Asignación de polos con control **Proporcional-Derivado** o **Proporcional-Integral**, sistema **segundo** orden, symbolic toolbox

*Este código funcionó sin errores en Matlab `R2022b` (Linux)

Objetivo: calcular control **PD** o **PI** para cumplir ciertas especificaciones de error ante cambios de referencia en bucle cerrado en un proceso de **SEGUNDO** orden sin integradores (**tipo 0**).

Presentaciones en vídeo:

http://personales.upv.es/asala/YT/V/ord2ejPD.html

http://personales.upv.es/asala/YT/V/ord2ejPI.html

Tabla de Contenidos


```
syms s
G(s)=27/(s+1)/(s+9);  %symbolic toolbox
Gs=tf(27,[1 10 9]); %control Toolbox
```

Control PROPORCIONAL, análisis bucle cerrado

```
syms K_p real
K=K_p;
E(s)=1/(1+G(s)*K) %FdT error ante cambios de referencia
```

```
E(s) =
```

$$\frac{1}{\frac{27\,K_p}{(s+1)\,(s+9)}+1}$$

```
E(s)=simplifyFraction(E(s)) %en "bonito"
```

E(s) =

$$\frac{(s+1)\,(s+9)}{s^2+10\,s+27\,K_p+9}$$

Análisis de especificaciones estáticas (error de posición)

```
e_p=simplify(E(0)) %valor final error ante ref=1
```

e_p =

$$\frac{1}{3\,K_p+1}$$

Haciendo K_p grande, el error puede ser todo lo pequeño que se desee (pero finito, >0).

Análisis de especificaciones dinámicas (polos de bucle cerrado)

```
[Ne,De]=numden(E);
De
```

De(s) = $s^2+10\,s+27\,K_p+9$

```
solve(De==0,s) %polos de bucle cerrado
```

ans =

$$\begin{pmatrix} -\sqrt{16-27\,K_p}-5 \\ \sqrt{16-27\,K_p}-5 \end{pmatrix}$$

Las raíces serán reales si $K_p \leq \frac{16}{27}$, una a la izquierda y otra a la derecha de -5.

SI $K_p > \frac{16}{27}$ entonces tendrá dos raíces complejas, $-5 \pm \omega_p \cdot j$ de parte real -5.

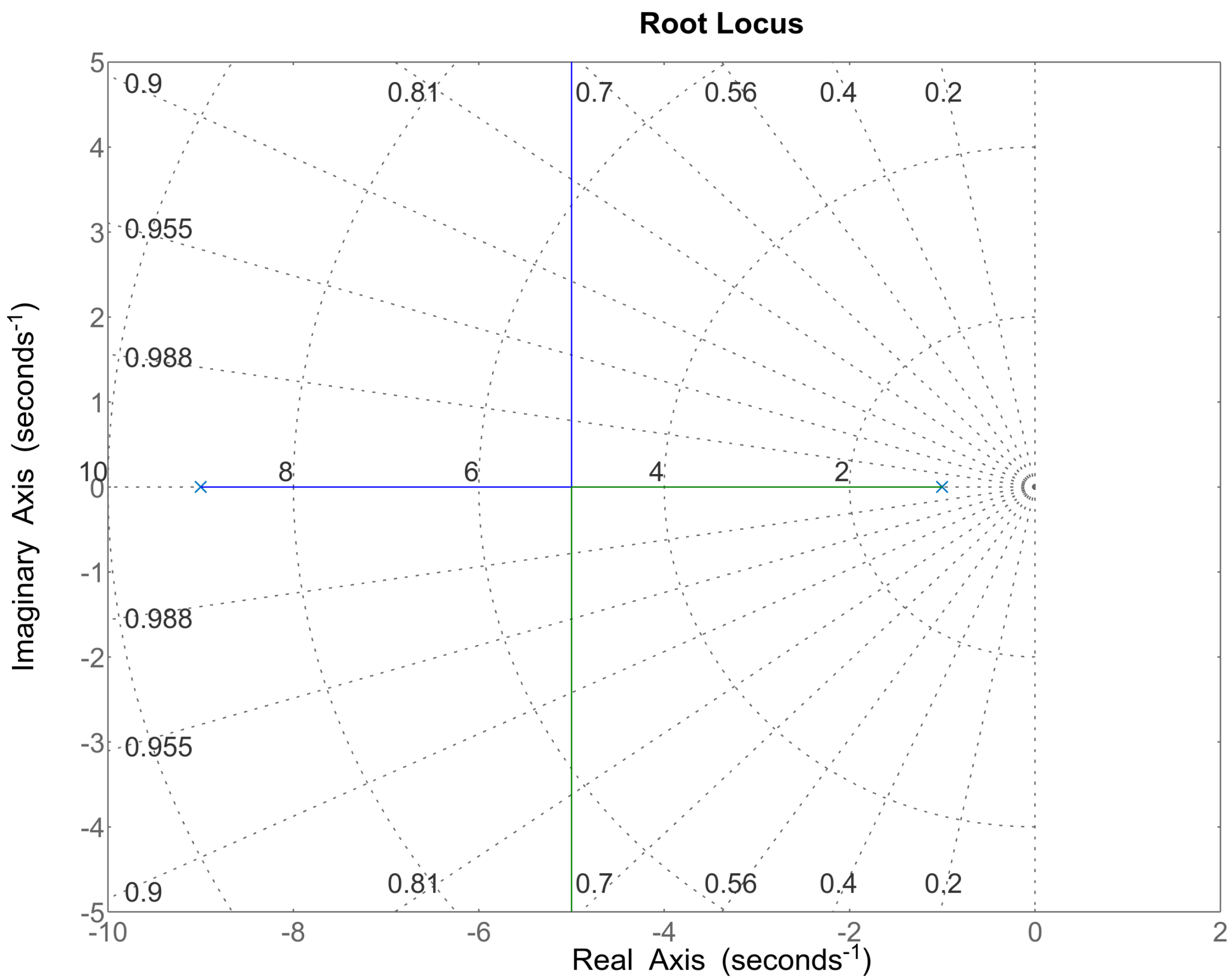
```
Klim_reales=16/27
```

Klim_reales = 0.5926

Por tanto, la parte real "más a la izquierda" del plano complejo que se podrá conseguir es -5, con tiempo de establecimiento aproximado de $4/5 = 0.8$ segundos.

En cuanto a sobreoscilación, el factor de amortiguamento se obtendría con $\omega_n = \sqrt{27K_p + 9}$, $2\xi\omega_n = 10$, o sea $\xi = \dfrac{5}{\sqrt{27K_p + 9}}$.

```
rlocus(Gs), grid on
```




```
Kp_test=16/27*[.5 0.9996 2 4 15];
```

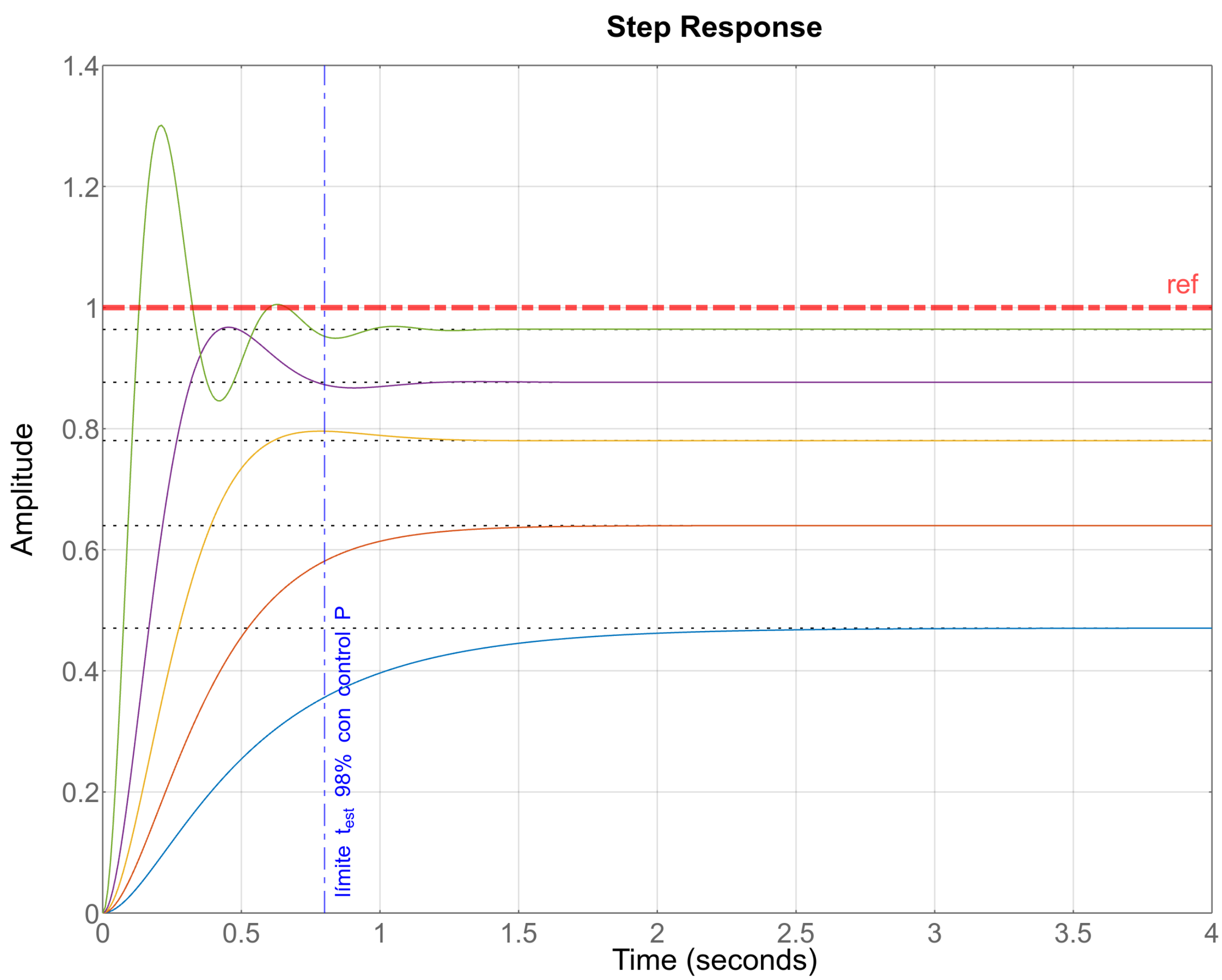
Si $t_{e,deseado} < 0.8$ entonces NO podrá hacerse con regulador proporcional.

Si el error de posición es muy pequeño, se necesita K_p elevada y, seguramente, la sobreoscilación sea inaceptable.

Antes de entrar en detalle, vamos a simular varias constantes

```
BC1=feedback(Gs*Kp_test(1),1);
BC2=feedback(Gs*Kp_test(2),1);
```

```
BC3=feedback(Gs*Kp_test(3),1);
BC4=feedback(Gs*Kp_test(4),1);
BC5=feedback(Gs*Kp_test(5),1);
step(BC1,BC2,BC3,BC4,BC5), grid on,
yline(1,'-.r',Label="ref",LineWidth=2)
xline(0.8,'-.b',Label="límite t_{est} 98% con control
P",FontSize=8,LabelVerticalAlignment="bottom")
```



Ejemplo de especificaciones NO factibles con regulador proporcional

Por ejemplo, t_{est} (98%) $\leq$ 1 s, $e_p \leq 0.04$, $\delta < 0.2$ se conseguiría con $\sigma > 4/1$,

```
syms K_p real
sol_sigma=solve(-5+sqrt(16-27*K_p)<=-4/1,ReturnConditions=true);
vpa(sol_sigma.conditions,4)
```

ans = $0.5556 \leq x \wedge x \leq 0.5926$

```
ep_deseado=0.04;
sol_e_p=solve(e_p<=ep_deseado, K_p>=0, ReturnConditions=true); %para obtener todas las soluciones que cumplen
vpa(sol_e_p.conditions,4)
```

ans = $8.0 \leq x$

```
delta_deseado=0.2;
tmp=log(delta_deseado)^2;
xi_deseado=sqrt(tmp/(tmp+pi^2))
```

```
xi_deseado = 0.4559
```

```
xi=5/sqrt(27*K_p+9);
sol_delta=solve(xi>=xi_deseado,ReturnConditions=true);
vpa(sol_delta.conditions,4)
```

ans = $x \leq 4.121 \wedge -0.3333 < x$

No podemos encontrar un K_p que verifique todas las desigualdades que se deducen de cada especificación. Necesitaríamos control más complicado (PI que daría error cero, o PD que daría error no cero pero permitiría subir K_p al amortiguar mejor con K_d). El detalle NO es objeto de este material por brevedad.

Solución con control PD [sin cancelación]

Empezamos "de cero", con el error en bucle cerrado con PD, en simbólico.

t_{est} (98%) $\leq$ 1 s, $e_p \leq 0.04$, $\delta < 0.2$

```
syms K_d
K=K_p+K_d*s
```

K = $K_p + K_d\, s$

```
E(s)=1/(1+G(s)*K) %error ante cambios de referencia
```

E(s) =

$$\frac{1}{\frac{27\ (K_p + K_d\, s)}{(s+1)\ (s+9)} + 1}$$

```
E(s)=collect(simplifyFraction(E(s)),s)
```

E(s) =

$$\frac{s^2 + 10\, s + 9}{s^2 + (27\, K_d + 10)\, s + 27\, K_p + 9}$$

```
[Ne,De]=numden(E)
```

Ne(s) = $s^2 + 10\,s + 9$

De(s) = $27\,K_p + 10\,s + 27\,K_d\,s + s^2 + 9$

Análisis de especificaciones estáticas (error de posición)

```
e_p=simplify(E(0)) %valor final error
```

e_p =

$$\frac{1}{3\,K_p + 1}$$

Haciendo K_p grande, el error puede ser todo lo pequeño que se desee (pero finito, >0).

Análisis de especificaciones dinámicas (polos de bucle cerrado)

Tenemos "controlabilidad total", dado que podemos despejar K_d y K_p a partir de cualquier valor deseado de los polos.

Por tanto, para solucionar el problema pedido, necesitamos primero saber una cota de K_p, que coincidirá con la obtenida para el caso "proporcional" porque la acción "derivada" no tiene efecto en equilibrio (velocidad cero); salía $K_p \geq 8$.

Por tanto, el término independiente del denominador será, como mínimo

```
27*8+9
```

ans = 225

Ese término será el producto de las raíces (si reales) o el módulo (si complejas).

Si queremos polos reales, para asegurar que no hay sobreoscilación, entonces:

```
sqrt(225)
```

ans = 15

esto está dentro de la zona permitida por tiempo de establecimiento (parte real < -4), con lo que vamos a asignar los polos ahí:

```
Den_deseado=expand((s+15)^2)
```

Den_deseado = $s^2 + 30\,s + 225$

```
sol_pd1=solve(De==Den_deseado,{K_p,K_d})
```

```
sol_pd1 = struct with fields:
    K_p: 8
    K_d: 20/27
```

```
BC_pd1=simplify(subs(G*K/(1+G*K),sol_pd1)); %FdT referencia a salida en bucle cerrado, para simular.
[Nubc,Debc]=numden(BC_pd1);vpa(solve(Debc),4)
```

ans =

$$\begin{pmatrix} -15.0 \\ -15.0 \end{pmatrix}$$

Si nos conformamos con polos complejos, podemos pedir, fijando $\omega_n = \sqrt{225} = 15$ porque no puede ser menor (por el error de posición), el amortiguamiento deseado:

```
Den_deseado=vpa(s^2+2*xi_deseado*15*s+15^2,5)
```

Den_deseado = $s^2 + 13.678\, s + 225.0$

Tenemos que comprobar que cumple la parte real adecuada para el tiempo de establecimiento (< -4):

```
vpa(solve(Den_deseado),4) %OK también parte real.
```

ans =

$$\begin{pmatrix} -6.839 - 13.35\,\mathrm{i} \\ -6.839 + 13.35\,\mathrm{i} \end{pmatrix}$$

```
sol_pd2=solve(De==Den_deseado,{K_p,K_d})
```

```
sol_pd2 = struct with fields:
    K_p: 8.0
    K_d: 0.13624053048422663576073116726345
```

```
vpa(sol_pd2.K_d,4)
```

ans = 0.1362

sale menor K_d.

Simulemos, pues el resultado:

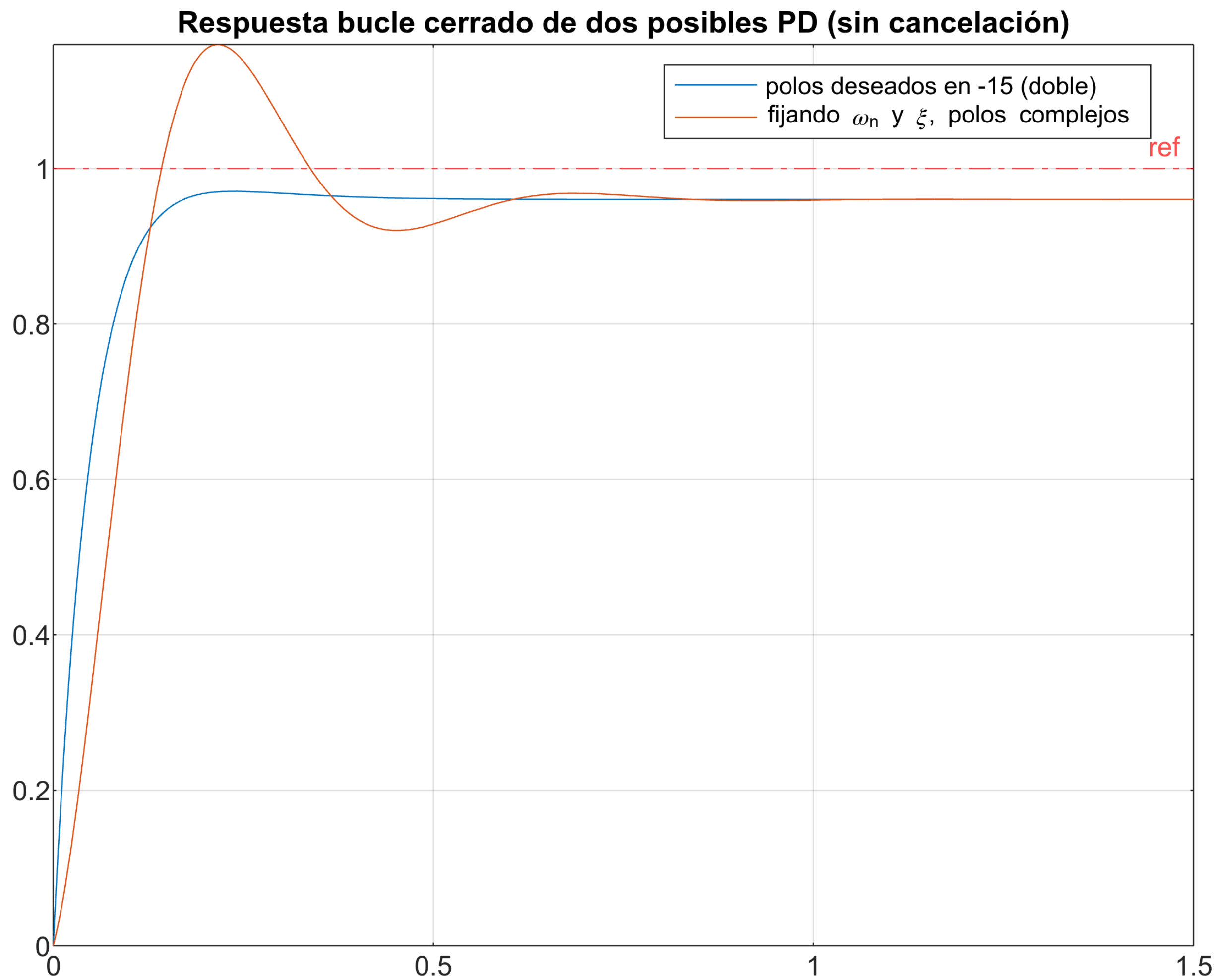
```
BC_pd2=simplify(subs(G*K/(1+G*K),sol_pd2));
[Nubc,Debc]=numden(BC_pd2);vpa(solve(Debc),4)
```

ans =

$$\begin{pmatrix} -6.839 - 13.35\,\mathrm{i} \\ -6.839 + 13.35\,\mathrm{i} \end{pmatrix}$$

```
fplot([ilaplace(BC_pd1/s);ilaplace(BC_pd2/s)],[0 1.5]), grid on,
yline(1,'-.r',Label="ref")
```

```
legend("polos deseados en -15 (doble)","fijando \omega_n y \xi, polos
complejos",Location="best")
title("Respuesta bucle cerrado de dos posibles PD (sin cancelación)")
```



Solución con control PD [con cancelación]

Empezamos nuevamente "de cero", con el error en bucle cerrado con PD, en simbólico.

t_{est} (98%) $\leq$ 1 s, $e_p \leq 0.04$, $\delta < 0.2$

```
syms K_d
K=K_d*(s+9); %cancelamos el polo de G más alejado del origen con este
sistema 2o orden
E(s)=1/(1+G(s)*K) %error ante cambios de referencia
```

```
E(s) =
```

$$\frac{1}{\frac{27\,K_d}{s+1}+1}$$

```
E(s)=collect(simplifyFraction(E(s)),s)
```

```
E(s) =
```

$$\frac{s+1}{s+27\,K_d+1}$$

```
[Ne,De]=numden(E)
```

Ne(s) = $s+1$

De(s) = $27\,K_d+s+1$

Análisis de especificaciones estáticas (error de posición)

```
e_p=E(0) %valor final error
```

```
e_p =
```

$$\frac{1}{27\,K_d+1}$$

Haciendo K_d grande, el error puede ser todo lo pequeño que se desee (pero finito, >0).

Análisis de especificaciones dinámicas (polos de bucle cerrado)

Tenemos "controlabilidad total", dado que podemos despejar K_d a partir de cualquier valor deseado de los polos (sólo uno).

Por tanto, para solucionar el problema pedido, necesitamos primero saber una cota de K_d.

Por tanto, el término independiente del denominador será, como mínimo

```
assume(K_d>0) %negativo es inestable.
solve(e_p<=ep_deseado,ReturnConditions=true)
```

```
ans = struct with fields:
           K_d: x
    parameters: x
    conditions: 8/9 <= x
```

En cuanto a tiempo de establecimiento, tendríamos $27K_d+1>4$, o sea $K_d>1/9$.

Lo más restrictivo es $K_d=8/9$, vamos a verlo.

Simulemos, pues el resultado:

```
BC_pd3=simplify(subs(G*K/(1+G*K),K_d,8/9))
```

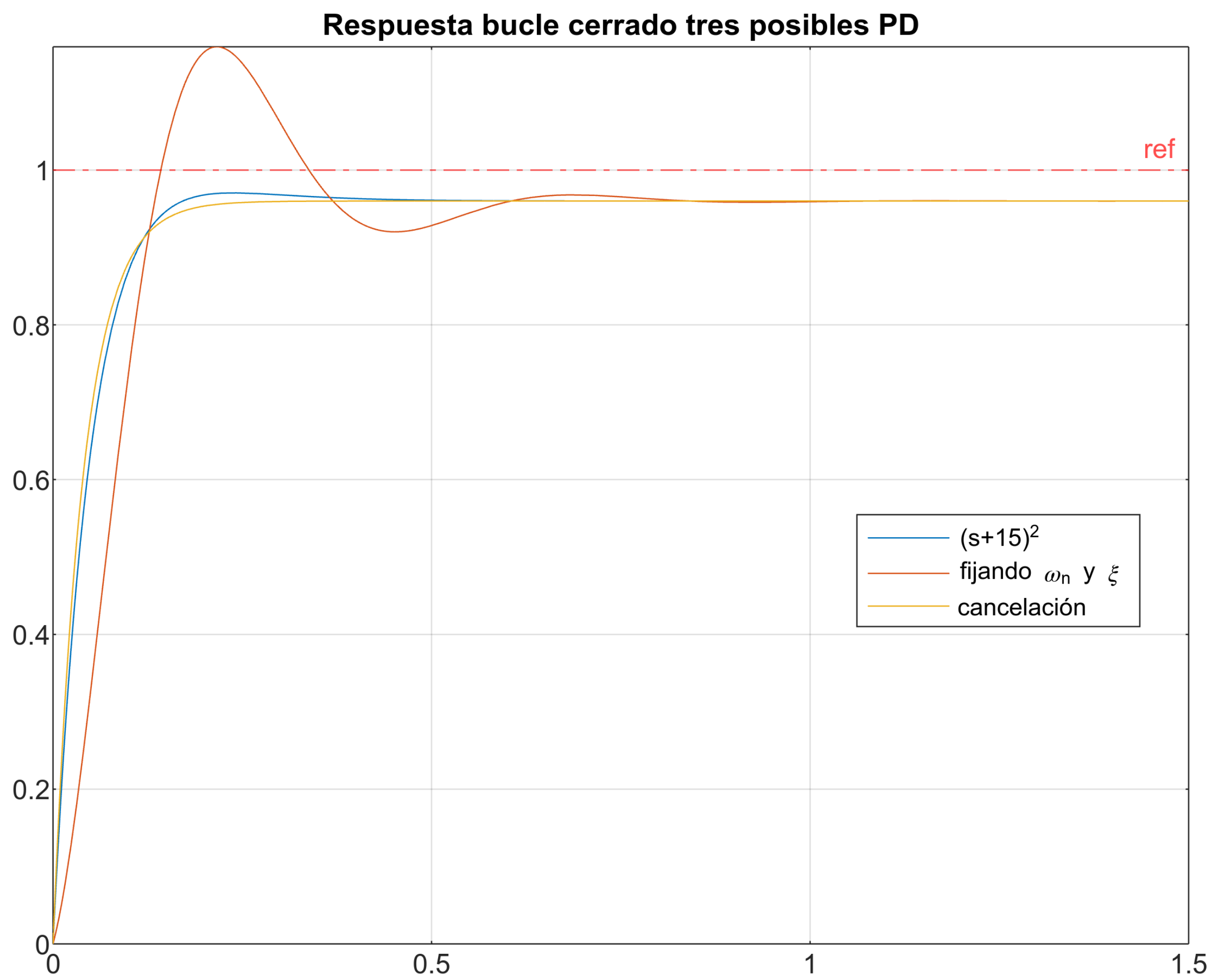
```
BC_pd3(s) =
```

$$\frac{24}{s+25}$$

```
[Nubc,Debc]=numden(BC_pd3);vpa(solve(Debc),4)
```

ans = -25.0

```
fplot([ilaplace(BC_pd1/s);ilaplace(BC_pd2/s);ilaplace(BC_pd3/s)],[0 1.5])
grid on, yline(1,'-.r',Label="ref"), legend("(s+15)^2","fijando \omega_n y
\xi ","cancelación",Location="best")
title("Respuesta bucle cerrado tres posibles PD")
```



Discusión sobre validez práctica del resultado (PD)

Los reguladores "PD" aquí diseñados NO son realizables, esto es, $K_p + K_d s$ tiene un "derivador puro", que amplifica "infinitamente" la alta frecuencia. De hecho, ante escalón de referencia, la derivada del error es "infinita", un "impulso".

Los reguladores "prácticos" deben tener, entre otras cosas, un filtro $K_p + K_d s \cdot \dfrac{1}{\tau_f \cdot s + 1}$, con un τ_f lo suficientemente "pequeño" para no afectar a la dinámica dominante de bucle cerrado ($1/\tau_f >> \sigma_{deseada}$) , pero lo suficientemente "grande" para evitar una amplificación excesiva del ruido de alta frecuencia. Los detalles NO son objetivo de este material introductorio, pero deben ser abordados antes de una implementación real.

Solución con control PI [con cancelación]

Empezamos otra vez "de cero", con el error en bucle cerrado con PI.

t_{est} (98%) $\leq$ 1 s, $e_p \leq 0.04$, $\delta < 0.2$

El error será "cero", un problema menos. Estamos obligados (si queremos usar fórmulas de 2o grado, usual en asignaturas introductorias) a usar cancelación para no tener orden 3.

```
syms K_i
K=K_p*(s+1)/s; %cancelamos el polo de G más cercano al origen en PI.
E(s)=1/(1+G(s)*K) %error ante cambios de referencia
```

E(s) =

$$\frac{1}{\frac{27\,K_p}{s\,(s+9)}+1}$$

```
E(s)=collect(simplifyFraction(E(s)),s)
```

E(s) =

$$\frac{s^2+9\,s}{s^2+9\,s+27\,K_p}$$

```
[Ne,De]=numden(E)
```

Ne(s) = $s^2+9\,s$

De(s) = $s^2+9\,s+27\,K_p$

Análisis de especificaciones estáticas (error de posición)

```
e_p=E(0) %valor final error
```

e_p = 0

Análisis de especificaciones dinámicas (polos de bucle cerrado)

```
polos_pi=solve(De)
```

```
polos_pi =
```

$$\begin{pmatrix} -\frac{3\sqrt{3}\,\sqrt{3-4\,K_p}}{2}-\frac{9}{2} \\ \frac{3\sqrt{3}\,\sqrt{3-4\,K_p}}{2}-\frac{9}{2} \end{pmatrix}$$

Las raíces estarán izquierda/derecha de -4.5 o complejas. El tiempo de establecimiento con control PI como hemos planteado estará limitado a 4/4.5... por suerte, nos vale.

```
sol_pi=solve(polos_pi(2)<-4,ReturnConditions=true)
```

```
sol_pi = struct with fields:
           K_p: x
    parameters: x
    conditions: 20/27 < x & x <= 3/4
```

Lo más restrictivo es $K_p = 20/27$, vamos a verlo

Simulemos, pues el resultado:

```
BC_pi=simplify(subs(G*K/(1+G*K),K_p,20/27))
```

```
BC_pi(s) =
```

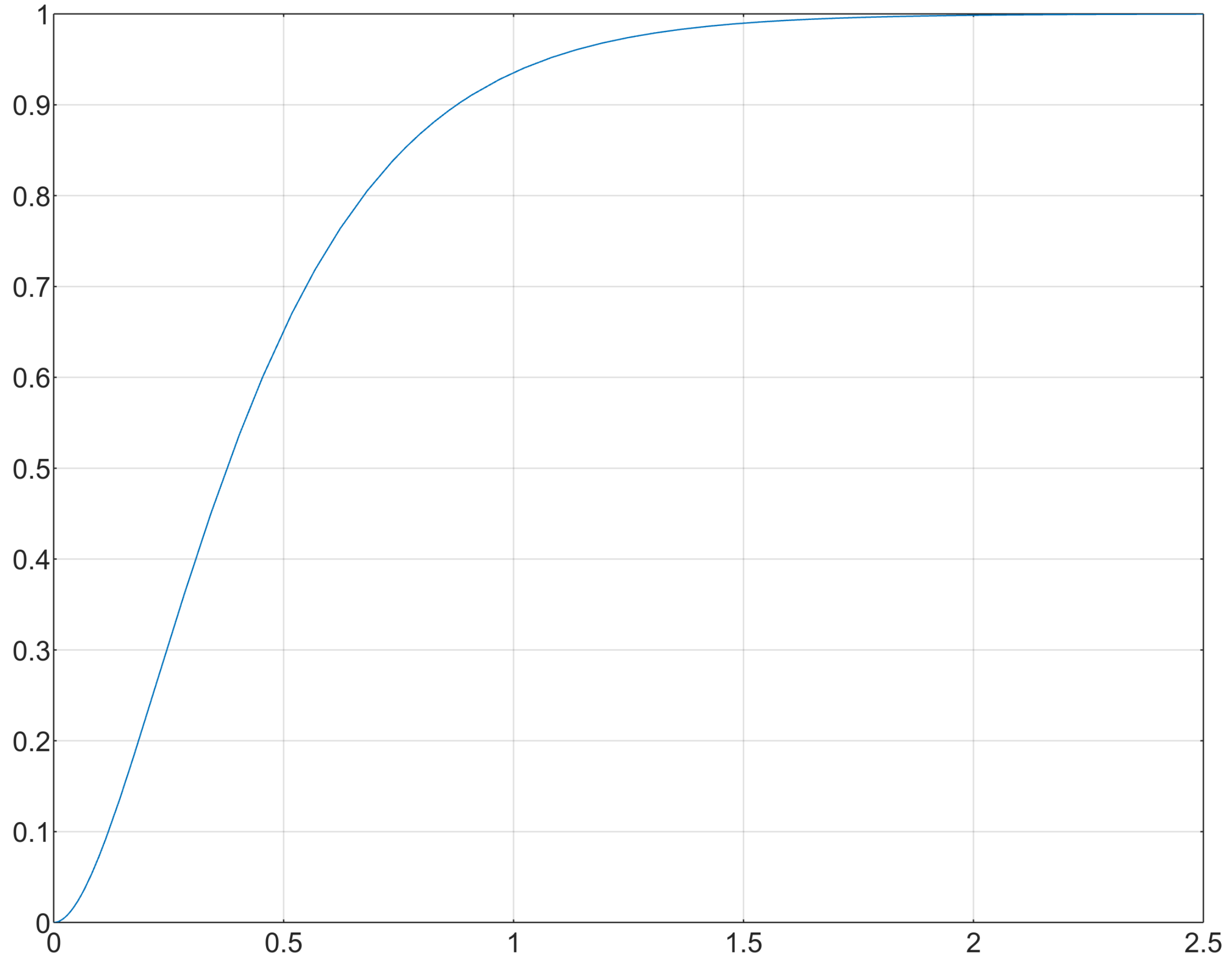
$$\frac{20}{s^2+9\,s+20}$$

```
[Nubc,Debc]=numden(BC_pi);vpa(solve(Debc),4)
```

```
ans =
```

$$\begin{pmatrix} -5.0 \\ -4.0 \end{pmatrix}$$

```
fplot(ilaplace(BC_pi/s),[0 2.5]), grid on
```



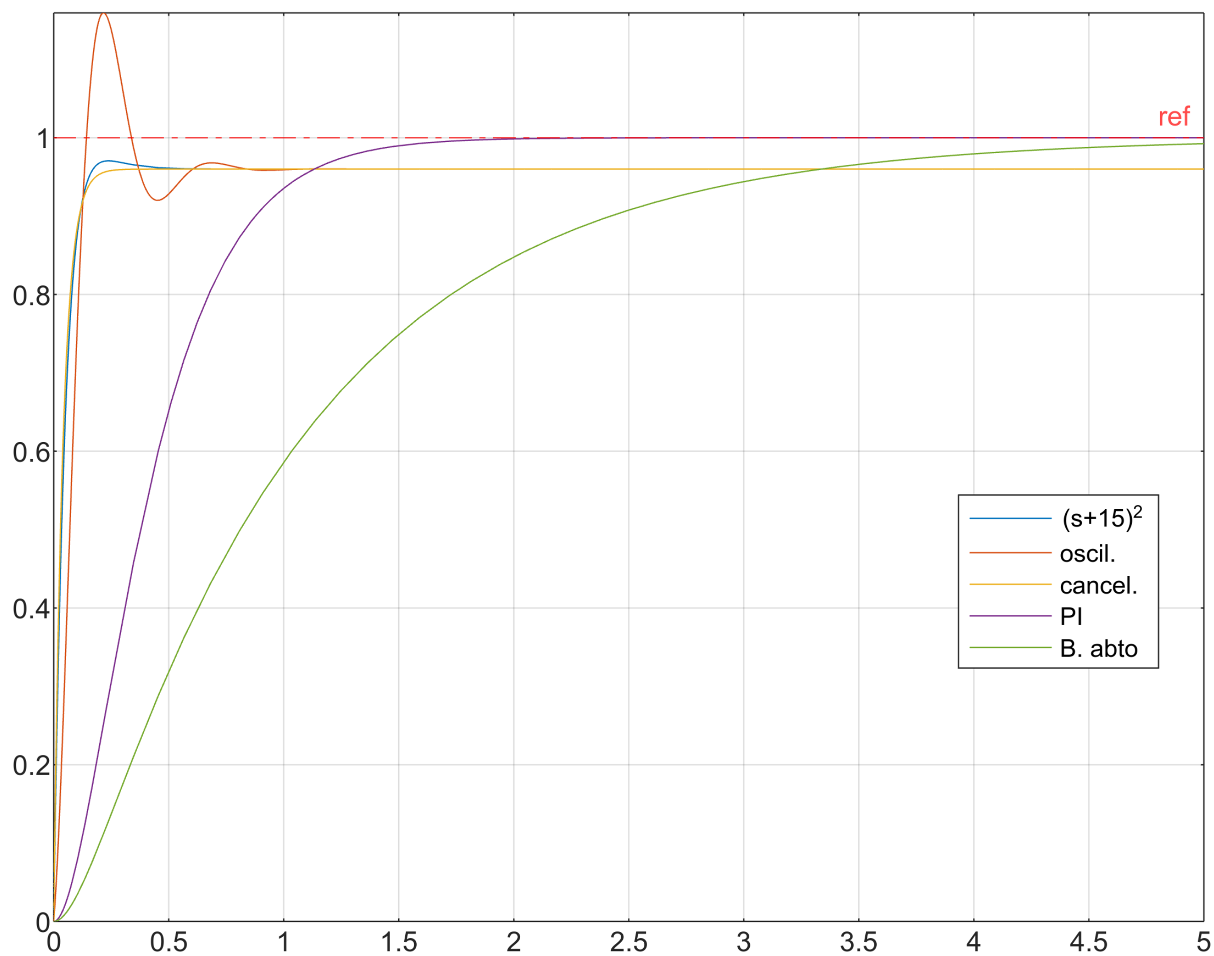
Discusión sobre validez práctica del resultado PI

Los reguladores "PI" son (marginalmente) inestables si se desconecta el bucle debido al término $1/s$... dicha desconexión suele producirse por saturación o por paso a "manual". En esos casos, hay variables internas del controlador que irían a infinito en forma de "rampa". El fenómeno se llama "windup" y es necesario, en la mayoría de casos, modificar el código/ecuaciones de la acción integral antes de una aplicación práctica realista.

Comparación de las soluciones

Comparemos TODAS las soluciones, incluyendo el control "en bucle abierto", escalón de 1/ganancia:

```
fplot([ilaplace(BC_pd1/s);ilaplace(BC_pd2/s);ilaplace(BC_pd3/
s);ilaplace(BC_pi/s);ilaplace(G/s/G(0))],[0 5])
grid on, yline(1,'-.r',Label="ref"),
legend("(s+15)^2","oscil.","cancel.","PI","B. abto", Location="best")
```



Las soluciones basadas en PD se justifican intuitivamente en "subir K_p para bajar e_p", "subir K_d para amortiguar oscilaciones"... tienen un tiempo de subida más rápido que la PI. La PI se justifica en "con control Integral, ya tengo error cG(0)ero, no hace falta subir K_p tanto"... pero la PI no tiene "controlabilidad total", ha habido suerte con el tiempo de establecimiento.

Acelerar "demasiado" el proceso requiere un sobredimensionamiento de actuadores, y amplifica mucho el ruido (el "derivativo" necesita filtros de alta frecuencia a la hora de implementarse)... Quizás en este caso sea la opción PI la más "suave" con el proceso.

Ejemplo de especificaciones NO factibles con P, PD o PI

Por ejemplo, t_{est} (98%) $\leq$ 0.75 s, $e_p = 0$, $\delta < 0.2$ se conseguiría con $\sigma > 4/0.75$

```
4/0.75
```

```
ans = 5.3333
```

Según lo visto antes, necesitamos "I" para error cero, pero no puede hacer la parte real de los polos más negativa de -4.5. Tampoco cancelando el "s+9" lo conseguiríamos (quedarían $s(s+1)$ de polos y las raíces estarían simétricas alrededor de -1/2, mucho peor). Por tanto, necesitaríamos un regulador PID completo. Por brevedad, lo veremos en otros materiales.