Desacoplamiento por realimentación del estado, dinámica no observable (dinámica cero) en sistemas lineales

Presentación en vídeo: http://personales.upv.es/asala/YT/V/fbdclin.html

Este código funcionó correctamente con Matlab **R2020b**

Objetivo: El objetivo de este material es poner un ejemplo de desacoplamiento por realimentación en un sistema lineal donde el "grado relativo" de cada salida será diferente y requerirá, por tanto, un orden de derivación diferente. Se discutirá la dinámica observable (cancelación de ceros).

Tabla de Contenidos

Modelo y análisis de propiedades

```
s=tf('s');
```

Caso 1: sistema de fase mínima; desacoplamiento por realimentación funciona.

```
%G=[1/(s^2+s+4) 1/(s+1)^2;
%    -2/(s+1) 5/(s+1)];
```

Caso 2: este es un caso de fase no mínima, que no funcionará (desacopl. por realimentación será internamente inestable).

```
G=[-(s-1)/(s^2+s+4)/(s+1) 0;
    -2/(s+1) 5/(s+1)];
%Tiene el mismo grado relativo, para no modificar el código... en un caso
%general el número de derivadas de cada salida cambia según el proceso...
```

Grado relativo (polos menos ceros)

- es DOS en la primera salida (fila de G), y UNO en la segunda.

Representación interna:

```
sys=minreal(ss(G)); %pasemos a estado
sys.InputName={'u1','u2'};
```

```
sys.OutputName={'y1','y2'};
size(sys)
```

```
State-space model with 2 outputs, 2 inputs, and 4 states.
```

```
pole(sys)
```

```
ans = 4×1 complex
  -0.5000 + 1.9365i
  -0.5000 - 1.9365i
  -1.0000 + 0.0000i
  -1.0000 + 0.0000i
```

```
tzero(sys) %CASO 1: hay dos ceros, de fase mínima (parte real <0)
```

```
ans = 1.0000
```

```
%CASO 2: un cero de fase no mínima (parte real>=0)
```

Resulta conveniente pasarlo a un objeto de la Symbolic Toolbox para poder hacer jacobianos y manipulaciones diversas:

```
Estado=sym('x',[size(sys.A,2) 1],'real');
syms u1 u2 real
Entrada=[u1;u2];
dxdt=sys.A*Estado+sys.B*Entrada %simbólica
```

```
dxdt =
```

$$\begin{pmatrix} u_1 - 2\,x_1 - \frac{5\,x_2}{2} - 2\,x_3 \\ 2\,x_1 \\ x_2 \\ \frac{5\,u_2}{2} - u_1 - x_4 \end{pmatrix}$$

```
EcSalida=sys.C*Estado+sys.D*Entrada %simbólica
```

```
EcSalida =
```

$$\begin{pmatrix} \frac{x_3}{2} - \frac{x_2}{2} \\ 2\,x_4 \end{pmatrix}$$

Derivadas de las salidas hasta que aparezcan las acciones de control

Si $y = Cx$, entonces $\dot{y} = C\dot{x} = CAx + CBu$; obviamente C es $\frac{\partial y}{\partial x}$, si usamos "jacobian" el código vale para "no lineal", si tenemos $y(x)$, entonces $\frac{dy}{dt} = \frac{\partial y}{\partial x} \cdot \frac{dx}{dt}$ (regla de la cadena).

```
jacobian(EcSalida,Estado) %esto es sys.C, en simbólico
```

```
ans =
```

$$\begin{pmatrix} 0 & -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

```
dSalidadt=jacobian(EcSalida,Estado)*dxdt
```

```
dSalidadt =
```

$$\begin{pmatrix} \frac{x_2}{2} - x_1 \\ 5\,u_2 - 2\,u_1 - 2\,x_4 \end{pmatrix}$$

Comparando con el cálculo lineal (usual en los libros de control lineal) a partir de (A, B, C, D), en efecto, la derivada de la salida $y = Cx$ será $\frac{dy}{dt} = C\dot{x} = C(Ax + Bu) = CAx + CBu$

```
Cvelocidad=sys.C*sys.A
```

```
Cvelocidad = 2×4
   -1.0000    0.5000         0         0
         0         0         0   -2.0000
```

```
Dvelocidad=sys.C*sys.B
```

```
Dvelocidad = 2×2
     0     0
    -2     5
```

- La primera de las velocidades no es función directa de las entradas, hay que derivarla una vez mas;

la aceleración de la salida 1 es $\frac{d^2y_1}{dt^2}$:

```
d2salida1dt2=jacobian(dSalidadt(1),Estado)*dxdt
```

```
d2salida1dt2 =
```

$$3\,x_1 - u_1 + \frac{5\,x_2}{2} + 2\,x_3$$

Realmente, eso valdría incluso para no lineal, al usar "jacobian". El caso "lineal" podría manejarse con las matrices de la representación interna con $\ddot{y} = C\ddot{x} = C(A\dot{x} + B\dot{u}) = C\,A^2x + CABu + CB\dot{u}$, pero sólo hemos derivado aquellas salidas con $CB = 0$.

Como la primera fila de `Dvelocidad` es cero, hacemos otra vez lo mismo, la derivada de $\xi = C_{vel}x$ será $\dot{\xi} = C_{vel}Ax + C_{vel}Bu$:

```
Cacel1=Cvelocidad(1,:)*sys.A
```

```
Cacel1 = 1×4
    3.0000    2.5000    2.0000         0
```

```
Dacel1=Cvelocidad(1,:)*sys.B
```

```
Dacel1 = 1×2
    -1     0
```

Que concuerda con las fórmilas de arriba:

```
[sys.C(1,:)*sys.A^2    sys.C(1,:)*sys.A*sys.B]
```

```
ans = 1×6
    3.0000    2.5000    2.0000         0   -1.0000         0
```

Lo gastaremos todo junto luego

```
aceleracion1_y_velocidad2=[d2salida1dt2;dSalidadt(2)]
```

```
aceleracion1_y_velocidad2 =
```

$$\begin{pmatrix} 3\,x_1 - u_1 + \frac{5\,x_2}{2} + 2\,x_3 \\ 5\,u_2 - 2\,u_1 - 2\,x_4 \end{pmatrix}$$

Nota: como hemos dicho,realmente no hubiese sido necesario para nada la symbolic toolbox en el caso lineal, operando directamente con $\dot{x} = Ax + Bu,\ y = Cx$. La ventaja del Symbolic es que puede hacer lo mismo con sistemas NO LINEALES (la técnica que sigue entonces se denominará "desacoplamiento y ***linealización*** por realimentación del estado"), mientras que las fórmulas con C·A C·B y similar sólo valen para el caso lineal, evidentemente.

Desacoplamiento por realimentación del estado

El desacoplamiento por realimentación (también sería linealización si las expresiones hubiesen sido no lineales) implica poder calcular las entradas para que las derivadas adecuadas de las salidas tomen un valor arbitrario:

```
syms acel_y1 vel_y2 real %serán arbitrarias por el momento, luego las decidirá un contr
sol=solve(dSalidadt(2)==vel_y2,d2salida1dt2==acel_y1,{u1,u2});
sol.u1
```

```
ans =
```

$$3\,x_1 - \mathrm{acel}_{y1} + \frac{5\,x_2}{2} + 2\,x_3$$

```
sol.u2
```

```
ans =
```

$$\frac{\mathrm{vel}_{y2}}{5}-\frac{2\,\mathrm{acel}_{y1}}{5}+\frac{6\,x_1}{5}+x_2+\frac{4\,x_3}{5}+\frac{2\,x_4}{5}$$

No da errores: podemos hacer que la aceleración de la salida 1 y la velocidad de la 2 sean lo que nosotros queramos.

Obviamente, sin necesidad de simbólico, podemos hacerlo con las matrices (A, B, C, D), resultando:

```
Cacel1vel2=[Cacel1;Cvelocidad(2,:)]
```

```
Cacel1vel2 = 2×4
    3.0000    2.5000    2.0000         0
         0         0         0   -2.0000
```

```
Dacel1vel2=[Dacel1;Dvelocidad(2,:)]
```

```
Dacel1vel2 = 2×2
    -1     0
    -2     5
```

que coinciden con la versión Symbolic Toolbox:

```
aceleracion1_y_velocidad2
```

```
aceleracion1_y_velocidad2 =
```

$$\begin{pmatrix} 3\,x_1-u_1+\frac{5\,x_2}{2}+2\,x_3 \\ 5\,u_2-2\,u_1-2\,x_4 \end{pmatrix}$$

de modo que "despejar las entradas" es pasar lo que depende de x al otro lado (`Cacel1vel2`) e invertir lo que multiplica a u (esto es, `Dacel1vel2`):

```
FeedbackDecoupler=Dacel1vel2\[-Cacel1vel2 eye(2)]
```

```
FeedbackDecoupler = 2×6
    3.0000    2.5000    2.0000         0   -1.0000         0
    1.2000    1.0000    0.8000    0.4000   -0.4000    0.2000
```

El sistema será desacoplable por realimentación si tras derivar las salidas hasta que aparezcan las entradas, el $D_{derivadasdesalidas}$ adecuado es invertible. En este caso, si `Dacel1vel2` es invertible, que lo es.

Dinámica del sistema desacoplado

Las ecuaciones de estado en bucle cerrado tras el desacoplamiento por realimentación serán:

```
dxbcdt=subs(dxdt,{u1,u2},{sol.u1,sol.u2})
```

```
dxbcdt =
```

$$\begin{pmatrix} x_1 - \mathrm{acel}_{y1} \\ 2\,x_1 \\ x_2 \\ \frac{\mathrm{vel}_{y2}}{2} \end{pmatrix}$$

```
Adesac=eval(jacobian(dxbcdt,Estado))
```

```
Adesac = 4×4
     1     0     0     0
     2     0     0     0
     0     1     0     0
     0     0     0     0
```

```
Bdesac=eval(jacobian(dxbcdt,[acel_y1,vel_y2]))
```

```
Bdesac = 4×2
   -1.0000         0
         0         0
         0         0
         0    0.5000
```

Como las ecuaciones de salida no cambian (sólo hemos calculado unas entradas), podemos formar el sistema en bucle cerrado para analizar sus propiedades:

```
sysbc=ss(Adesac,Bdesac,sys.C,sys.D);
sysbc.InputName={'acel1','vel2'};
sysbc.OutputName={'y1','y2'};
```

En efecto, son los integradores en la diagonal:

```
minreal(tf(sysbc))
```

```
ans =

  From input "acel1" to output...
         1
   y1:  ---
        s^2

   y2:  0

  From input "vel2" to output...
   y1:  0

        1
   y2:  -
        s

Continuous-time transfer function.
```

Los polos y ceros son:

```
eig(sysbc)
```

```
ans = 4×1
     0
     0
     1
     0
```

```
tzero(sysbc)
```

```
ans = 1.0000
```

Análisis de la dinámica desacoplada por realimentación (incluyendo la no observable)

Todos los estados son controlables:

```
rank(ctrb(sysbc))
```

```
ans = 4
```

Pero hay dos no observables:

```
rank(obsv(sysbc))
```

```
ans = 3
```

Si eliminamos estados no observables:

```
sys2=minreal(sysbc);
```

```
1 state removed.
```

Resulta un sistema con 3 integradores (como la minreal de tf(sysbc), claro):

```
eig(sys2)
```

```
ans = 3×1
10^-7 ×
   -0.1286
    0.1286
         0
```

que no tiene ningún cero

```
tzero(sys2)
```

```
ans =

  0×1 empty double column vector
```

```
tf(sys2) %redondeos numéricos aparte, es diag(1/s^2,1/s)
```

```
ans =

  From input "acel1" to output...
                       s - 5.256e-35
   y1:  ---------------------------------------------
        s^3 + 9.166e-16 s^2 - 6.575e-16 s - 4.612e-36

                          1.933e-17
```

```
 y2:  ------------------------------------------
      s^3 + 9.166e-16 s^2 - 6.575e-16 s - 4.612e-36

From input "vel2" to output...
                -7.704e-34 s + 3.852e-34
 y1:  ------------------------------------------
      s^3 + 9.166e-16 s^2 - 6.575e-16 s - 4.612e-36

              s^2 - 2.288e-16 s - 3.664e-16
 y2:  ------------------------------------------
      s^3 + 9.166e-16 s^2 - 6.575e-16 s - 4.612e-36

Continuous-time transfer function.
```

Caso 1: la dinámica no observable es estable (cancelación polo/cero en lado izquierdo), habrá dos estados internos que "se moverán" siguiendo a las entradas nuevas, pero no se irán "a infinito", ni influenciarán en mis salidas. Si no existe ningún tipo de especificación de control o umbral de seguridad para ellos, pueden ser ignorados, y la estrategia funcionará en la práctica.

Caso 2: Se ha cancelado un cero en $s = +1$ (fase no mínima) y, por tanto, esta estrategia **NO funcionará** (no será internamente estable) en la práctica.

Ambos casos: formalmente requerimos "medir" el estado completo; El desacoplamiento dinámico no sería perfecto con observadores del estado y ruido de proceso/medida, o con errores de modelado. Dar valor a la "derivada quinta" de una salida en un sistema con mucho ruido y errores de modelado, o/y mal condicionado... igual no funciona (bueno, el caso 2 seguro que no). Dar valor a la "derivada primera" en un sistema bien condicionado funciona razonablemente bien (es una herramienta importante para "control cinemático de robots").