Diseño de controlador IMC para implementación discreta (I): diseños basados en tiempo continuo

Presentación en vídeo: http://personales.upv.es/asala/YT/V/imcdt1.html

Este código funcionó correctamente con Matlab **R2021b**

Objetivo: Comparar varias opciones de diseño de reguladores IMC para ser implementados por computador (tiempo discreto), pero basadas en un diseño en tiempo contínuo (el diseño directo discreto se verá en otros materiales).

Tabla de Contenidos

Preliminares

Modelo en tiempo continuo

```
Ac=[0 1 ;-3 -1.1]; Bc=[0;3]; C=[1 0];
sysc=ss(Ac,Bc,C,0);sysc.InputName='ureal';sysc.OutputName='y';
gan=dcgain(sysc)
```

```
gan = 1
```

```
tf(sysc)
```

```
ans =

  From input "ureal" to output "y":
         3
  ---------------
  s^2 + 1.1 s + 3

Continuous-time transfer function.
```

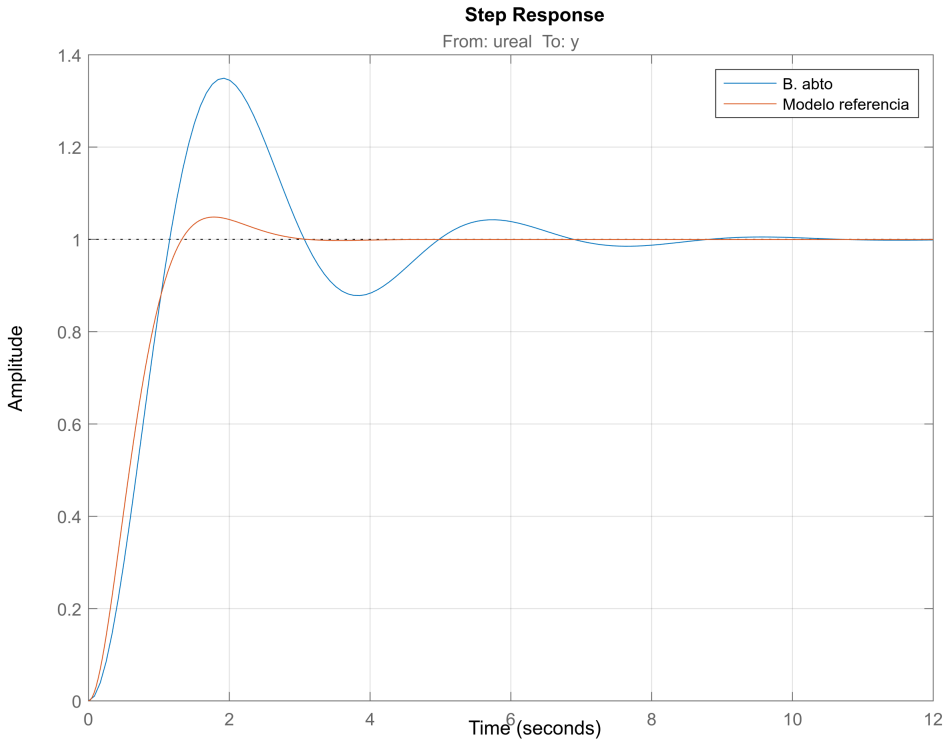
Objetivos de control

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

```
ModeloRefContinuo=6/(s^2+3.4*s+6);
eig(ModeloRefContinuo)
```

```
ans = 2×1 complex
  -1.7000 + 1.7635i
  -1.7000 - 1.7635i
```

```
step(sysc,ModeloRefContinuo), grid on, legend("B. abto","Modelo referencia")
```

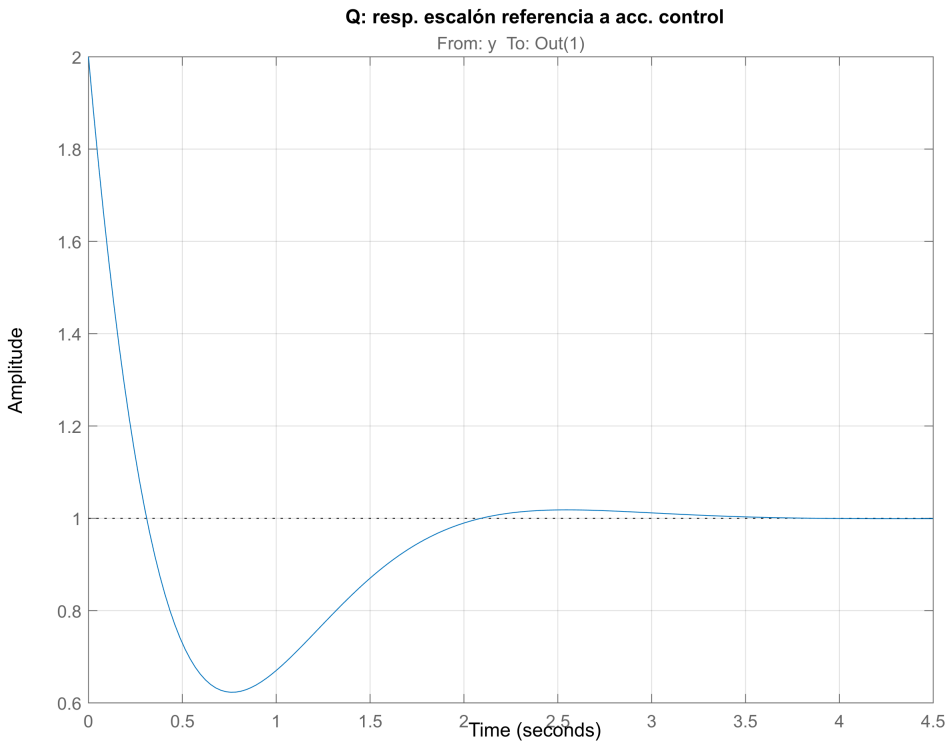


Diseño Control IMC en tiempo contínuo

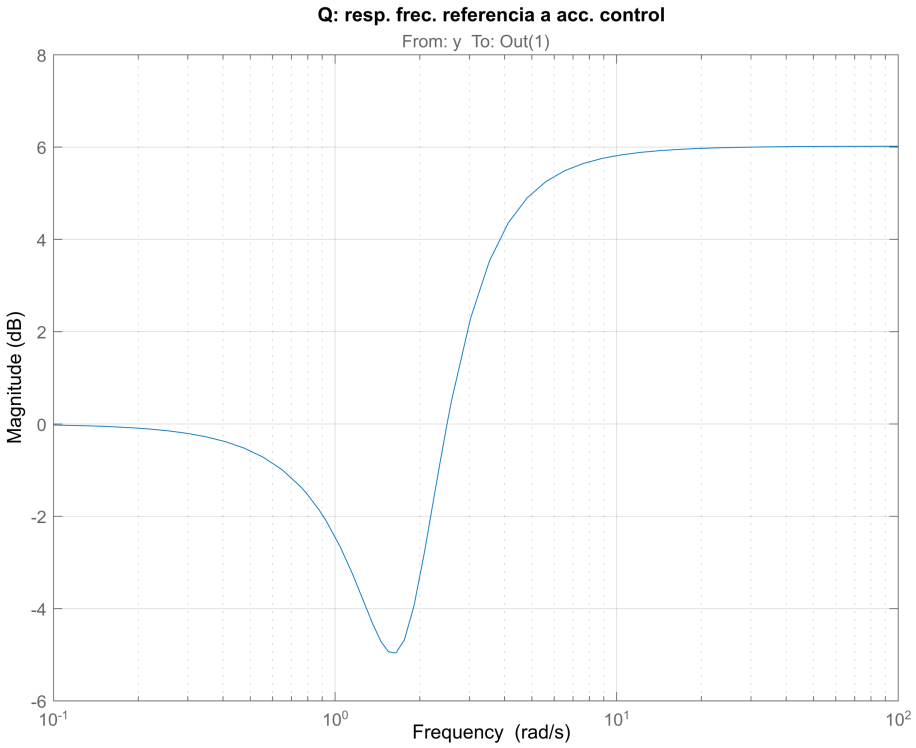
```
Qc=minreal(ModeloRefContinuo/sysc); zpk(Qc) %Q cancela la planta y la sustituye por otr
```

```
3 states removed.

ans =

  From input "y" to output:
  2 (s^2 + 1.1s + 3)
  ------------------
   (s^2 + 3.4s + 6)

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

```
step(Qc), grid on, title("Q: resp. escalón referencia a acc. control")
```




```
bodemag(Qc), grid on, title("Q: resp. frec. referencia a acc. control")
```



A partir de Q (parámetro de Youla), formamos el regulador "convencional" (IMC implícito):

```
Kc=minreal(feedback(Qc,-sysc));
```

```
2 states removed.
```

```
Kc.InputName='e'; Kc.OutputName='u';
zpk(Kc)
```

```
ans =

  From input "e" to output "u":
  2 (s^2 + 1.1s + 3)
  ------------------
      s (s+3.4)

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

Pero este diseño será implementado por computador y, por tanto, nuestro objetivo será obtener algo "parecido" en tiempo discreto.

Tenemos las opciones de utilizar `Q` o/y Kc para calcular un control discreto (objetivo de este material)

o

Selección período de muestreo

Nuestro objetivo será $t_{est} \approx 2$ s, $\omega_{propia} = 1.76$ rad/s, $\omega_n = \sqrt{6} = 2.45$ rad/s.

Períodos bien elegidos serían:

```
Ts=0.1;
%todas las opciones bien, quizás elegir esto si vamos a usar
%discretización de diseño continuo, por ir "a lo seguro".

Ts=0.25;
%también razonable, hasta aquí llegaríamos con las reglas de "andar por casa".
```

Otras opciones menos aconsejables "intuitivamente" (porque hay "pocas muestras" en tiempo de subida/establecimiento/frecuencia propia, porque se "mide poco" y las perturbaciones de alta frecuencia darán "aliasing", etc...):

```
%Ts=0.5; %discretizado tustin regular, otras opciones resultan aceptables
%Ts=1.15; %tustin bastante mal a partir de este período (incluso inestable a partir de
```

Luego veremos simulaciones para comprobar si está o no bien escogido.

Nota: el ruido de medida de alta frecuencia podría recomendar T_s pequeño (para tener más muestras que "promediar") para "observar" correctamente al sistema, independientemente de las necesidades de T_s para "controlar" correctamente. En general, mejor cuando más pequeño sea T_s.

Diseño IMC discreto a partir del contínuo

Opción 1: discretización del regulador

```
Kd1=c2d(Kc,Ts,'Tustin');
zpk(Kd1)
```

```
ans =

  From input "e" to output "u":
  1.6623 (z^2 - 1.609z + 0.7678)
  ------------------------------
         (z-1) (z-0.4035)

Sample time: 0.25 seconds
Discrete-time zero/pole/gain model.
```

Esto finalizaría el diseño, que luego simularemos, comparando con el otro diseño a continuación. NO hay garantía de estabilidad.

Opción 2: diseño contínuo teniendo en cuenta efecto discretización.

Para un regulador genérico, se recomienda si el período de muestreo es pequeño, pero "no tanto para que la opción 1 funcione bien", el rediseñar añadiendo un retardo de medio muestreo... bueno, su aproximación $e^{-\frac{T_s}{2}s} \approx (1 - \frac{T_s}{2}s)$.

Como el factor de fase no mínima no debe cancelarse, el resultado sería un parámetro Q idéntico, y un regulador:

```
factorMuestreo=(1-Ts*s/2);zpk(factorMuestreo)
```

```
ans =

  -0.125 (s-8)

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

```
Kc2=minreal(feedback(Qc,-sysc*factorMuestreo));
```

```
4 states removed.
```

```
Kc2.InputName='e'; Kc2.OutputName='u';
zpk(Kc2)
```

```
ans =

  From input "e" to output "u":
  2 (s^2 + 1.1s + 3)
```

```
  ------------------
      s (s+4.15)

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

```
Kd2=c2d(Kc2,Ts,'tustin');
zpk(Kd2)
```

```
ans =

  From input "e" to output "u":
  1.5597 (z^2 - 1.609z + 0.7678)
  ------------------------------
         (z-0.3169) (z-1)

Sample time: 0.25 seconds
Discrete-time zero/pole/gain model.
```

Opción 2b: Realmente, en vez de `sysc*(1+Ts*s/2)` debería diseñarse el Q de IMC para:

```
sysc_zoh=d2c(c2d(sysc,Ts,'zoh'),'tustin');
zpk(sysc_zoh)
```

```
ans =

  From input "ureal" to output "y":
  -0.002214 (s-8) (s+173.7)
  -------------------------
   (s^2 + 1.146s + 3.076)

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

Esa última opción garantiza estabilidad en bucle cerrado al discretizar por Tustin un regulador que estabilice a `sysc_zoh` ... En el caso particular IMC, eso es muy parecido a la opción 3 abajo (usando Tustin `c2d(feedback(Q,sysc_zoh)) = feedback(c2d(Q),c2d(sysc_zoh))`, y `c2d(sysc_zoh)` es la discretización zoh del proceso.

La diferencia es que `sysc_zoh` tiene los polos y ceros diferentes a `sysc` y eso cambiaría el diseño de Q... pero también el modelo de referencia sufriría una deformación de la respuesta transitoria/frecuencial al hacer tustin si el período no es pequeño... con lo cual, para meterse en esos "berenjenales" mejor hacer diseño discreto directo: el diseño de reguladores discretos basado en cálculos en continuo debe mantener cierta "simplicidad" para merecer la pena.

Opción 3: discretización del parámetro de Youla Q

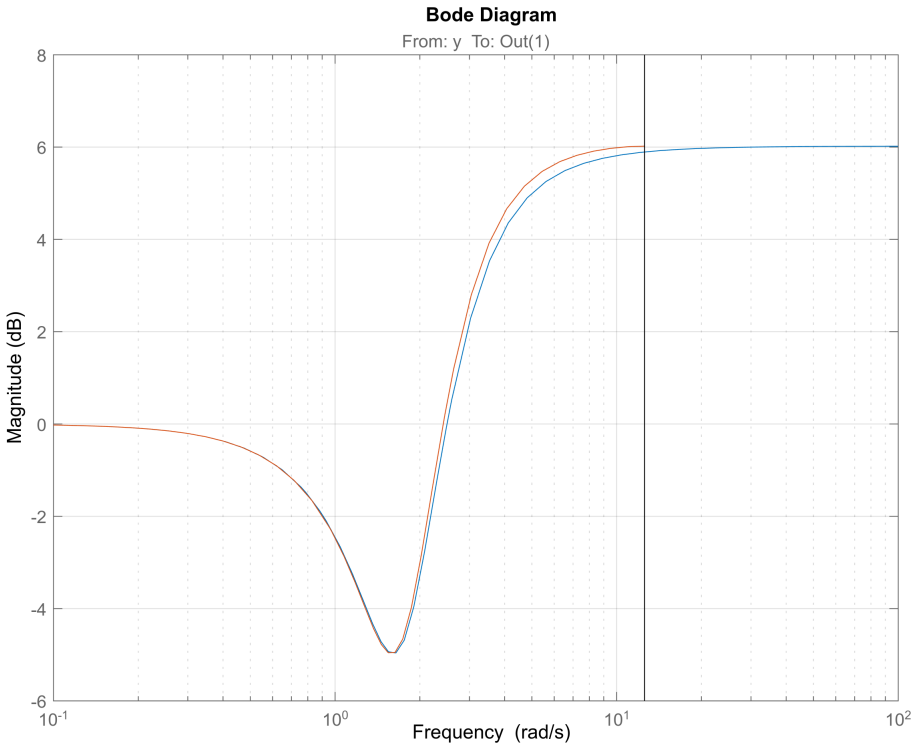
En vez de discretizar "Kc", con garantía de estabilidad del regulador pero no del bucle cerrado, podemos discretizar Q, que daría un Q_d discreto estable. Eso, como el proceso controlado es estable, garantizará estabilidad en bucle cerrado (si Q_d es estable, el bucle IMC es estable):

```
Qd=minreal(c2d(Qc,Ts,'tustin'));
```

```
zpk(Qd)
```

```
ans =

  From input "y" to output:
  1.5597 (z^2 - 1.609z + 0.7678)
  ------------------------------
     (z^2 - 1.193z + 0.4403)

Sample time: 0.25 seconds
Discrete-time zero/pole/gain model.
```

```
bodemag(Qc,Qd), grid on
```

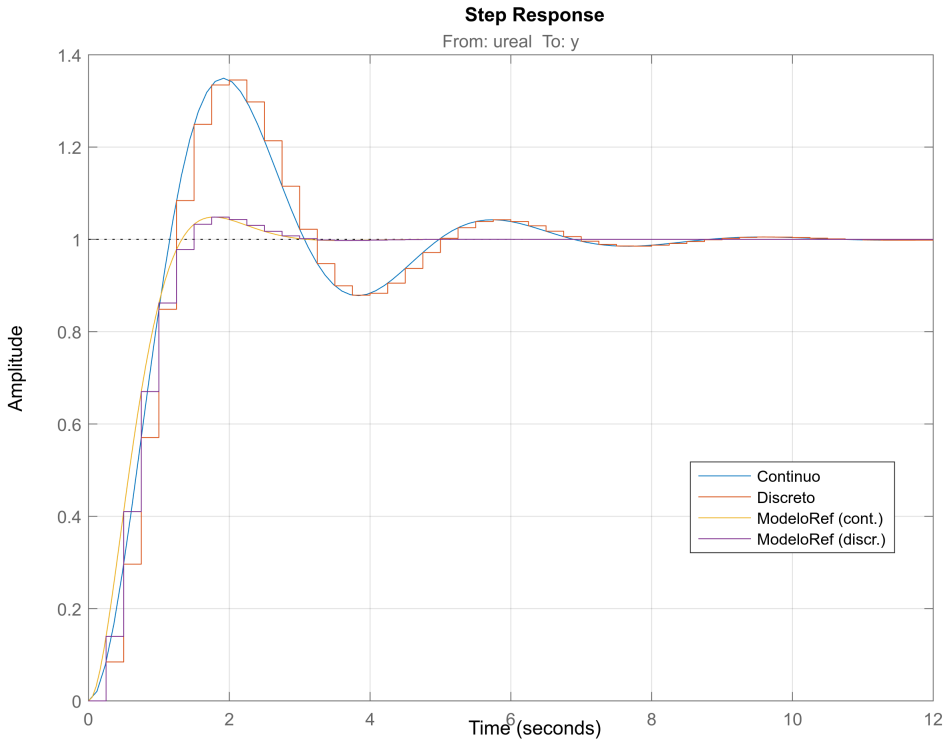


Con ese parámetro Q discretizado (que seguro será estable, porque Tustin preserva estabilidad), el IMC se completaría con la discretización del proceso:

```
sysd=c2d(sysc,Ts,'zoh');
```

Comprobemos si el período de muestreo está bien elegido

```
step(sysc,sysd,ModeloRefContinuo, c2d(ModeloRefContinuo,Ts,'zoh'),12), grid on
legend('Continuo','Discreto','ModeloRef (cont.)','ModeloRef (discr.)',Location="best")
```



El regulador final en esta opción sería:

```
Kd3=minreal(feedback(Qd,-sysd));
Kd3.InputName='e';Kd3.OutputName='u';
zpk(Kd3)
```

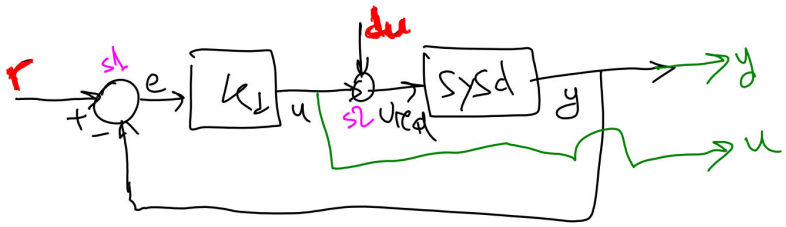
```
ans =

  From input "e" to output "u":
  1.5597 (z^2 - 1.609z + 0.7678) (z^2 - 1.598z + 0.7596)
  ---------------------------------------------------
         (z-0.315) (z-1) (z^2 - 1.608z + 0.7692)

Sample time: 0.25 seconds
Discrete-time zero/pole/gain model.
```

y finaliza el diseño, con garantía de estabilidad. Aunque, bueno, el tustin de "`Q`" no tiene los ceros en el mismo sitio que polos de `sysd`, y por tanto sale de orden elevado al cerrar el bucle para construir Kd3 (no hay cancelaciones).

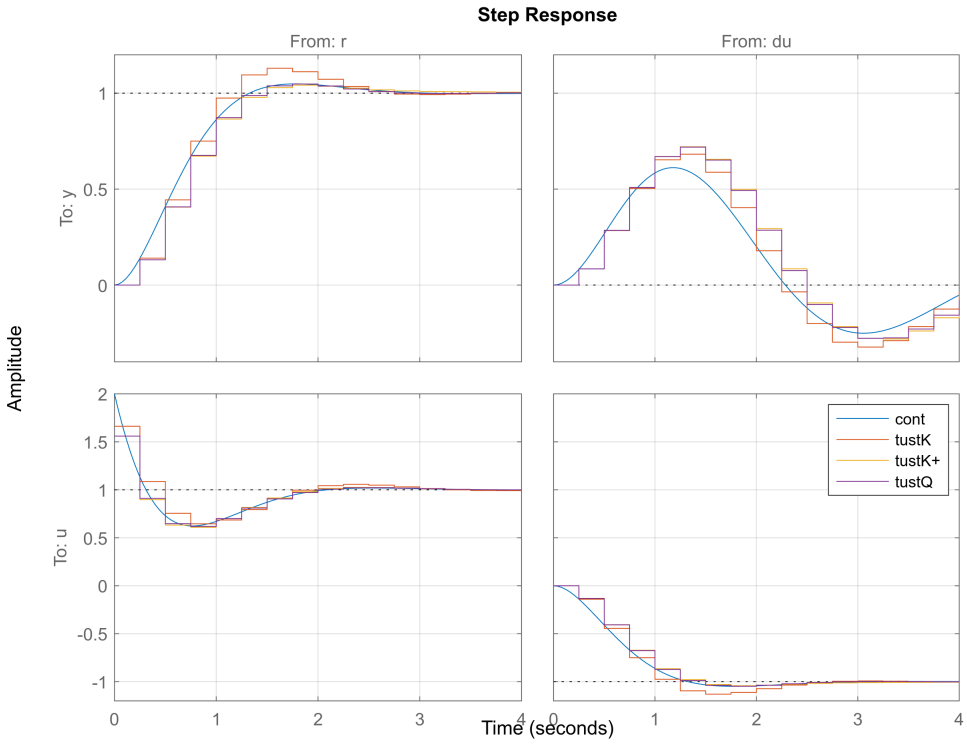
Prestaciones en bucle cerrado

Prestaciones en bucle cerrado discreto

Implementemos el siguiente diagrama de bloques con "connect":



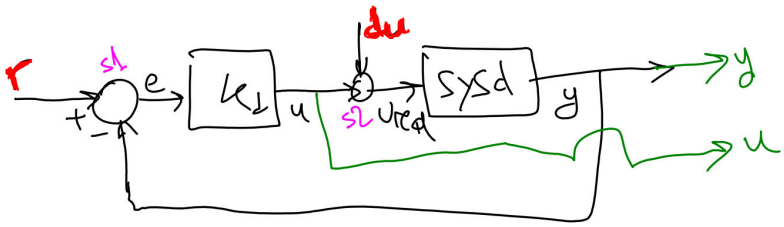

```
s1=sumblk('e=r-y');s2=sumblk('ureal=u+du');
BCcont=connect(s1,s2,Kc,sysc,{'r','du'},{'y','u'});
BCtustK=connect(s1,s2,Kd1,sysd,{'r','du'},{'y','u'}); %diseño discretizando K por tusti
BCtustK2=connect(s1,s2,Kd2,sysd,{'r','du'},{'y','u'}); %diseño teniendo en cuenta efect
BCtustQ=connect(s1,s2,Kd3,sysd,{'r','du'},{'y','u'}); %diseño discretizando Q, garantía
step(BCcont,BCtustK,BCtustK2,BCtustQ,  4), grid on, legend('cont','tustK','tustK+','tus
```



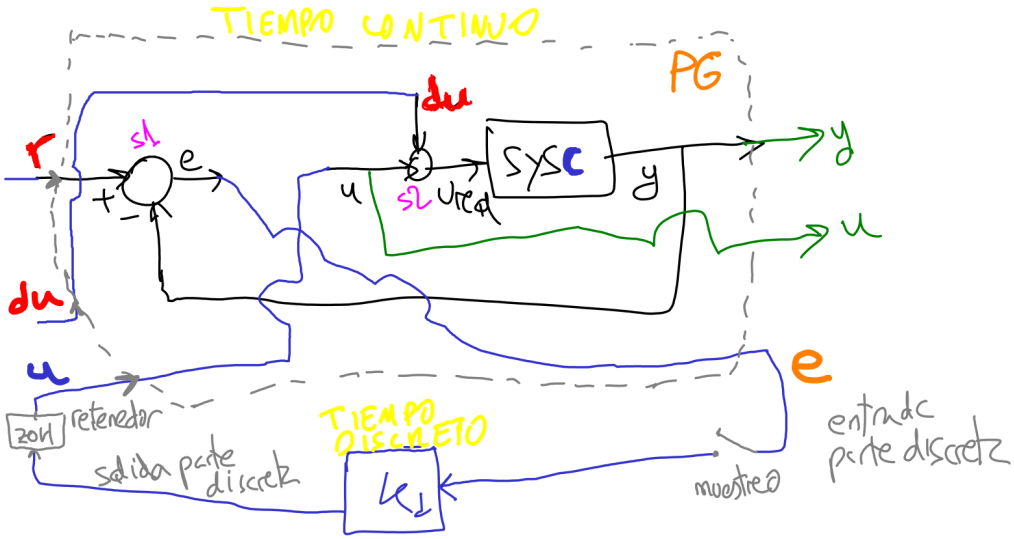
Prestaciones intermuestreo

El proceso controlado es realmente continuo, `sdlsim` puede simular respuesta intermuestreo.

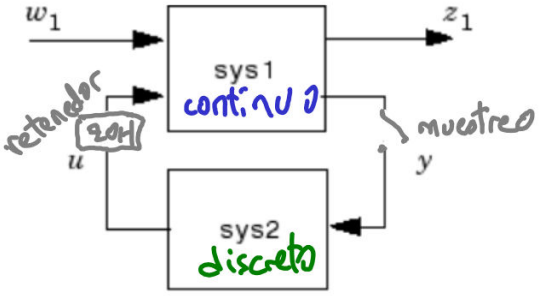
Debemos separar todo lo que es contínuo y todo lo que es discreto del diagrama original:



convirtiéndolo a:



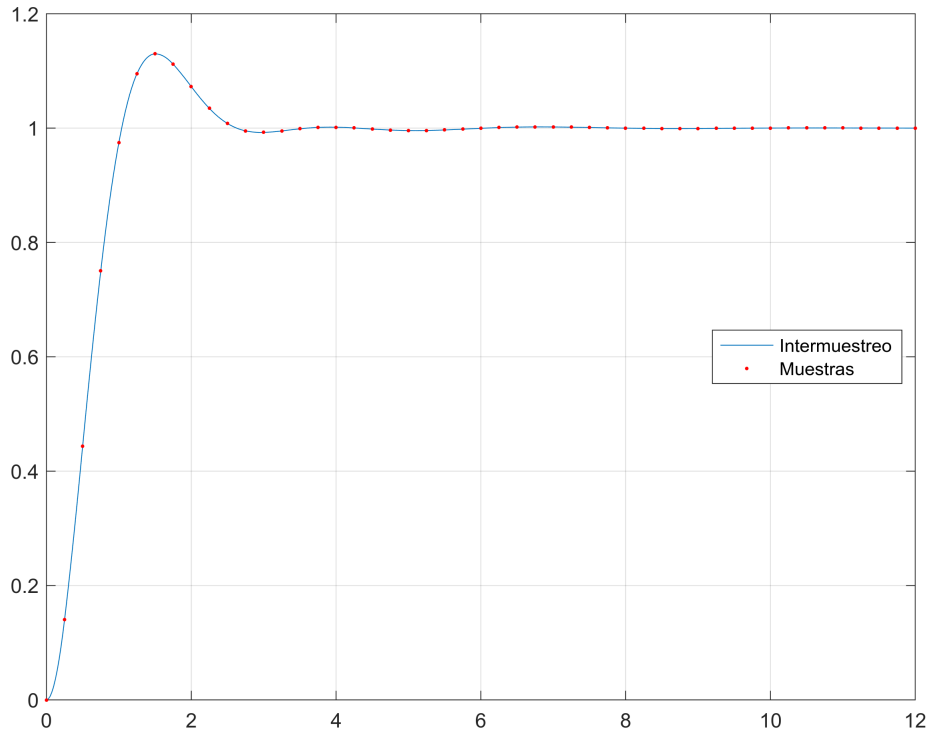
de modo que es una interconexión "lower Linear Fractional Transformation (LFT)" entre un sistema contínuo y uno discreto:



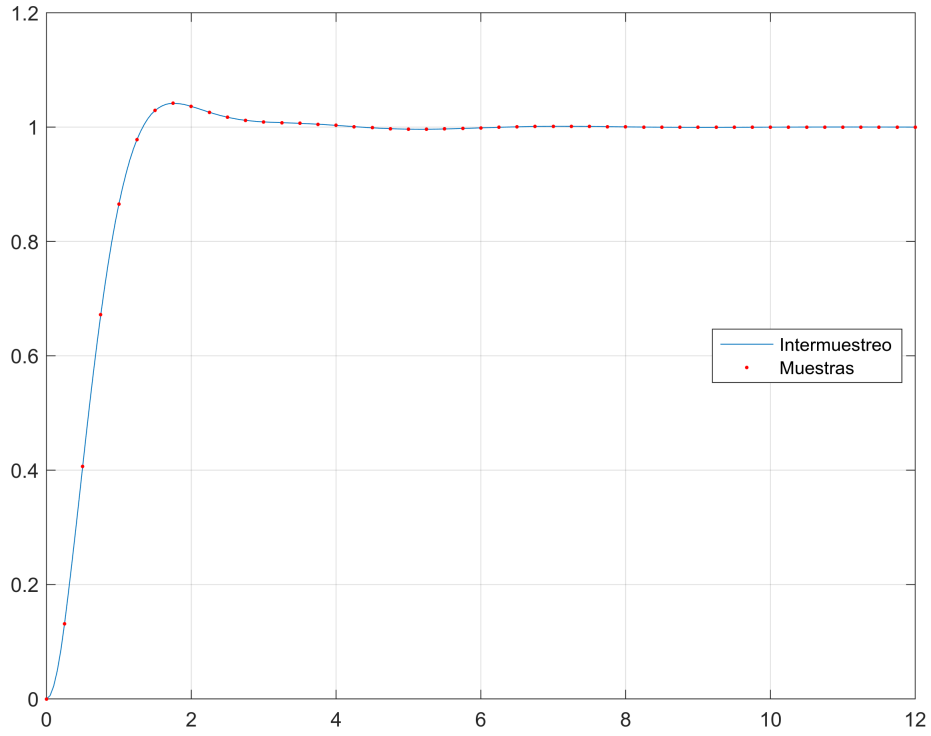
Siendo $w_1 = (r, d_u)$ y $z_1 = (y, u)$... aunque, como u es realmente un tren de escalones (es una señal discreta retenida), no necesitamos sacarlo con sdlsim porque será idéntica a la simulación discreta... por tanto, en lo que sigue dejaremos $z_1 \equiv y$.

```
PG=connect(s1,s2,sysc,{'r','du','u'},{'y','e'}); %no sacamos "U", porque es discreta, p
PGd=c2d(PG,Ts,'zoh');
Tsimc=0:0.025:12; %tiempos de "interpolación" continuos deseados
Tsimd=0:Ts:12; %tiemos de muestreo discretos
UUc=ones(length(Tsimc),1)*[1 0]; %entradas generalizadas continuas (interpoladas): no r
UUd=ones(length(Tsimd),1)*[1 0]; %entradas sólo en instantes muestreo para simulación d
```

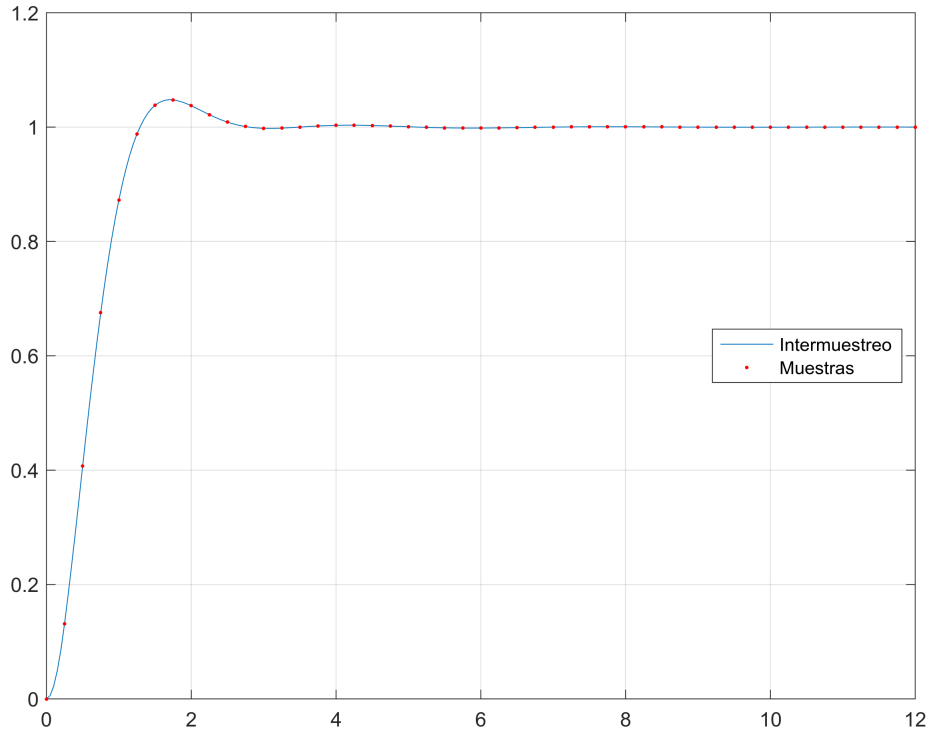
```
Y0=lsim(lft(PG,Kc),UUc,Tsimc); %simulación tiempo CONTINUO
Y1=sdlsim(PG,Kd1,UUc,Tsimc); %simulación HÍBRIDA (contínuo+discreto: sampled-data)
Y1d=lsim(lft(PGd,Kd1),UUd,Tsimd); %simulación tiempo DISCRETO
plot(Y1{1},Y1{2}), grid on, hold on
plot(Tsimd,Y1d,'.r'),hold off, legend("Intermuestreo","Muestras",Location="best")
```




```
Y2=sdlsim(PG,Kd2,UUd,Tsimd); %simulación HÍBRIDA (contínuo+discreto: sampled-data)
Y2d=lsim(lft(PGd,Kd2),UUd,Tsimd); %simulación tiempo DISCRETO
plot(Y2{1},Y2{2}), grid on, hold on
plot(Tsimd,Y2d,'.r'),hold off, legend("Intermuestreo","Muestras",Location="best")
```

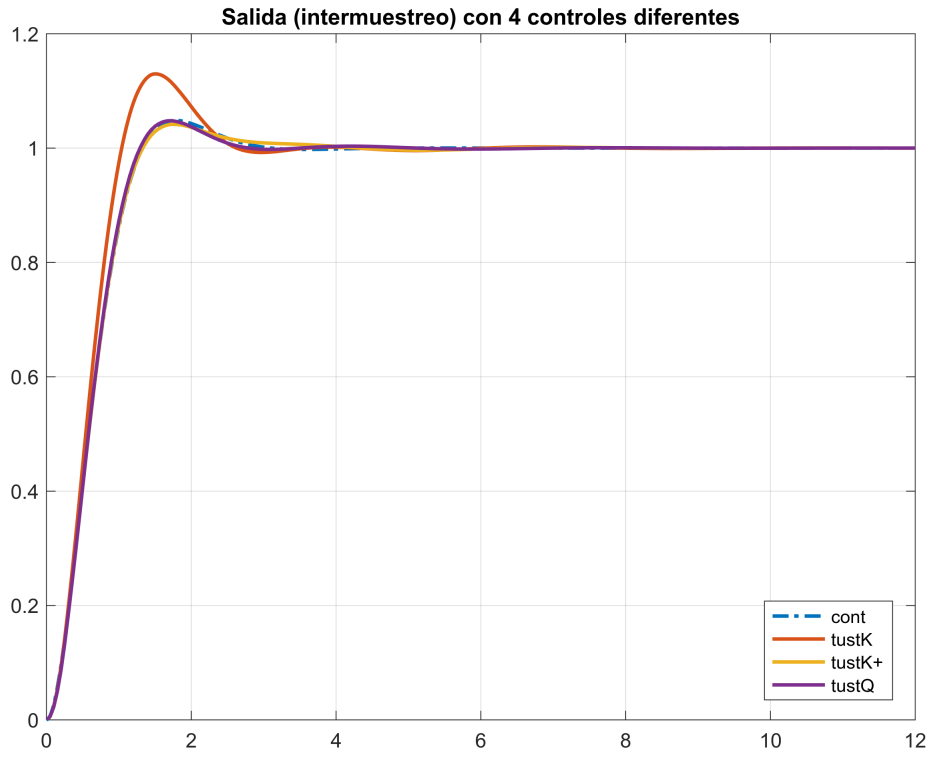



```
Y3=sdlsim(PG,Kd3,UUd,Tsimd); %simulación HÍBRIDA (contínuo+discreto: sampled-data)
Y3d=lsim(lft(PGd,Kd3),UUd,Tsimd); %simulación tiempo DISCRETO
plot(Y3{1},Y3{2}), grid on, hold on
plot(Tsimd,Y3d,'.r'),hold off, legend("Intermuestreo","Muestras",Location="best")
```



Todos juntos:

```
plot(Tsimc,Y0,'-.',LineWidth=2), grid on, hold on
plot(Y1{1},Y1{2},LineWidth=2),
plot(Y2{1},Y2{2},LineWidth=2), plot(Y3{1},Y3{2},LineWidth=2), hold off
legend('cont','tustK','tustK+','tustQ',Location="best"), title("Salida (intermuestreo)
```



Conclusiones

Basarse en el tiempo contínuo para diseñar reguladores discretos tiene validez a períodos "pequeños". En IMC, lo más recomendable si el período no es "tan pequeño" es la **opción 2**, calcular el regulador (IMC implícito) haciendo el feedback (realim.+) de $Q(s)$ y $G(s) \cdot (1 + \frac{T_s}{2} s)$, en vez de sólo $G(s)$, o bien la **opción 3**, directamente discretizar (Tustin) el parámetro Q y implementarlo bien explícitamente bien implíticamente cerrando $Q_d(z)$ y $G_d(z)$, siendo $G_d(z)$ la discretización (exacta) ZOH del proceso; resulta un regulador de orden mayor que la opción 2 pero tiene garantía de estabilidad para cualquier T_s, aunque no de prestaciones.