Linealización de un modelo de tanque de mezclado

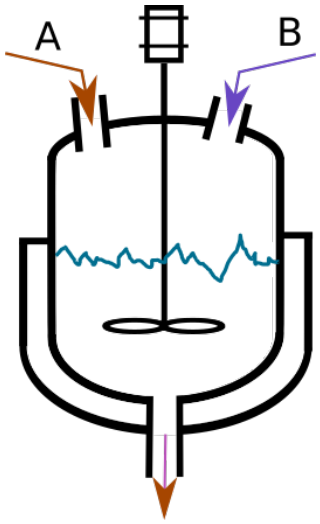
© 2019, Antonio Sala Piqueras, Universitat Politècnica de València. Todos los derechos reservados.

Este código ejecutó correctamente en Matlab R2019a

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

Motivación y objetivos: El objetivo de este vídeo es modelar un tanque de mezclado, incorporando no-linealidades en caudal de salida (raíz cuadrada) y concentración (cociente). Se linealizará y se comparará la simulación del modelo linealizado (vbles. incrementales) con el modelo original.

Modelado



Ec. físicas

```
syms v_a v_b dv_adt dv_bdt %2 estados y 2 deriv. estado
syms x_a x_b q_s h %4 otras variables que aparecen en ecs modelo
syms q_ain q_bin %2 variables de entrada
S=0.85;k_out=4; %parámetros constantes
Modelo=[ h==(v_a+v_b)/S; q_s==k_out*sqrt(h); dv_adt==-q_s*x_a+q_ain;...
         dv_bdt==-q_s*x_b+q_bin; x_a==v_a/(v_a+v_b); x_b==1-x_a];
```

Linealización

1.-Calculamos punto de funcionamiento

Por diseño, nos dicen que caudales entrada son 3 y 1 unidades. Resolvemos numéricamente ec. estado en equilibrio:

```
sol=vpasolve([Modelo;dv_adt==0;dv_bdt==0;q_ain==3;q_bin==1])
```

```
sol = struct with fields:
    dv_adt: [1×1 sym]
    dv_bdt: [1×1 sym]
         h: [1×1 sym]
```

```
      q_ain: [1×1 sym]
      q_bin: [1×1 sym]
        q_s: [1×1 sym]
        v_a: [1×1 sym]
        v_b: [1×1 sym]
        x_a: [1×1 sym]
        x_b: [1×1 sym]
```

```
q_apf=eval(sol.q_ain)
```

```
q_apf = 3
```

```
q_bpf=eval(sol.q_bin)
```

```
q_bpf = 1
```

```
v_apf=eval(sol.v_a)
```

```
v_apf = 0.6375
```

```
v_bpf=eval(sol.v_b)
```

```
v_bpf = 0.2125
```

```
qs_pf=eval(sol.q_s)
```

```
qs_pf = 4
```

```
xa_pf=eval(sol.x_a)
```

```
xa_pf = 0.7500
```

```
xb_pf=eval(sol.x_b)
```

```
xb_pf = 0.2500
```

```
h_pf=eval(sol.h)
```

```
h_pf = 1
```

Modelo linealizado en representación interna:

La linealización de las seis ecuaciones del modelo sería hacer deriv. parciales por incremento, pero para simulación, control, etc. conviene usar repr. interna normalizada, estados v_a y v_b, entradas $q_{a,in}$, $q_{b,in}$.

```
ReprInterna=solve(Modelo,dv_adt,dv_bdt,q_s,x_a,x_b,h);
derivEstado=[ReprInterna.dv_adt;ReprInterna.dv_bdt]
```

```
derivEstado =
```

$$\begin{pmatrix} q_{\text{ain}} - \dfrac{4\,v_a\,\sqrt{\dfrac{20\,v_a}{17}+\dfrac{20\,v_b}{17}}}{v_a+v_b} \\ q_{\text{bin}} + 4\,\sqrt{\dfrac{20\,v_a}{17}+\dfrac{20\,v_b}{17}}\,\left(\dfrac{v_a}{v_a+v_b}-1\right) \end{pmatrix}$$

Sólo nos interesa simular composición x_a y nivel h (elegidas arbitrariamente):

```
EcSalida=[ReprInterna.x_a;ReprInterna.h]
```

```
EcSalida =
```

$$\begin{pmatrix} \frac{v_a}{v_a+v_b} \\ \frac{20\,v_a}{17}+\frac{20\,v_b}{17} \end{pmatrix}$$

```
%linealizamos
%1: haciendo derivadas parciales
Asym=simplify(jacobian(derivEstado,[v_a v_b]))
```

```
Asym =
```

$$\begin{pmatrix} -\frac{\sqrt{85}\ (40\,v_a+80\,v_b)}{170\ (v_a+v_b)^{3/2}} & \frac{4\ \sqrt{85}\ v_a}{17\ (v_a+v_b)^{3/2}} \\ \frac{4\ \sqrt{85}\ v_b}{17\ (v_a+v_b)^{3/2}} & -\frac{4\ \sqrt{85}\ (2\,v_a+v_b)}{17\ (v_a+v_b)^{3/2}} \end{pmatrix}$$

```
Bsym=simplify(jacobian(derivEstado,[q_ain q_bin]))
```

```
Bsym =
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```
Csym=simplify(jacobian(EcSalida,[v_a v_b]))
```

```
Csym =
```

$$\begin{pmatrix} \frac{v_b}{(v_a+v_b)^2} & -\frac{v_a}{(v_a+v_b)^2} \\ \frac{20}{17} & \frac{20}{17} \end{pmatrix}$$

```
Dsym=simplify(jacobian(EcSalida,[q_ain q_bin]))
```

```
Dsym =
```

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

```
%2: las deriv. parciales son evaluadas en punto de funcionamiento
Alin=eval(subs(Asym,sol))
```

```
Alin = 2×2
   -2.9412    1.7647
    0.5882   -4.1176
```

```
Blin=eval(subs(Bsym,sol))
```

```
Blin = 2×2
     1     0
     0     1
```

```
Clin=eval(subs(Csym,sol))
```

```
Clin = 2×2
    0.2941   -0.8824
    1.1765    1.1765
```

```
Dlin=eval(subs(Dsym,sol))
```

```
Dlin = 2×2
     0     0
     0     0
```

El modelo linealizado tiene esta ecuación de estado:

```
dxdt_lin=@(incx,incu) Alin*incx+Blin*incu;
```

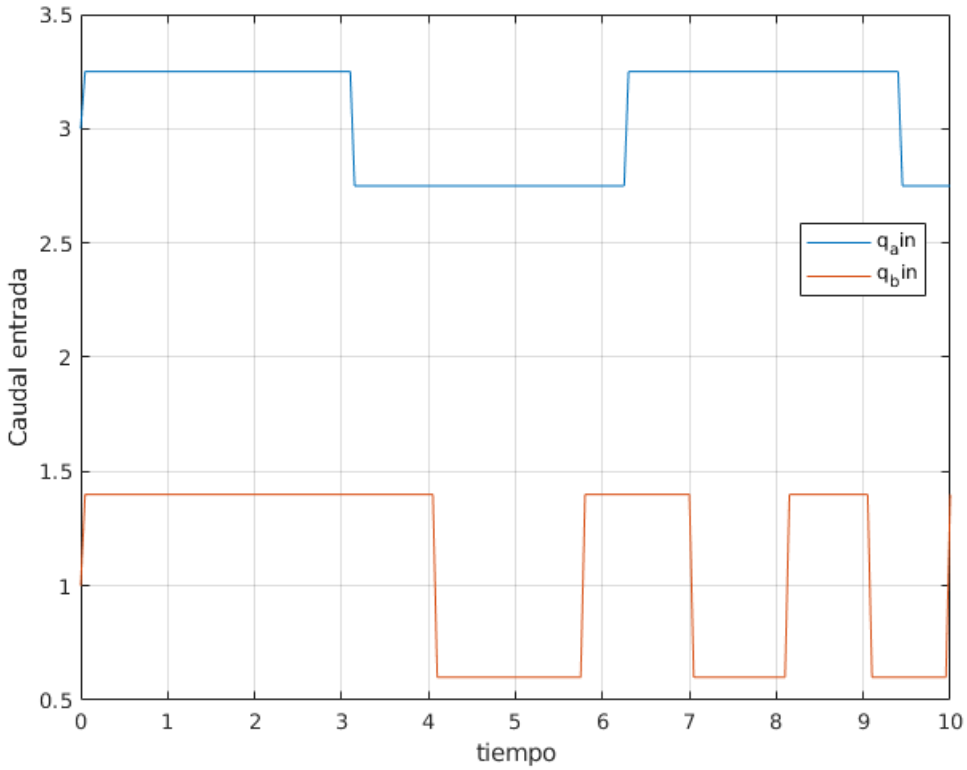
y esta ecuación de salida:

```
EcSalidaLinealizado=@(incx,incu) Clin*incx+Dlin*incu;
```

Simulación del modelo original y el linealizado

Las entradas (caudales de A y B) tendrán la siguiente forma:

```
Tiempos=0:0.05:10;
q_ain_simula=q_apf+0.25*sign(sin(Tiempos));
q_bin_simula=q_bpf+0.4*sign(sin(Tiempos.^2*0.19));
vector_entradas_sim=[q_ain_simula;q_bin_simula];
plot(Tiempos,vector_entradas_sim), grid on
legend('q_ain','q_bin','Location','best'), xlabel('tiempo'), ylabel('Caudal entrada')
```



Simulación de la ecuación de estado

La ecuación NO lineal de estado es:

```
dxdt=matlabFunction(derivEstado,'Vars',{[v_a;v_b],[q_ain;q_bin]});
```

Preparamos ode45 para simular el no lineal:

```
CondInicial=[0.05;0.2];
modeloparasimular=@(t,estado) dxdt(estado, interp1(Tiempos,vector_entradas_sim',t)');
[tiempossimulados,estadosimulado]=ode45(modeloparasimular,Tiempos,CondInicial);
```

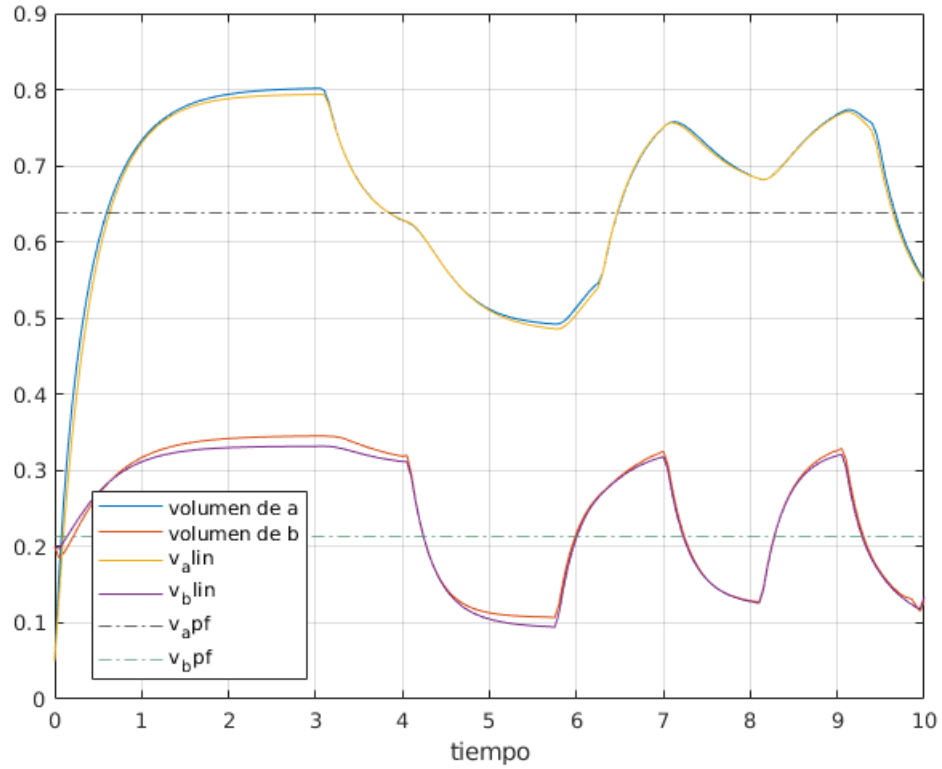
Preparamos ode45 para simular el linealizado (entradas, estados y salidas son incrementales):

```
entrada_incre=vector_entradas_sim-[q_apf;q_bpf];
CondInic_incre=CondInicial-[v_apf;v_bpf];
modelsimulalinealiz=@(t,estado) dxdt_lin(estado, interp1(Tiempos,entrada_incre',t)');
[tiempossimulaL,inc_estadosimul]=ode45(modelsimulalinealiz,Tiempos,CondInic_incre);
```

Comparamos ambos resultados (trayectorias vbles. de estado v_a, v_b):

```
plot(tiempossimulados,estadosimulado), grid on, hold on
plot(tiempossimulaL,inc_estadosimul+[v_apf v_bpf]), hold off %deshacemos incrementos!
```

```
yline(v_apf,'-.');yline(v_bpf,'-.','Color','#559977');
legend('volumen de a','volumen de b','v_alin','v_blin','v_apf','v_bpf','Location','best
```



Evaluación de la ecuación de salida

Una vez simuladas las ecuaciones de estado, hay que sustituirlas en la de salida.

En el modelo no-lineal:

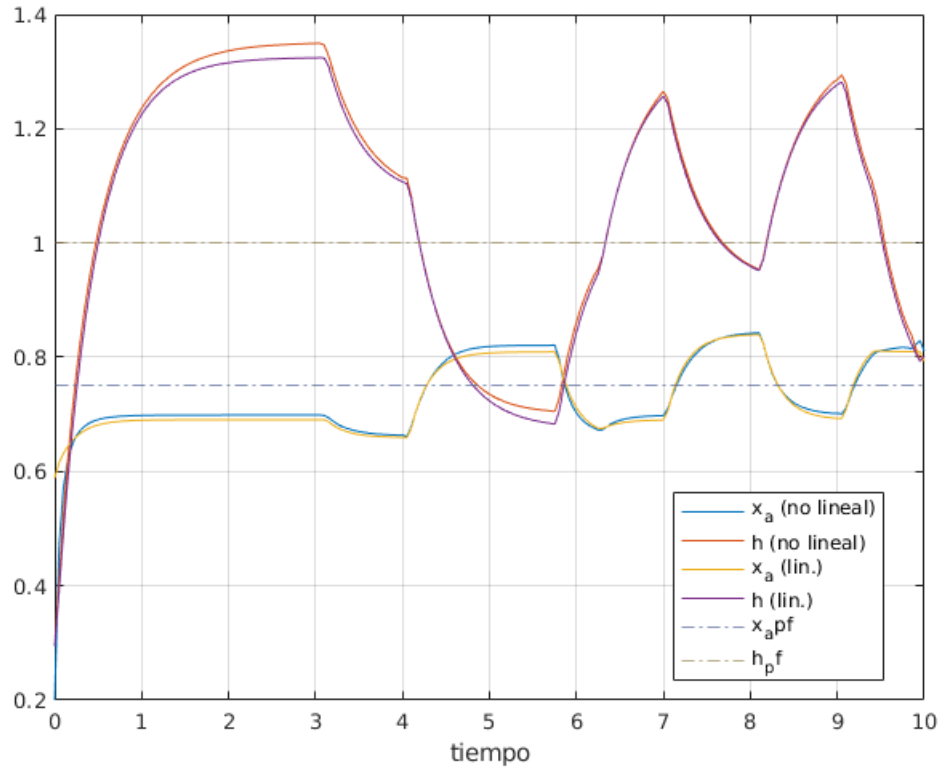
```
EcSal_numerica=matlabFunction(EcSalida,'Vars',{[v_a;v_b]});
salidassimuladas=EcSal_numerica(estadosimulado');
```

En el modelo linealizado resulta

```
inc_salidas_linealizado=EcSalidaLinealizado(inc_estadosimul',entrada_incre);
```

Lo visualizamos todo en una gráfica para comparar:

```
plot(Tiempos,salidassimuladas), grid on, hold on
plot(Tiempos,inc_salidas_linealizado+[xa_pf;h_pf]), hold off %deshacemos incrementos
yline(xa_pf,'-.','Color','#556699');yline(h_pf,'-.','Color','#998855');
legend('x_a (no lineal)','h (no lineal)','x_a (lin.)','h (lin.)','x_apf','h_pf','Locati
```



Si almacenamos el modelo linealizado como un "objeto sistema lineal" de la Control System Toolbox, podremos analizarlo y controlarlo (no en objetivos de este material) con las herramientas de dicha toolbox:

```
TanqueMezcladoLin=ss(Alin,Blin,Clin,Dlin);
TanqueMezcladoLin.InputName={'q_ain','q_bin'};
TanqueMezcladoLin.OutputName={'x_a','h'};
TanqueMezcladoLin.StateName={'vol_a','vol_b'};
size(TanqueMezcladoLin)
```

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

```
TanqueMezcladoLin
```

```
TanqueMezcladoLin =

  A =
           vol_a   vol_b
   vol_a  -2.941   1.765
   vol_b  0.5882  -4.118

  B =
          q_ain  q_bin
   vol_a      1      0
   vol_b      0      1

  C =
          vol_a    vol_b
   x_a   0.2941  -0.8824
   h      1.176    1.176
```

```
D =
       q_ain  q_bin
 x_a       0      0
 h         0      0

Continuous-time state-space model.
```

```
save ModeloTanqueLin.mat TanqueMezcladoLin
```