2},{sol.w1,sol.w2}),50)
```

```
destadonewdt_bc =
```

$$\begin{pmatrix} v_x \\ v_y \\ v_y \cos(\theta) - v_x \sin(\theta) \end{pmatrix}$$

La dinámica no observable será la tercera de las ecuaciones:

$$\frac{d\theta}{dt} = v_y \cos(\theta) - v_x \sin(\theta)$$

Y ello dará lugar a una dinámica que, claro, dependerá de v_x y v_y qué puntos de equilibrio o trayectorias se siguien, y si determinado "equilibrio" es estable o no...

Trayectorias rectas: Suponiendo que se sigue una trayectoria recta $\dot{y}_p = \gamma \cdot \dot{x}_p$, esto es $v_y = \gamma v_x$, --*nota*: en rigor, el caso "vertical" no podría analizarse así, daría $\gamma = \infty$, sería necesario $v_x = \gamma' \cdot v_y$-- entonces el equilibrio será con derivadas a cero:

$$0 = v_x(\gamma \cos(\theta_{eq}) - \sin(\theta_{eq})) = v_x \cos(\theta_{eq}) \cdot \left(\gamma - \frac{\sin(\theta_{eq})}{\cos(\theta_{eq})}\right),$$

o sea $\tan(\theta_{eq}) = \gamma$... en el que hay dos ángulos de equilibrio, con 180 grados de diferencia, claro, "marcha hacia adelante" o "hacia atrás".

Su linealización será $\frac{d\Delta\theta}{dt} = -v_x \cdot (\gamma \sin(\theta_{eq}) + \cos(\theta_{eq})) \cdot \Delta\theta$. Haciendo operaciones:

$$\frac{d\Delta\theta}{dt} = -v_x(\gamma^2 + 1)\cos(\theta_{eq}) \cdot \Delta\theta$$

de modo que uno de los dos equilibrios será estable y otro no, según el signo del coseno y de la velocidad v_x, para que se verifique que el polo sea negativo, esto es, $-v_x(\gamma^2 + 1)\cos(\theta_{eq}) < 0$:

En concreto,

- si $v_x > 0$ (moviéndose hacia la derecha), entonces será estable con $\cos(\theta_{eq}) > 0$, esto es, robot "apuntando a la derecha".

- si $v_x < 0$ (moviéndose hacia la izquierta), entonces será estable con $\cos(\theta_{eq}) < 0$, esto es, robot "apuntando a la izquierda".

Como conclusión, la presente ley de control no podría usarse para guiar al robot "haciendo marcha atrás".