Control PID con filtro de ruido, caso estudio ejemplo proceso orden 2

Este código funcionó correctamente en Matlab **R2022b** (Linux)

Objetivo: Comprender las limitaciones del PID ideal $K_p + K_i \cdot 1/s + K_d \cdot s$ por "***no realizabilidad***". Diseñar un control PID con filtro de ruido que tenga una amplificación de ruido de medida a alta frecuencia sobre acción de control "razonable" ($\approx$ salto inicial ante escalón). Requerirá menor sobredimensionamiento de actuadores y tendrá más probabilidad de funcionar en la práctica (robustez) que otros diseños anteriores en este caso de estudio.

Presentaciones en vídeo:

http://personales.upv.es/asala/YT/V/ord2ejPIDF1.html

http://personales.upv.es/asala/YT/V/ord2ejPIDF2.html

Tabla de Contenidos

Motivación (no validez de diseños anteriores)

En otros vídeos se diseñaron PI, PD, PIDs por distintos métodos. Escojamos por ejemplo el PID por "cancelación total" para conseguir t_{est} (98%) $\leq$ 0.666667 s, $e_p = 0$, $\delta < 0.2$, que resultó (no es la única opción) en:

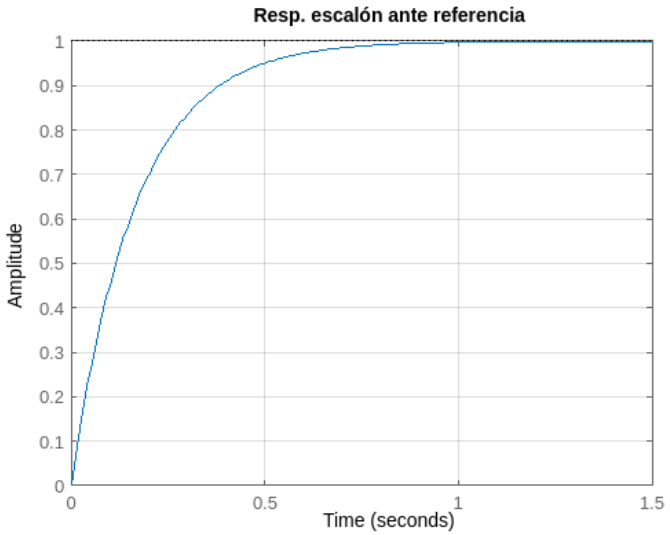
```
s=tf('s');
G=27/(s+1)/(s+9); %control systems Toolbox
K1=0.2222*(s+1)*(s+9)/s; %propuesta regulador cancelación
BC1=minreal(feedback(G*K1,1));
zpk(BC1)
```

```
ans =

   5.9994
  ---------
  (s+5.999)

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

```
step(BC1), grid on, title("Resp. escalón ante referencia")
```




```
u1sinfiltro_cst=feedback(K1,G); % K/(1+GK)
minreal(zpk(u1sinfiltro_cst),1e-6) %no realizable
```

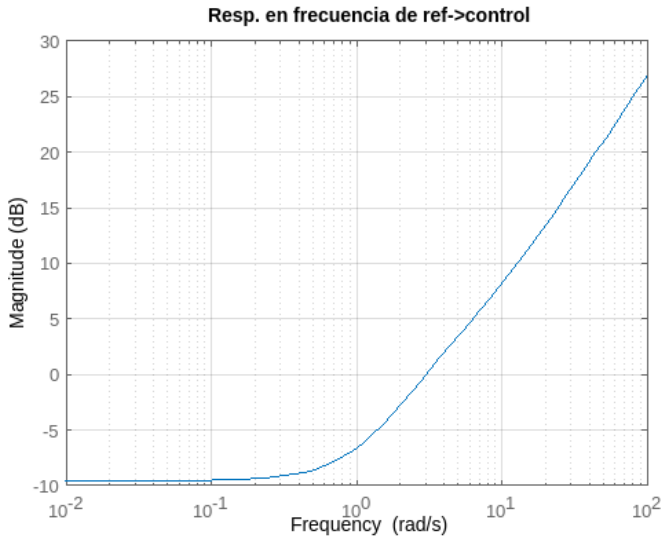
```
ans =

  0.2222 (s+9) (s+1)
  ------------------
      (s+5.999)

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

Como tiene 2 ceros y 1 polo, el comando "step" comentado abajo da error. No se puede simular con señales "finitas".

```
%step(u1sinfiltro_cst)
bodemag(u1sinfiltro_cst), grid on, title("Resp. en frecuencia de
ref->control")
```



En la práctica, el derivador puro amplificaría infinitamente el ruido de medida de alta frecuencia ($K_d j\omega$ tiende a infinito conforme sube la frecuencia).

Si la referencia cambia en "escalón", su derivada en el instante de cambio será "infinita" (un "impulso", $\delta(t)$).

Ningún diseño PD ni el PID con cancelación parcial de los vídeos anteriores se libran de este inconveniente (grave).

En general, ninguna FdT cuyo diagrama de amplitud de Bode tienda a infinito a alguna frecuencia podrá ser válido como solución a alguna FdT de un bucle cerrado de control entre señales físicas: o bien será no realizable (si tiende a infinito cuando $\omega \to \infty$) o bien será marginalmente inestable (polos en el eje imaginario) y, por tanto, el puro ruido se acumulará y hará tender la salida hacia una varianza infinita.

Un posible "apaño" (Workaround): filtrar con polo filtro no dominante

La acción derivada debe ser filtrada, por ejemplo con un filtro de cte de tiempo 1/10 de la cte de tiempo de bucle cerrado, para que no influya en la dinámica:

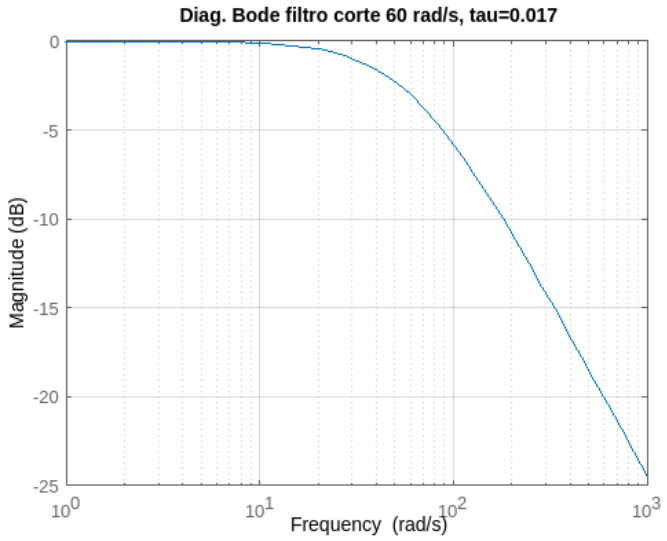
Consideramos un PID con derivada filtrada:

$$K_p + K_i \cdot \frac{1}{s} + K_d s \cdot \frac{1}{\tau_f s + 1}$$

```
sigma_filtro=6*10;
tau_filtro=1/sigma_filtro
```

```
tau_filtro = 0.0167
```

```
Filtro=1/(tau_filtro*s+1);
bodemag(Filtro), grid on, title("Diag. Bode filtro corte 60 rad/s,
tau=0.017")
```




```
K1_practico=2.222+2/s+0.22*s*Filtro;  %PID con filtro de derivada
```

```
zpk(K1_practico)
```

```
ans =

  15.422 (s+7.774) (s+1.001)
  --------------------------
           s (s+60)

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

Nota: el filtro en dericada quita el cero de +9, pasa a ser +7. El otro apenas se mueve de +1... Podríamos haber filtrado "todo" con K1_practico=K1*Filtro, para mantener la idea de la "cancelación"... eso es lo que se hará en el segundo diseño en este material (vídeo 2).

Aquí, por simplicidad, se filtra únicamente lo que "sí o sí" debe ser filtrado para realizabilidad: la derivada.

Si simulamos la salida, tenemos:

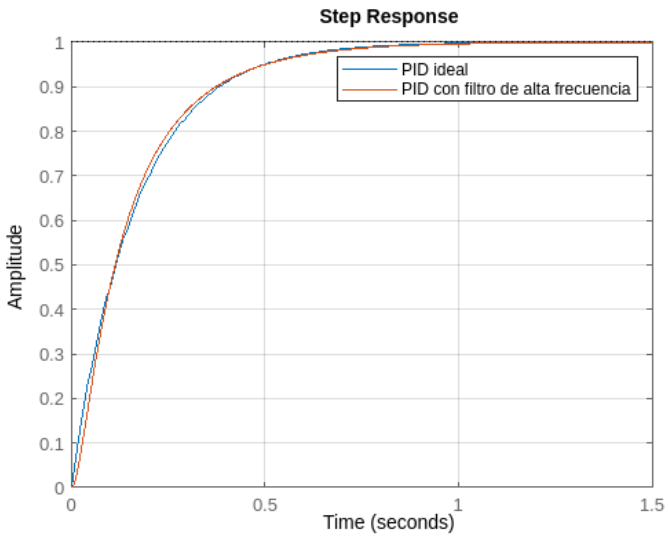
```
BCy_practico=feedback(G*K1_practico,1);minreal(zpk(BCy_practico),1e-3)
```

```
ans =

        416.39 (s+7.774)
  -----------------------------
  (s+51.72) (s+12.12) (s+5.164)

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

```
step(feedback(G*K1,1),BCy_practico), grid on
legend("PID ideal","PID con filtro de alta frecuencia")
```




```
BC1u=feedback(K1_practico,G); zpk(BC1u)
```

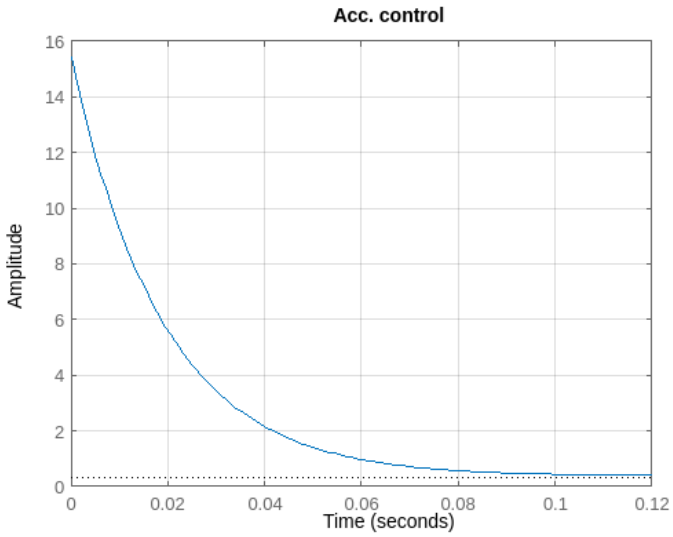
```
ans =

  15.422 (s+9) (s+7.774) (s+1.001) (s+1)
  --------------------------------------
```

```
   (s+51.72) (s+12.12) (s+5.164) (s+1.001)

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

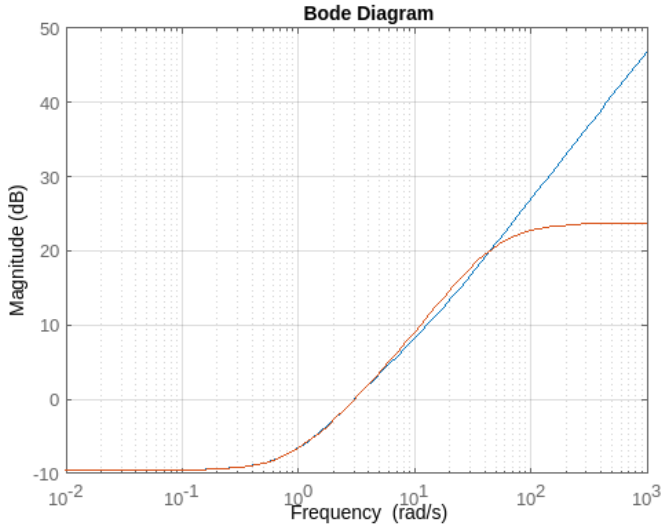
```
step(BC1u), grid on, title("Acc. control")
```




```
20*log10(15)
```

```
ans = 23.5218
```

```
bodemag(u1sinfiltro_cst,feedback(K1_practico,G)), grid on
```



Desde un punto de vista "práctico" nos pasamos "tres pueblos" en intensidad de la acción derivada... el "pico" de u es 15, comparado con 0.333 de valor final: es un "sobredimensionamiento de 45" en la fase inicial... Y era "infinito" sin el filtro, ufff!

Esto es debido a ciertas posibles causas:

- En muchos casos prácticos se deriva la "salida", no el "error", para evitar "derivative kick" ("patada derivativa") inicial ante cambios escalón de referencia. Cambian las ecuaciones de bucle cerrado, no son objetivo de este material introductorio.
- La "cancelación" no es la solución única, puede haber otras opciones menos "enérgicas".
- El "filtrado" puede hacerse más intenso (polo más lento) sin distorsionar en exceso la respuesta.
- Acelerar muy rápido la dinámica del proceso requiere acciones muy enérgicas... igual nos hemos pasado en pedir t_{est} deseado demasiado pequeño.

Vamos a abordar la tercera de las opciones.

***Nota:** en implementaciones en "tiempo discreto", a un período de muestreo que no sea demasiado pequeño, la diferencia finita $\frac{dy}{dt} \approx \frac{y_k - y_{k-1}}{T}$ tiene un cierto efecto de filtrado, porque el muestreo elimina las altas frecuencias (más allá de π/T rad/s, frecuencia de Nyquist), por lo que diseños "sin filtro" podrían funcionar... Esto queda fuera de los objetivos de este material, pero si eres un lector avanzado y sabes de lo que estoy hablando, quizás puedes pararte a pensar un poco en ello.

Diseño directo PID con filtro de ruido (con "truco" de la cancelación)

Un PID "completo", con filtro de ruido puede expresarse como:

$$K_p + K_i \cdot \frac{1}{s} + K_d s \cdot \frac{1}{\tau_f s + 1} = \frac{K_p s(\tau_f s + 1) + K_i(\tau_f s + 1) + K_d s^2}{s(\tau_f s + 1)}$$

$$K_p + K_i \cdot \frac{1}{s} + K_d s \cdot \frac{1}{\tau_f s + 1} = \widetilde{K}_c \frac{(s+\alpha)(s+\beta)}{s(\tau_f s + 1)} = K_c \frac{(s+\alpha)(s+\beta)}{s(s+p)}$$

Usaremos la interpretación "factorizada" de la derecha para calcular nuestro controlador.

Cancelaremos 2 polos del proceso para que GK sea de orden 2:

```
syms s
G=27/(s+1)/(s+9);
syms K_c p real
K=K_c*(s+1)*(s+9)/s/(s+p);
L=G*K
```

```
L =
```

$$\frac{27\,K_c}{s\,(p+s)}$$

```
E(s)=simplifyFraction(1/(1+L))
```

```
E(s) =
```

$$\frac{s\,(p+s)}{s^2+p\,s+27\,K_c}$$

Tenemos controlabilidad total, para especificaciones dinámicas.

Error de posición ante referencia escalón unitario:

```
E(0) %es cero, porque hay acción integral
```

```
ans = 0
```

Esfuerzo de control:

```
KS=simplifyFraction(K/(1+G*K))
```

```
KS =
```

$$\frac{K_c\,(s+1)\,(s+9)}{s^2+p\,s+27\,K_c}$$

Amplificación del control a alta frecuencia (valor inicial)

```
limit(KS,s,inf)
```

```
ans = K_c
```

Como tenemos controlabilidad total, y la parte real deseada era -6 de los polos, para tiempo de establecimiento, haremos:

```
DenDeseado=(s+6)^2;
[Ne,De]=numden(E);
sol=solve(De==DenDeseado,{K_c,p})
```

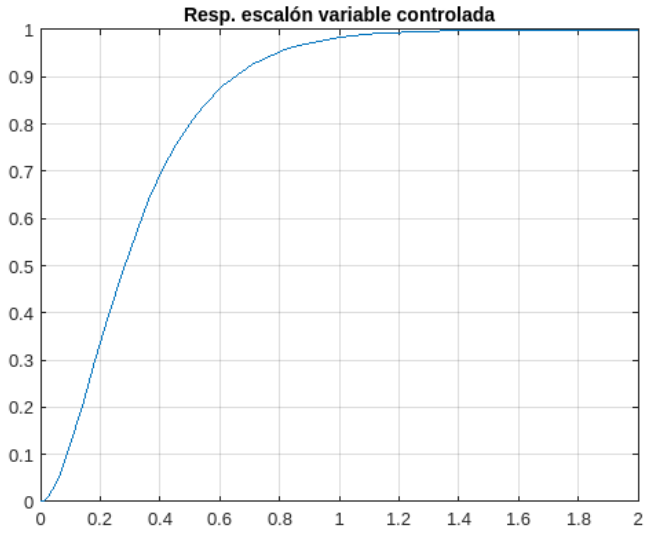
```
sol = struct with fields:
    K_c: 4/3
      p: 12
```

```
T=simplify(subs(G*K/(1+G*K),sol))
```

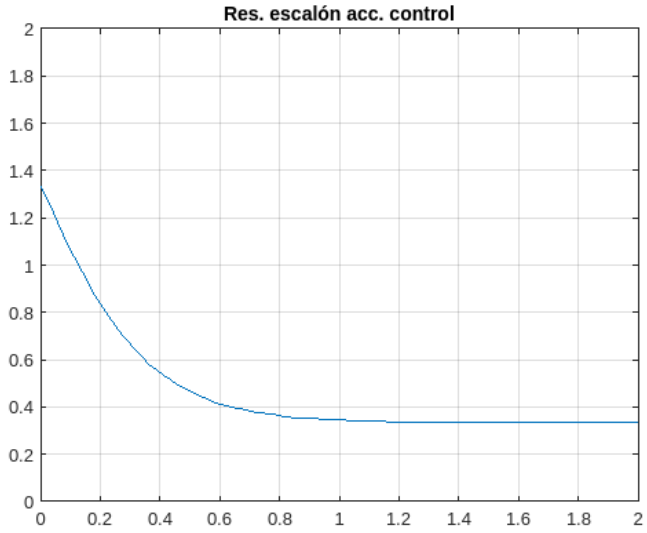
```
T =
```

$$\frac{36}{(s+6)^2}$$

```
fplot(ilaplace(T/s),[0 2]), grid on, title("Resp. escalón variable controlada")
```




```
KS=simplify(subs(K/(1+G*K),sol));
fplot(ilaplace(KS/s),[0 2]), grid on, ylim([0 2]), title("Res.
escalón acc. control")
```



Este diseño, con un "sobredimensionamiento" de 4 (pico transitorio 1.33, valor final 0.33) es muy razonable. Acelerar la respuesta de 4s de t_{est} en bucle abierto a 0.7 segundos en bucle cerrado requiere, inevitablemente, un "sobreesfuerzo" (boost) inicial... Es el precio a pagar por mover más rápido al sistema. Que ese sobredimensionamiento sea de 15 no es muy aceptable, de 4 sí lo podría ser y estaría el diseño listo para ser probado experimentalmente.

Nota:** el diseño todavía ***no está 100% validado para implementación final:

- no hemos considerado la respuesta ante "perturbaciones". La cancelación de polos lentos $(s+1)$ puede darnos alguna sorpresa en este aspecto, pero eso está fuera de los objetivos de este material introductorio.
- No hemos implementado "antiwindup" para evitar inestabilidad interna cuando los actuadores saturan.