Universidad Rey Juan Carlos

E.T.S. Ingeniería de Telecomunicación

Ingeniería de Control

Práctica 2

Autor:
Javier Izquierdo Hernández

November 16, 2023

Práctica 1

Consider the simplified planar model of the control system for self balancing of a motorcycle represented in Figure 1, in which the motorcycle is represented by a bar that rotates around point $\mathbf{b}$. Points $\mathbf{c}_1$ and $\mathbf{c}_2$ are the positions of the centers of mass of the motorcycle and the reaction wheel, respectively. The control system is composed of an actuated reaction wheel, which rotates around point $\mathbf{c}_2$. A motor controls the rotation of the reaction wheel applying a torque τ, which is the only input of the control system. Angles q_1, q_2 and their time derivatives are the state variables of the system.

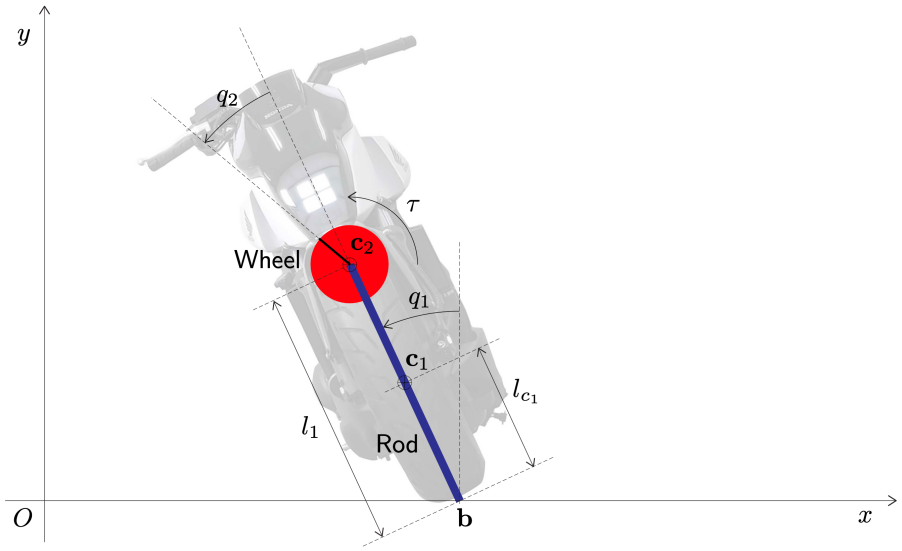

Figure 1: Sketch of the control system for self balancing of a motorcycle

The dynamic model of this system is

$$\begin{aligned}(m_1 l_{c_1}^2 + m_2 l_1^2 + I_1 + I_2)\ddot{q}_1 + I_2\ddot{q}_2 - (m_1 l_{c_1} + m_2 l_1) g \sin(q_1) &= 0,\\ I_2\ddot{q}_1 + I_2\ddot{q}_2 &= \tau,\end{aligned}$$

with the following parameters

- mass of the motorcycle $m_1 = 200$ [kg],
- mass of the reaction wheel $m_2 = 50$ [kg],
- barycentric moment of inertia of the motorcycle $I_1 = 25$ [kg m^2],
- barycentric moment of inertia of the reaction wheel $I_2 = 5$ [kg m^2],
- length of the rod $l_1 = 1$ [m],

- $l_{c_1} = \frac{l_1}{2} = 0.5$ [m],
- gravity acceleration: $g = 9.81$ [m/s^2].

1) Intentionally omitted.

2) Calculate the state space representation of the system, assuming that $\mathbf{x} = (x_1, x_2, x_3, x_4)^T = (q_1, q_2, \dot{q}_1, \dot{q}_2)^T$, where angles are expressed in [rad] and angular velocities in [rad/s]. (Contesta en el informe y sube el código `Matlab` a Aula Virtual en el fichero `statespace.m`)

3) Calculate all the operating points of the system and explain the obtained result. (Contesta en el informe y sube el código `Matlab` a Aula Virtual en el fichero `equilibrium.m`)

4) Find the operating point that corresponds to $\overline{x}_1 = 0, \overline{u} = 0$. Linearize the system around this operating point. (Contesta en el informe y sube el código `Matlab` a Aula Virtual en el fichero `linearization.m`)

5) Is the linearized system controllable? (Contesta en el informe y sube el código `Matlab` a Aula Virtual en el fichero `controllability.m`)

6) Is the linearized system observable by means of the output $y = x_1$? Is the linearized system controllable by means of the output $y = x_2$? (Contesta en el informe y sube el código `Matlab` a Aula Virtual en el fichero `observability.m`)

7) Using the pole placement method and an observer, design an output feedback controller to control the inclination of the motorcycle. We want to steer the motorcycle from the state $\mathbf{x} = (q_1, q_2, \dot{q}_1, \dot{q}_2)^T = (0.26179, 0, 0, 0)^T$ to the state $\mathbf{x} = (0, 0, 0, 0)^T$. Give the eigenvalues that have been assigned to the controlled system and illustrate the behaviour of the controller by plotting the relevant state and control variables and by a graphical animation. (Contesta en el informe y sube el código `Matlab` a Aula Virtual en la carpeta `controller`)

Originality and completeness of the written answers will be the aspects that will be taken into account in the grading of the exam, and therefore, the `Matlab` code alone will not be considered.

Solution

2) Calculate the state space representation of the system, assuming that $\mathbf{x} = (x_1, x_2, x_3, x_4)^T = (q_1, q_2, \dot{q}_1, \dot{q}_2)^T$, where angles are expressed in [rad] and angular velocities in [rad/s]. (Contesta en el informe y sube el código `Matlab` a Aula Virtual en el fichero `statespace.m`)

Para calcular las ecuaciones del espacio de estados se utiliza la función *solve()* de MATLAB. Para luego poder decidir si sustituir o no, las variables g, m1, m2, l1, etc.se utilizan de forma simbólica. Por último lo único a tener en cuenta es que *u* es igual a τ.

File `statespace.m`

```
close all
clc
clear

syms q1 q1dot q1ddot q2 q2dot q2ddot tao m1 m2 I1 I2 l1 lc1 g u;

% Known variables
% m1 = 200;
% m2 = 50;
% I1 = 25;
% I2 = 5;
% l1 = 1;
% lc1 = 0.5;
% g = 9.81;
tao = u

% State space representation x = [q1, q2, q1dot, q2dot];

% Define the dynamic equations
ecuaciones = [((m1*lc1^2 + m2*l1^2 + I1 + I2)*q1ddot + I2*q2ddot -(m1*lc1 +m2*l1)*g*sin(q1)==0),
(I2*q1ddot + I2*q2ddot - tao ==0)];
% Solve q1ddot and q2ddot
[q1ddot, q2ddot] = solve (ecuaciones,[q1ddot, q2ddot]);

% State representation
xdot = [q1dot; q2dot; q1ddot; q2ddot]
```

3) Calculate all the operating points of the system and explain the obtained result. (Contesta en el informe y sube el código `Matlab` a Aula Virtual en el fichero `equilibrium.m`)

Para calcular los puntos operativos del sistema, se tiene que usar las ecuaciones del espacio de estados calculadas anteriormente. Para utilizarlas las definimos en *f1, f2, f3 y f4* , que luego añadimos a la matriz *eqs*, igualando todas las ecuaciones del espacio de estados a 0 para así encontrar los puntos operativos.
Se vuelve a usar otra vez la función *solve()* de MATLAB, para las variables del espacio de estados (*x1, x2, x3, x4 y u*) y con las todas las variables símbolicas

usadas anteriormente sustituidas por sus valores reales, para así tener unos puntos operativos más exactos.
El resultado resultante, indica que el estado estará en equilibrio cuando ambas velocidades sean 0, es decir, $q3$ y $q4$ como son $\dot{q1} = q3$ y $\dot{q2} = q2$ respectivamente, derivadas del espacio. En cuanto a u, esta será 0. Luego $q1$, posición angular del péndulo, será $k\pi$, siendo $k = 1, 2, 3, ...$, es decir, el estado de equilibrio ocurrirá cuando el péndulo se encuentre totalmente hacia arriba o hacia abajo, sin ninguna velocidad, y además como $q2 = x$ esto indica que da igual la posición en la que se encuentre la rueda respecto a su eje.

File `equilibrium.m`

```
close all
clc
clear

syms x1 x2 x3 x4 tao m1 m2 I1 I2 l1 lc1 g u;

% Known variables
m1 = 200;
m2 = 50;
I1 = 25;
I2 = 5;
l1 = 1;
lc1 = 0.5;
g = 9.81;
tao = u;

f1 = x3;
f2 = x4;
f3 = (g*l1*m2*sin(x1) - tao + g*lc1*m1*sin(x1))/(m2*l1^2 + m1*lc1^2 + I1);
f4 = (m2*tao*l1^2 - I2*g*m2*sin(x1)*l1 + m1*tao*lc1^2 - I2*g*m1*sin(x1)*lc1 + I1*tao + I2*tao)/(I2*(m2*l1^2 + m1*lc1^2

% Estado de equilibrio
eqs = [f1 == 0; f2 == 0; f3 == 0; f4 == 0 ];
S = solve(eqs, [x1, x2, x3, x4, u], 'ReturnConditions', true, 'Real', true);

q1 = S.x1
q2 = S.x2
q3 = S.x3
q4 = S.x4
barrau = S.u
```

4) Find the operating point that corresponds to $\overline{x}_1 = 0, \overline{u} = 0$. Linearize the system around this operating point. (Contesta en el informe y sube el código `Matlab` a Aula Virtual en el fichero `linearization.m`)

Con el resultado del apartado anterior se sabe que hay un punto operativo en $\overline{x}_1 = 0, \overline{u} = 0$, luego usando las ecuaciones del espacio de estados podemos calcular las matrices A, B, C y D, y en este caso podemos que suponer que $y = x2$ aunque esto no vaya a importar en los siguientes pasos.
Para conseguir las matrices anteriores se usa la función *jacobian()* de MATLAB, y luego se usa otra función de MATLAB, *subs()*, para substituir los

valores que ya sabemos que son 0, como $x1, x3, x4, u$.
Hay que recordar que volvemos a dejar todas las constantes de forma simbólica para poder sustituirlas más adelante.

File `linearization.m`

```
close all
clc
clear

syms x1 x2 x3 x4 tao m1 m2 I1 I2 l1 lc1 g u;

% Known variables
% m1 = 200;
% m2 = 50;
% I1 = 25;
% I2 = 5;
% l1 = 1;
% lc1 = 0.5;
% g = 9.81;
tao = u;

f1 = x3;
f2 = x4;
f3 = (g*l1*m2*sin(x1) - tao + g*lc1*m1*sin(x1))/(m2*l1^2 + m1*lc1^2 + I1);
f4 = (m2*tao*l1^2 - I2*g*m2*sin(x1)*l1 + m1*tao*lc1^2 - I2*g*m1*sin(x1)*lc1 + I1*tao + I2*tao)/(I2*(m2*l1^2 + m1*lc1^2

% Estado de equilibrio
eqs = [f1; f2; f3; f4];
y = x2

A = jacobian(eqs,[x1, x2, x3, x4])
B = jacobian(eqs, u)
C = jacobian(y, [x1, x2, x3, x4])
D = jacobian(y, u)

Aeq = subs(A, [x1,x3,x4,u], [0,0,0,0])
eig(Aeq)
Beq = subs(B, [x1,x3,x4,u], [0,0,0,0])
Ceq = subs(C, [x1,x3,x4,u], [0,0,0,0])
```

5) Is the linearized system controllable? (Contesta en el informe y sube el código Matlab a Aula Virtual en el fichero `controllability.m`)

Usando las matrices Aeq y Beq, que son A y B después de realizar la sustitución, calculamos la matriz de controlabilidad usando la función *crtb()* de MATLAB y observando su rango.
Como el rango resultante es 4, que coincide con el rango máximo de las ecuaciones de estado, se puede decir que el sistema es controlable en $\overline{x}_1 = 0, \overline{u} = 0$.

File controllability.m

```
close all
clc
clear

syms x1 x2 x3 x4 tao m1 m2 I1 I2 l1 lc1 g u;

% Known variables
m1 = 200;
m2 = 50;
I1 = 25;
I2 = 5;
l1 = 1;
lc1 = 0.5;
g = 9.81;
tao = u;

f1 = x3;
f2 = x4;
f3 = (g*l1*m2*sin(x1) - tao + g*lc1*m1*sin(x1))/(m2*l1^2 + m1*lc1^2 + I1);
f4 = (m2*tao*l1^2 - I2*g*m2*sin(x1)*l1 + m1*tao*lc1^2 - I2*g*m1*sin(x1)*lc1 + I1*tao + I2*tao)/(I2*(m2*l1^2 + m1*lc1^2

% Estado de equilibrio
eqs = [f1; f2; f3; f4];

A = jacobian(eqs,[x1, x2, x3, x4])
B = jacobian(eqs, u);

Aeq = subs(A, [x1,x3,x4,u], [0,0,0,0]);
Beq = subs(B, [x1,x3,x4,u], [0,0,0,0]);

eig(Aeq)

Co = ctrb(Aeq, Beq)
rangoContrl = rank(Co) % Como el rango es maximo en 4 entonces es controlable
```

6) Is the linearized system observable by means of the output $y = x_1$? Is the linearized system observable by means of the output $y = x_2$? (Contesta en el informe y sube el código Matlab a Aula Virtual en el fichero observability.m)

Aquí se repiten los mismos pasos que en apartado 4, ya que se necesita linearizar el sistema para $y = x1$ e $y = x2$. A los pasos seguidos en ese apartado hay que añadir el propio de la observabilidad, que es el cálculo de la matriz del mismo nombre y el estudio de su rango.
Si el rango de la matriz de observabilidad es igual al del rango máximo de las ecuaciones de estado, es decir, 4, entonces es observable.
Esto solo se cumple cuando $y = x2$, ya que cuando $y = x1$ el rango de la matriz resultante es 2, que es menor que el rango máximo.

File observability.m

```
close all
clc
clear

syms x1 x2 x3 x4 tao m1 m2 I1 I2 l1 lc1 g u;

% Known variables
m1 = 200;
m2 = 50;
I1 = 25;
I2 = 5;
l1 = 1;
lc1 = 0.5;
g = 9.81;
tao = u;

f1 = x3;
f2 = x4;
f3 = (g*l1*m2*sin(x1) - tao + g*lc1*m1*sin(x1))/(m2*l1^2 + m1*lc1^2 + I1);
f4 = (m2*tao*l1^2 - I2*g*m2*sin(x1)*l1 + m1*tao*lc1^2 - I2*g*m1*sin(x1)*lc1 + I1*tao + I2*tao)/(I2*(m2*l1^2 + m1*lc1^2

% Estado de equilibrio
eqs = [f1; f2; f3; f4];
y = x1;

A = jacobian(eqs,[x1, x2, x3, x4]);
B = jacobian(eqs, u);
C = jacobian(y, [x1, x2, x3, x4]);
D = jacobian(y, u);

Aeq = subs(A, [x1,x3,x4,u], [0,0,0,0]);
Beq = subs(B, [x1,x3,x4,u], [0,0,0,0]);
Ceq = subs(C, [x1,x3,x4,u], [0,0,0,0]);

Ob = obsv(Aeq, Ceq)
rangoObsv = rank(Ob) % Como el rango es 2 y el maximo es 4 entonces no es observable

y = x2;

A = jacobian(eqs,[x1, x2, x3, x4]);
B = jacobian(eqs, u);
C = jacobian(y, [x1, x2, x3, x4]);
D = jacobian(y, u);

Aeq = subs(A, [x1,x3,x4,u], [0,0,0,0]);
Beq = subs(B, [x1,x3,x4,u], [0,0,0,0]);
Ceq = subs(C, [x1,x3,x4,u], [0,0,0,0]);

Ob = obsv(Aeq, Ceq)
rangoObsv = rank(Ob) % Como el rango es maximo en 4 entonces es observable
```

7) Using the pole placement method and an observer, design an output feedback controller to control the inclination of the motorcycle. We want to steer the motorcycle from the state $\mathbf{x} = (q_1, q_2, \dot{q}_1, \dot{q}_2)^T = (0.26179, 0, 0, 0)^T$ to the state $\mathbf{x} = (0, 0, 0, 0)^T$. Give the eigenvalues that have been assigned to the controlled system and illustrate the behaviour of the controller by plotting the relevant state and control variables and by a graphical animation. (Contesta en el informe y sube el código `Matlab` a Aula Virtual en la carpeta `controller`)

Se usa la estructura de código que se ha usado en clase para sistemas con observador, como por ejemplo el ejercicio 6.6, y se modifican las partes correspondientes a este sistema, como son la función $xdot()$ del fichero prac2_f.m (en el que se usan las ecuaciones del espacio de estados obtenidas en el apartado 2), la función $xhatdot()$ del fichero prac2_obs_f.m (en el que usando las matrices obtenidas al linearizar en el apartado 4 para poder calcular $\dot{\hat{x}}$) como se explicará más adelante) y el *main* del fichero prac2_main.m. También se ha modificado la función *draw* para poder realizar una representación adecuada del sistema.
Para todo esto se ha necesitado del cálculo de 3 matrices K, H y L usadas para obtener u y $\hat{x}$. La matriz K y L se obtienen usando la función *place()* de MATLAB con las matrices calculadas en el apartado 4, es decir, A, B y C, y además con los polos del sistemas que hemos puesto de forma arbitraria en -2 tanto para el controlador como para el observador, ya que este valor da un control rápido, aunque se puede usar un valor menor para un control menos brusco, o uno mayor para el resultado contrario. (Recordar que para poner polos múltiples en MATLAB hay que ponerlos un poco separados unos de otros). Luego para extraer la matriz H se usan las matrices A, B, K y una nueva matriz fila $E = [0, 1, 0, 0]$.
Con todo esto ya se obtiene la representación gráfica del control, pero cabe recordar que el estado inicial es el indicado en la pregunta $x = (0.26179, 0, 0, 0)^T$ tanto para x como para $\hat{x}$ del observador.