

Diseño de control para modelo linealizado de tanque de mezclado (V2): Control multibucle de nivel y concentración

© 2020, Antonio Sala Piqueras. Universitat Politècnica de València. Todos los derechos reservados.

Presentación en vídeo: <http://personales.upv.es/asala/YT/V/tmrml.html>

Este código funcionó correctamente con Matlab R2019b

Objetivos: Diseñar y simular en bucle cerrado un control multibucle para el tanque de mezclado siguiente, suponiendo ya diseñado previamente el control en cascada.

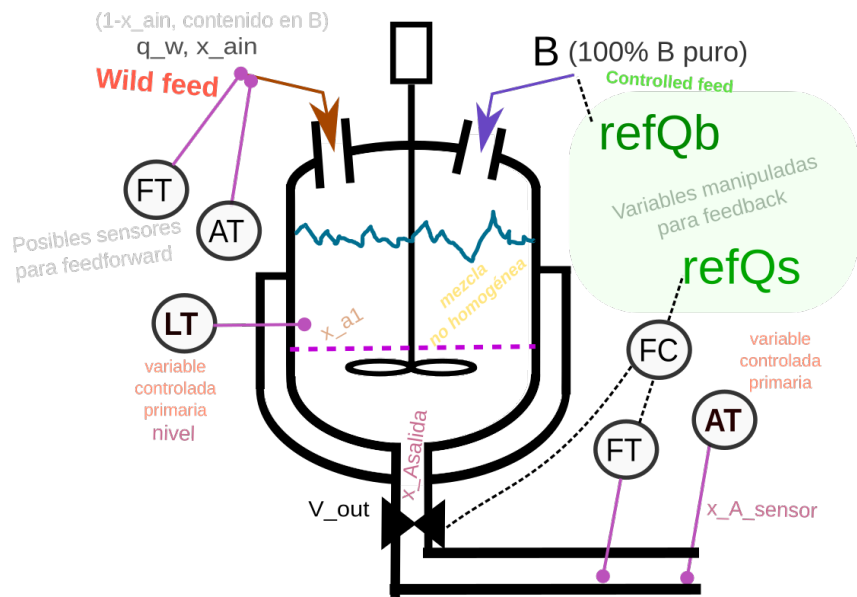
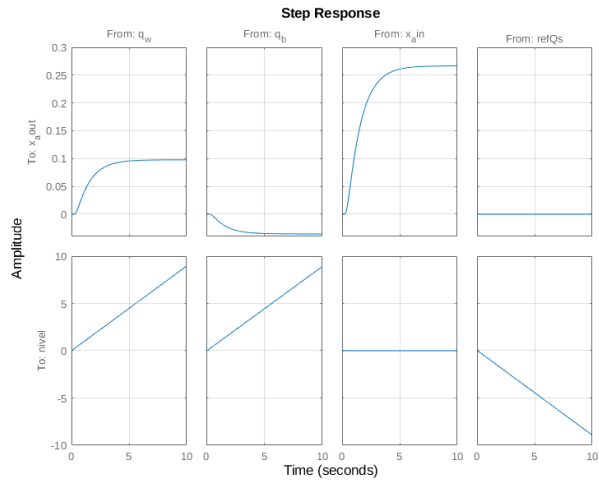


Tabla de Contenidos

*Preliminares: carga del modelo.....	1
1.- Emparejamiento.....	2
2.- Diseño de bucles de nivel/concentración de forma independiente.....	2
2.1- Control SISO de Nivel (PI).....	2
2.2- Control SISO de concentración de A (PI).....	4
3.- Simulación de ambos lazos de control a la vez.....	5

*Preliminares: carga del modelo

```
load ModeloConCascada.mat
s=tf('s');
step(G2,10), grid on
```



1.- Emparejamiento

Como las entradas manipuladas son q_{bin} y $refQ$ (las otras son perturbaciones), y $refQ$ no afecta la composición, no hace falta RGA para decidir que controlaremos composición con q_{bin} , y nivel con ref_Q . De todos modos, la RGA de una matriz triangular coincide con dicha intuición:

```
GanAprox=evalfr(G2(:, [2 4]), 2e-3)
```

```
GanAprox = 2x2
-0.0354    0.0000
447.8532 -447.8606
```

```
RGA=GanAprox.*inv(GanAprox')
```

```
RGA = 2x2
1.0000    -0.0000
-0.0000    1.0000
```

2.- Diseño de bucles de nivel/concentración de forma independiente

2.1- Control SISO de Nivel (PI)

```
G_nivel_refQs=minreal(G2(2,4));
```

```
3 states removed.
```

```
zpk(G_nivel_refQs)
```

```
ans =
```

```
From input "refQs" to output "nivel":
-4.7073 (s+92.74)
-----
s (s^2 + 25.53s + 487.3)
```

```
Continuous-time zero/pole/gain model.
```

```
[num,den]=tfdata(tf(G_nivel_refQs));
```

```
[res,pol]=residue(num{:},den{:})
```

```
res = 3x1 complex
    0.4479 - 0.1868i
    0.4479 + 0.1868i
   -0.8958 + 0.0000i
pol = 3x1 complex
  -12.7672 +18.0097i
  -12.7672 -18.0097i
   -0.0000 + 0.0000i
```

El cero no en origen y la frecuencia natural elevada podrían ser considerados "muy rápidos" si sólo buscamos especificaciones con constantes de tiempo de "segundos"... en ese caso, podríamos aproximar para el diseño por

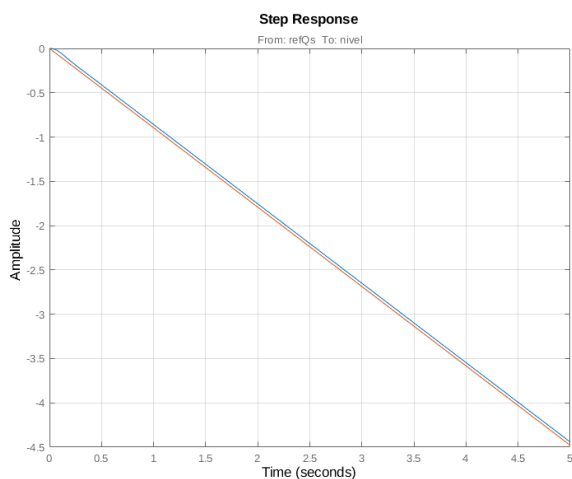
```
Gnivelapprox=res(3)/s
```

```
Gnivelapprox =
```

```
  -0.8958
  -----
         s
```

```
Continuous-time transfer function.
```

```
step(G_nivel_refQs,Gnivelapprox,5), grid on
```



En resumen, podríamos gastar Gnivelapprox como modelo para diseñar controladores que tardaran un par de segundos en estabilizarse.

Como el proceso para control de nivel es "casi" un integrador, podriamos pensar en un control *proporcional*, pero NO VALE porque tiene error ante perturbaciones a la entrada del proceso, cuya cancelación es el objetivo principal de este caso de estudio. Con integrador en la planta el error es cero ante REFERENCIA, pero no ante perturbación de entrada. Ello hará que el nivel tenga error ante incrementos de caudal o composición en el "wild feed" (componente A).

Gastaremos un PI como solución final, pero compararemos ambos diseños.

```
t_subida_nivel=1;
anchobandanivel=2.2/t_subida_nivel; %cuanto más ancho de banda, más "rápido".
K_nivel=pidtune(G_nivel_refQs,'PI',anchobandanivel,pidtuneOptions('DesignFocus','disturbar
```

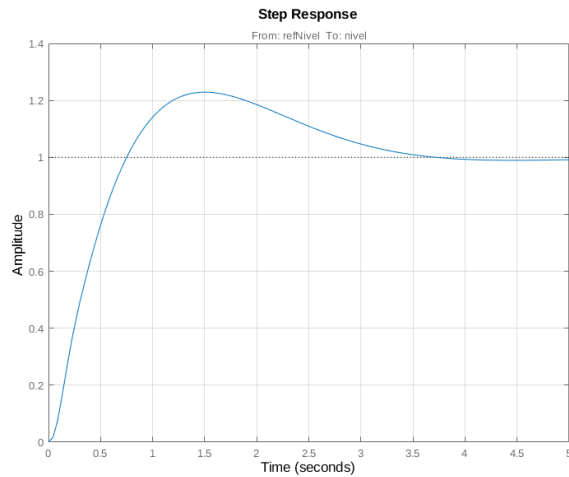
```
K_nivel =
```

$$K_p + K_i * \frac{1}{s}$$

with $K_p = -2.22$, $K_i = -2.25$

Continuous-time PI controller in parallel form.

```
K_nivel.InputName='eh'; K_nivel.OutputName='refQs';
BC_nivel=feedback(G_nivel_refQs*K_nivel,1);
BC_nivel.InputName='refNivel';
step(BC_nivel), grid on
```



2.2- Control SISO de concentración de A (PI)

```
G_xaout_qbin=minreal(G2(1,2),1e-7);tf(G_xaout_qbin)
```

3 states removed.

```
ans =
```

```
From input "q_b" to output "x_aout":
      -2.988e-11 s - 2.729
exp(-0.249*s) * -----
      s^3 + 25.39 s^2 + 111.5 s + 76.76
```

Continuous-time transfer function.

```
t_subida_concentr=4.1;
anchobandaconc=2.2/t_subida_concentr; %cuanto más ancho de banda, más "rápido".
K_conc=pidtune(G_xaout_qbin,'PI',anchobandaconc,pidtuneOptions('DesignFocus','disturbar
```

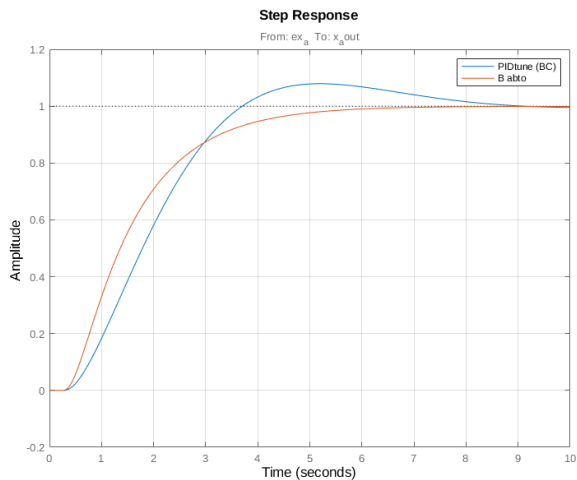
K_conc =

$$K_p + K_i * \frac{1}{s}$$

with Kp = -10.6, Ki = -17.1

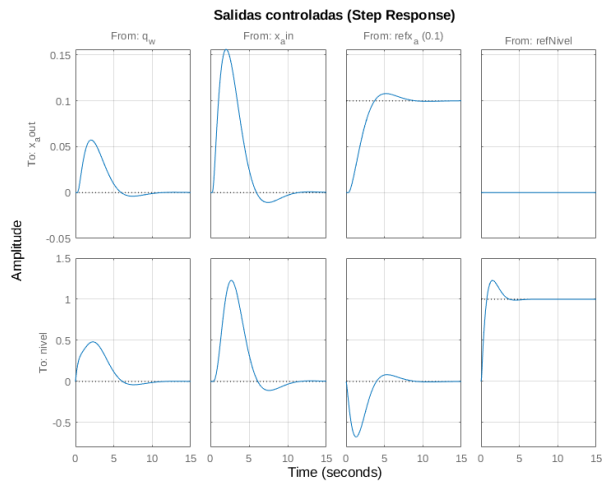
Continuous-time PI controller in parallel form.

```
K_conc.InputName='ex_a'; K_conc.OutputName='q_b';
BC_conc=feedback(G_xaout_qbin*K_conc,1);
step(BC_conc,G_xaout_qbin/dcgain(G_xaout_qbin)), grid on, legend('PIDtune (BC)', 'B alto')
```



3.- Simulación de ambos lazos de control a la vez

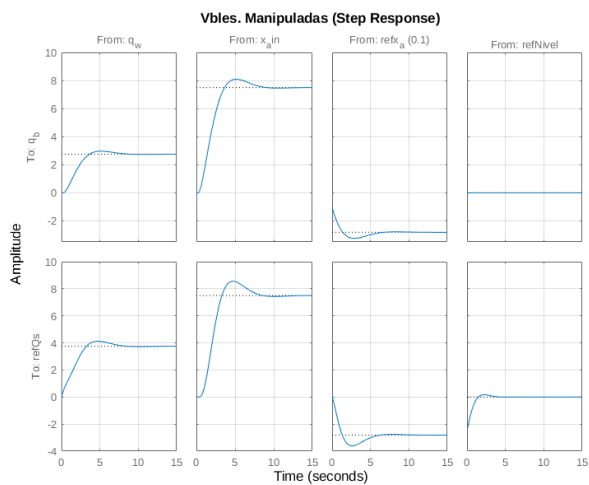
```
SumH=sumblk('eh=refNivel-nivel');
SumXa=sumblk('ex_a=refx_a-x_aout');
BC_multibucle=connect(G2,K_nivel,K_conc,SumH,SumXa, ...
    {'q_w','x_ain','refx_a','refNivel'}, ...
    {'x_aout','nivel','q_b','refQs','eh','ex_a'});
cosa=BC_multibucle*diag([1 1 .1 1]);%solo amplitud de 10% en concentracion...
cosa.InputName={'q_w','x_ain','refx_a (0.1)','refNivel'};
step(cosa(1:2,:),15), grid on, title('Salidas controladas (Step Response)')
```



Las variables controladas tienen error de posición cero ante referencia y ante perturbaciones, con los dos PI. Para reducir el efecto de q_w y $x_{a,in}$ así como el acoplamiento $refx_a \rightarrow nivel$, podría hacerse una realimentación más enérgica o/y incorporar feedforward y desacoplamiento en el bucle (fuera de objetivos de este material).

Si dibujamos las acciones de control:

```
step(cosa(3:4,:),15), grid on, title('Vbles. Manipuladas (Step Response)')
```



No se aprecia ningún comportamiento dinámico "extraño", con lo que podría pasar a simularse, por ejemplo, con el modelo no lineal.

```
save Multibucle.mat K_conc K_nivel BC_multibucle
```