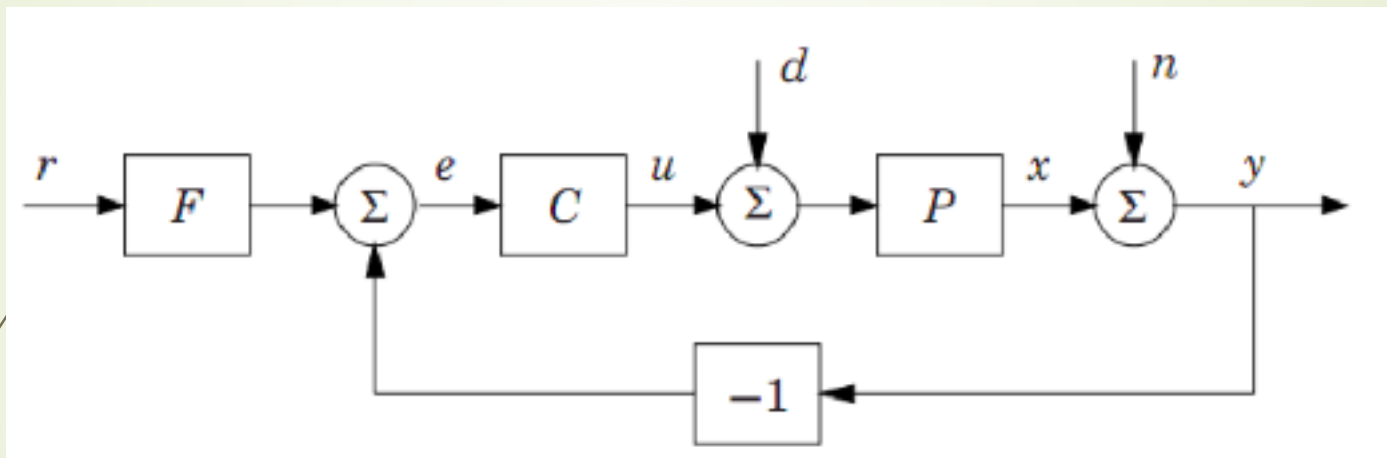


Sintonia de controladores PID baseados na resposta ao degrau



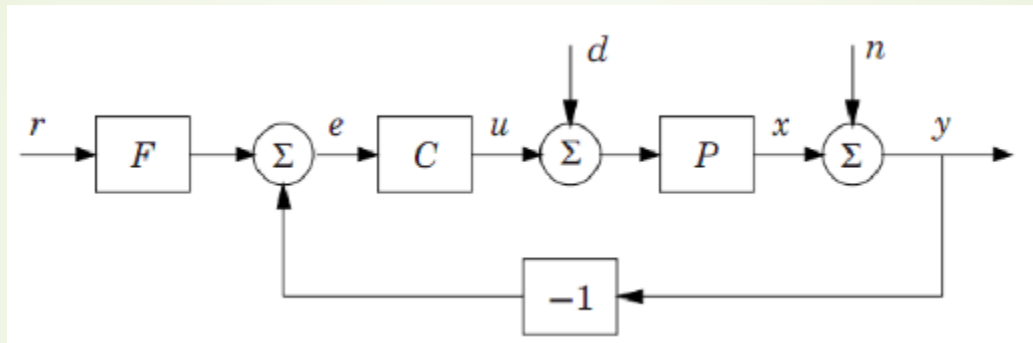
Sistemas Realimentados

O problema de controle para sistemas SISO



Arquitetura básica de uma malha de controle, envolvendo a planta P , o controlador C , o pré-compensador F , a referência r , a perturbação d e o ruído n , e a saída y .

Funções de transferência deste sistema



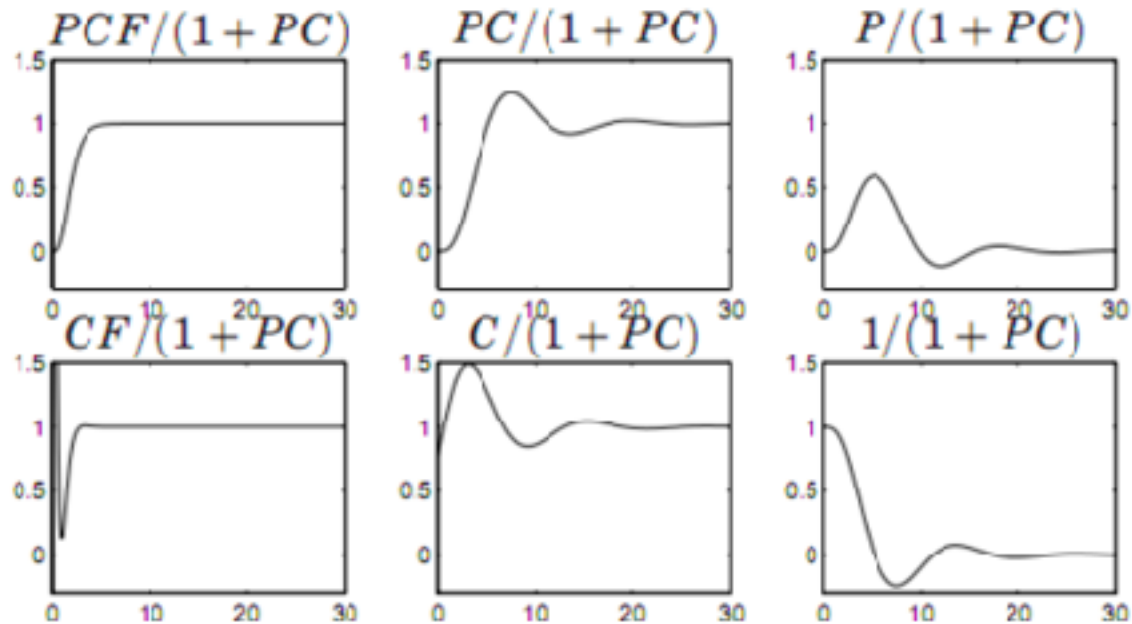
$$X = \frac{P}{1+PC}D - \frac{PC}{1+PC}N + \frac{PCF}{1+PC}R$$

$$Y = \frac{P}{1+PC}D + \frac{1}{1+PC}N + \frac{PCF}{1+PC}R$$

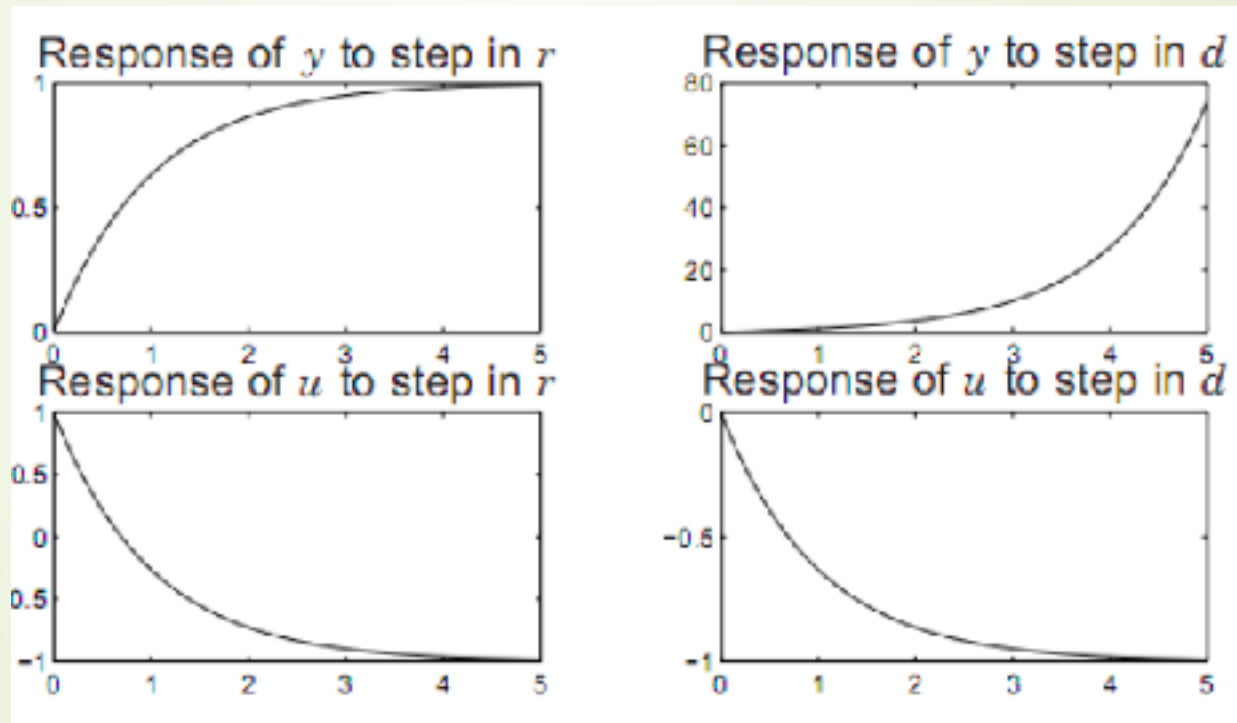
$$U = -\frac{PC}{1+PC}D - \frac{C}{1+PC}N + \frac{CF}{1+PC}R$$

Respostas ao degrau

PI control $k = 0.775$, $T_i = 2.05$ of $P(s) = (s + 1)^{-4}$ with $M(s) = (0.5s + 1)^{-4}$



Respostas para degraus aplicados em R e em D



Funções de sensibilidade

$$S = \frac{1}{1 + PC} = \frac{1}{1 + L}$$

Ambas dependem do ganho $L=PC$.

$$T = \frac{PC}{1 + PC} = \frac{L}{1 + L}$$

$$S + T = 1$$

Tipicamente, $S(0)$ é pequena e $S(\infty) = 1$.

Consequentemente, $T(0) = 1$ e $T(\infty)$ é pequena.

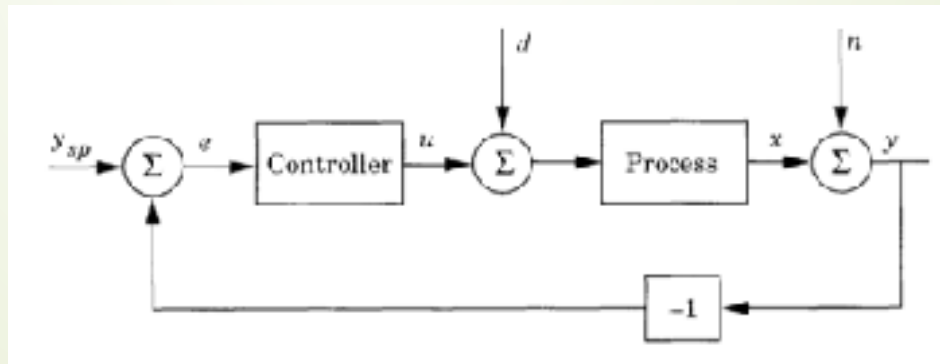
Elas refletem a robustez a variações no processo.

O Controlador PID

O controlador PID é geralmente dado pela equação

$$u(t) = K \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right)$$

onde $u(t)$ é o sinal de controle, $e(t)$ é o erro ($e(t) = y_{SP} - y(t)$)



O Controlador PID

Usando Laplace, este controlador PID é dado por

$$C(s) = K_p + \frac{K_p K_i}{s} + K_p K_d s = \frac{K_p (K_d s^2 + s + K_i)}{s}$$

Nesta forma é denominado série ou ideal, onde o ganho proporcional multiplica todos os outros ganhos.

O PID paralelo é dado por

$$C(s) = K_p s + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$



O Controlador PID

O sinal de controle é o resultado do efeito do ganho proporcional, integral e derivativo.

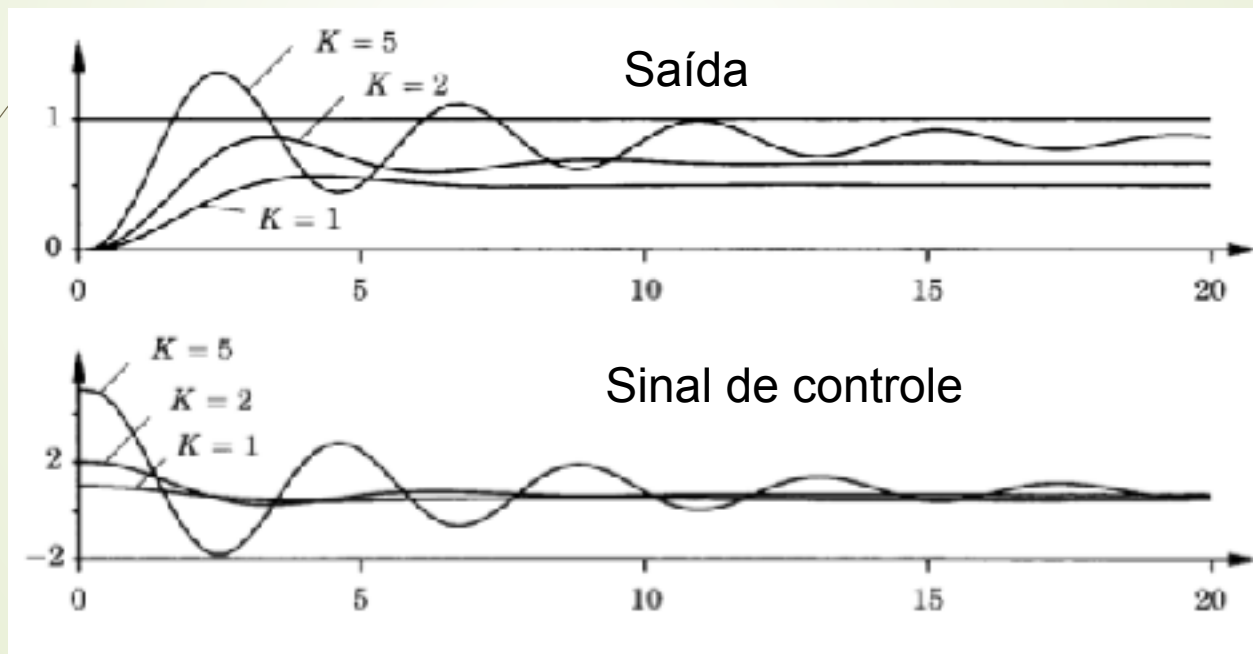
Analisaremos efeito de cada uma dessas partes, variando um ganho e fazendo os outros iguais a zero.

- Ver Cap 3 do livro Astrom, Karl J.; Hagglund, Tore. Advanced PID Control. ISA, 2006.

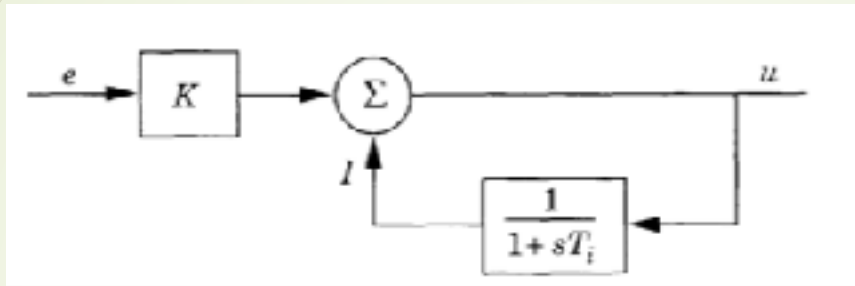
Efeito do ganho proporcional

$$u(t) = Ke(t)$$

$$G(s) = \frac{1}{(s+1)^3}$$

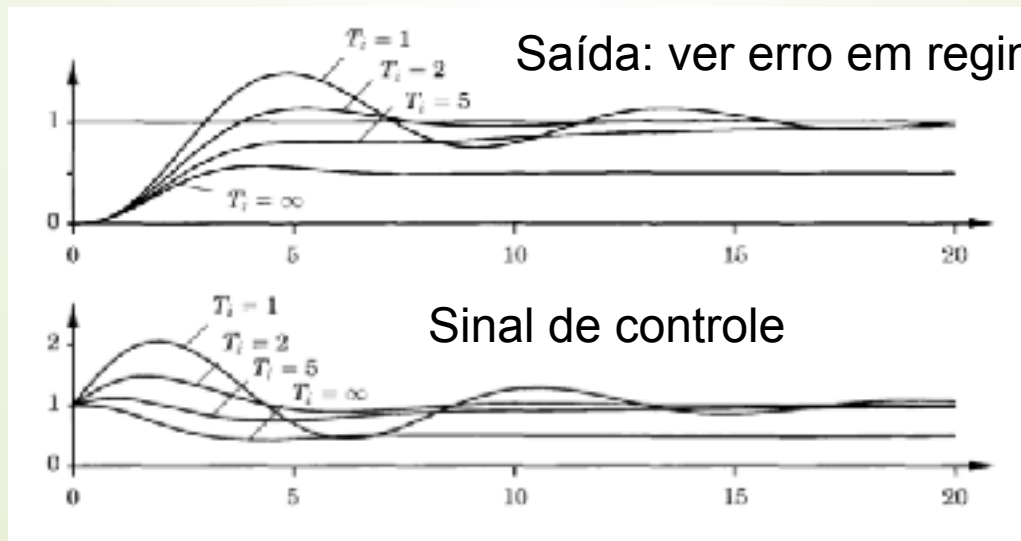


Efeito do ganho integral



$$G(s) = \frac{1}{(s+1)^3}$$

$K=1$

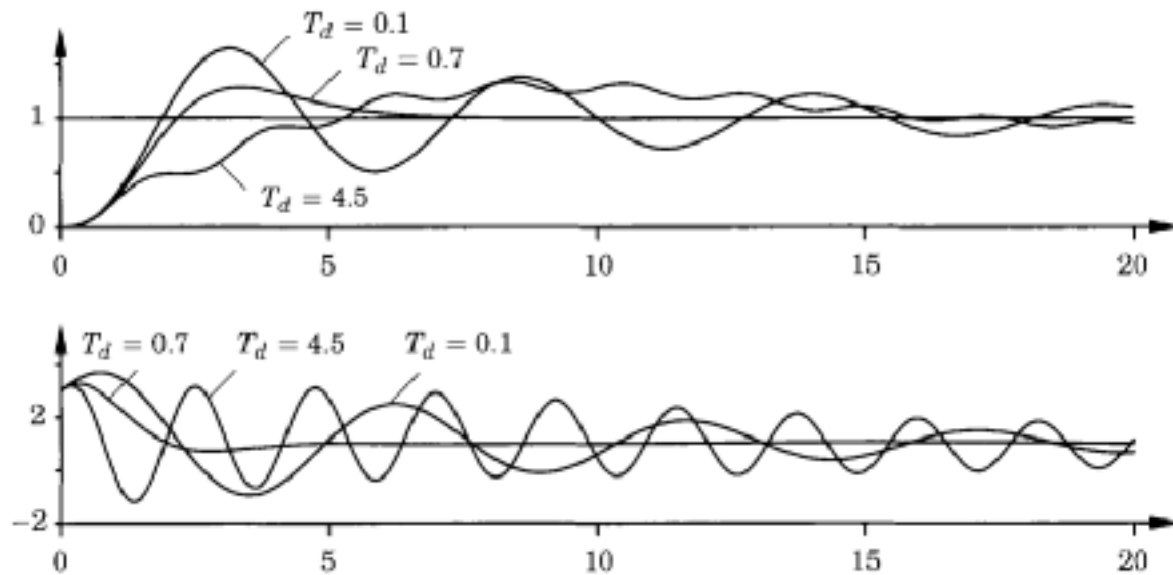


Efeito do ganho derivativo

$$u(t) = K \left(e(t) + T_d \frac{de(t)}{dt} \right)$$

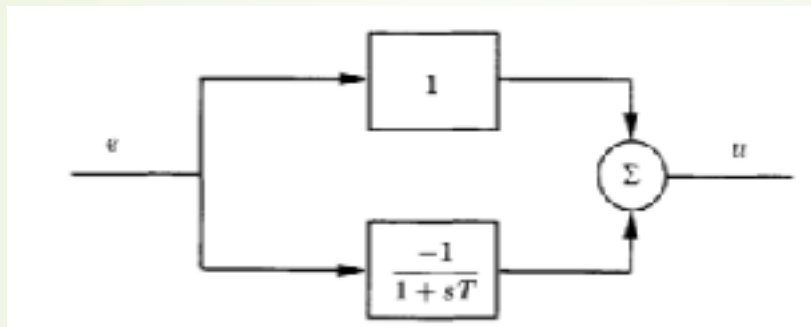
$$G(s) = \frac{1}{(s+1)^3}$$

$K=3, T_i=2$



Efeito do ganho derivativo

Implementação clássica do ganho derivativo



$$U(s) = \left(1 - \frac{1}{1+sT}\right)E(s) = \frac{sT}{1+sT}E(s).$$

Esta implementação torna o controlador realizável e introduz um filtro de primeira ordem

Resumindo o efeito dos ganhos

Ganho Proporcional: K_p

Valores mais altos de K_p reduzem o tempo de resposta e também o erro de estado estacionário.


Ganho Integral : K_i

O ganho integral introduz um polo na origem. O erro de estado estacionário para uma entrada em degrau torna-se zero. Valores mais altos de K_i tendem a tornar o sistema subamortecido.

Ganho Derivativo : K_d

O ganho derivativo introduz amortecimento, o que reduz o overshoot.

Valores mais altos de K_d tendem a aumentar os tempos de resposta.



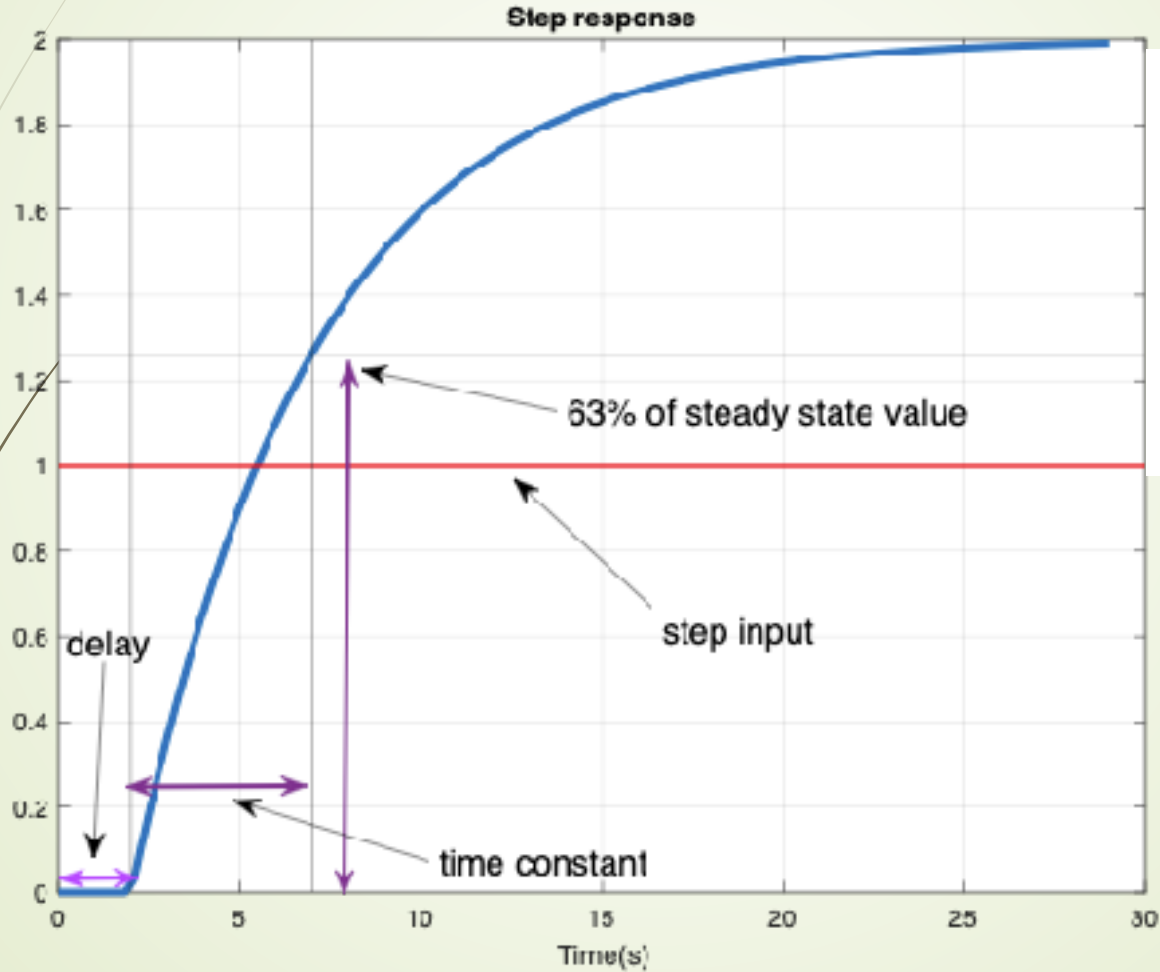
Métodos de sintonia de controladores PID

Os métodos de sintonia inicialmente propostos foram baseados nas características da resposta de grau obtidos em malha aberta.

Os parâmetros da resposta são usados para o cálculo dos ganhos do controlador, que pode ser P, P ou PID.

No próximo slide é mostrada uma resposta de grau típica usada para sintonia de controladores.

Resposta ao degrau para sintonia de controladores



$$G(s) = \frac{K e^{-\theta s}}{\tau s + 1}$$

$$K = \frac{\Delta Y}{\Delta U}$$

$\tau = \text{time constant}$

$\theta = \text{time delay}$

Alguns comentários sobre o modelo de primeira ordem+tempo morto FOPTD (first order+time delay)

O sistema em malha fechada é dado por

$$M(s) = \frac{G(s)}{1 + G(s)} = \frac{K e^{-ds}}{\tau s + 1 + K e^{-ds}}$$

Para poder analisar este sistema, o atraso deve ser aproximado por Padé, MacMaurin, Taylor.

Usamos agora a aproximação de Padé:

$$e^{-ds} = \frac{1 - \frac{ds}{2}}{1 + \frac{ds}{2}},$$

$$G(s) \approx \bar{G}(s) = \frac{K \left(1 - \frac{ds}{2}\right)}{\left(\tau s + 1\right) \left(\frac{d}{2}s + 1\right)} = \frac{K \left(1 - \frac{ds}{2}\right)}{\left(\tau + \frac{d}{2}\right)s^2 + \left(\tau + \frac{d}{2}\right)s + 1}$$

Comentários sobre o FOPTD

Com

$$G(s) \approx \bar{G}(s) = \frac{K \left(1 - \frac{ds}{2}\right)}{\left(\tau s + 1\right) \left(\frac{d}{2}s + 1\right)} = \frac{K \left(1 - \frac{ds}{2}\right)}{\left(\tau + \frac{d}{2}\right)s^2 + \left(\tau + \frac{d}{2}\right)s + 1}$$

O sistema em malha fechada é dado por

$$\bar{M}(s) \approx \frac{K \left(1 - \frac{ds}{2}\right)}{\left(\tau + \frac{d}{2}\right)s^2 + \left(\tau + \frac{d}{2}(1 - K)\right)s + 1 + K}$$

The transfer functions $\bar{G}(s)$ e $\bar{M}(s)$ are suitable for analysis.

Fazendo no Matlab:

g =

$$\exp(-2*s) * \frac{2}{5*s + 1}$$

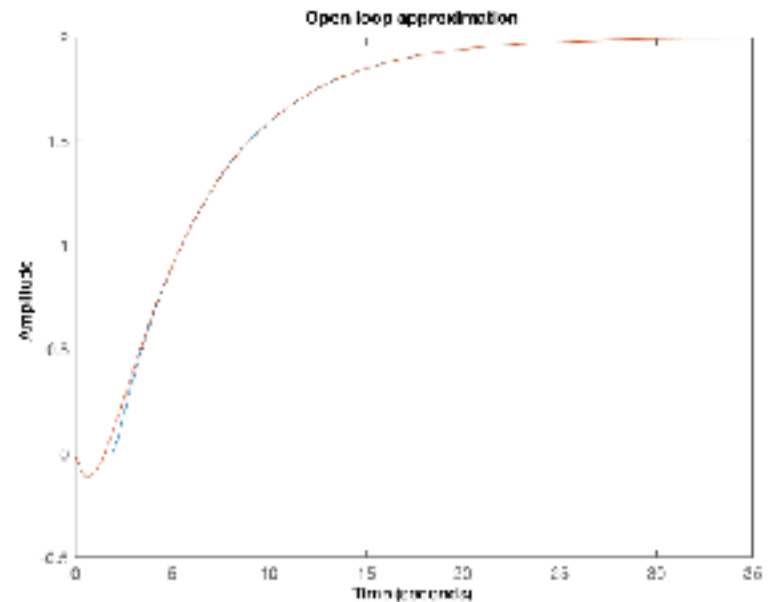
Continuous-time transfer function.

```
g1=pade(g)
```

g1 =

$$\frac{-2*s + 2}{5*s^2 + 6*s + 1}$$

Continuous-time transfer function.



Fazendo no Matlab:

```
m1=feedback(g1,1)
```

```
m1 =
```

$$\frac{-2s + 2}{5s^2 + 4s + 3}$$

Continuous-time transfer function.

```
m1m=feedback(g,1)
```

```
m1m =
```

```
A =
```

```
    x1  
    x1 -0.6
```

```
B =
```

```
    u1  
    x1 0.5
```

```
C =
```

```
    x1  
    y1 0.8
```

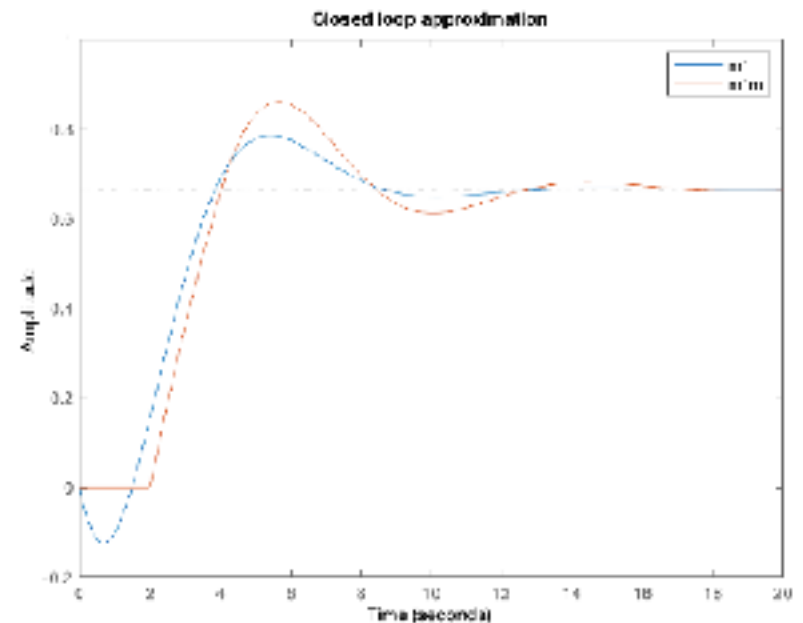
```
D =
```

```
    u1  
    y1 0
```

(values computed with all internal delays set to zero)

Internal delays (seconds): 2

Continuous-time state-space model.



Fazendo no Matlab:

Conclusão:

O Matlab faz uma melhor aproximação das respostas no tempo tanto em malha aberta quanto em malha fechada.

Entretanto os modelos utilizados para simulação do Matlab não podem ser utilizados para análise

```
>> rlocus(g)
Error using DynamicSystem/rlocus (line 65)
The "rlocus" command cannot be used for continuous-time models with delays. Use
the "pade" command to approximate delays.

>> rlocus(g1)
>>
```

Sintonia de controladores PID

Optimum Settings for Automatic Controllers

By J.G. ZIEGLER¹ and N. B. NICHOLS² • ROCHESTER, N. Y.

1942

In this paper, the three principle control effects found in present controllers are examined and practical names and units of measurement are proposed for each effect. Corresponding units are proposed for a classification of industrial processes in terms of the two principal characteristics affecting their controllability. Formulas are given which enable the controller settings to be determined from the experimental or calculated values of the lag and unit reaction rate of the process to be controlled. These units form the basis of a quick method for adjusting a controller on the job. The effect of varying each controller setting is shown in a series of chart records. It is believed that the conceptions of control presented in this paper will be of assistance in the adjustment of existing controller applications and in the design of new installations.

Sintonia de controladores PID

In these tuning rules, $K_i = \frac{1}{T_i}$, $K_d = T_d$.

Ziegler-Nichols tuning

Controller	K_p	T_i	T_d
P	$\frac{\tau}{K\theta}$	—	—
PI	$\frac{0.9\tau}{K\theta}$	3.33θ	—
PID	$\frac{1.2\tau}{K\theta}$	2θ	0.5θ

$$G(s) = \frac{K e^{-\theta s}}{\tau s + 1}$$

Sintonia de controladores PID

Uma função do Matlab para fazer estes cálculos foi fornecida

```
-> help pidtuning
```

```
function [C, c, P] = pidtuning(varargin)
```

```
Date: 17/6/2023    prof. Celso J. Munaro (DEE-CT-UFES)
```

Inputs:

G: model obtained via tf command: foptd, first order, second order (real or complex)
Models, methods, types

Model G	Methods	Types	Parameters
foptd	zie,chr,chr20,cohen	P,PI,PID	-
foptd	iaeot	PI, PID	-
foptd	lambda	PI, PID	lambda
order 1	lambda	PI	lambda
order 1	polealloc	PI	UP,ts
order 2	lambda	PID	lambda
order 2	polealloc	PI	UP,ts

Parameters:

lambda : for lambda tuning

UP, ts : for pole allocation

Outputs:

C = pid controller (C=pid(kp,kp*k1,kp*kd))

iae = integral of absolute error

Examples:

```
[C, iae]=pidtuning(g1,'method','lambda','type','PID','param',1)
```

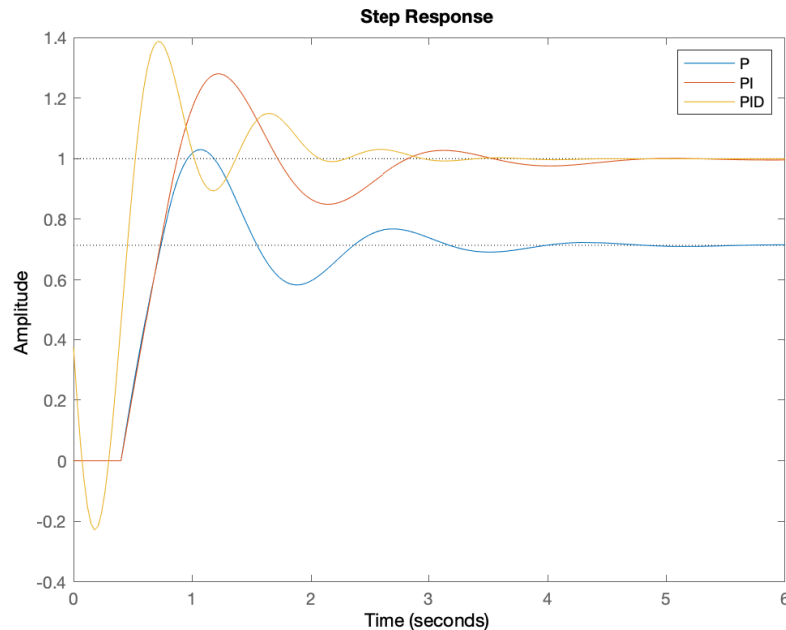
```
[C, iae]=pidtuning(g1,'method','polealloc','type','PID','param',[UP ts])
```

```
[C, iae]=pidtuning(g1,'method','zie','type','PI')
```


Sintonia de controladores PID

```
g=tf(2,[1 1],'InputDelay',0.4);  
cp=pidtuning(g,'method','zie','type','P');  
cpi=pidtuning(g,'method','zie','type','PI');  
cpid=pidtuning(g,'method','zie','type','PID');  
m1=feedback(cp*g,1);  
m2=feedback(cpi*g,1);  
g0=pade(g,2);  
m3=feedback(cpid*g0,1);  
step(m1,m2,m3);legend('P','PI','PID');shg
```

$$G(s) = \frac{K e^{-\theta s}}{\tau s + 1}$$

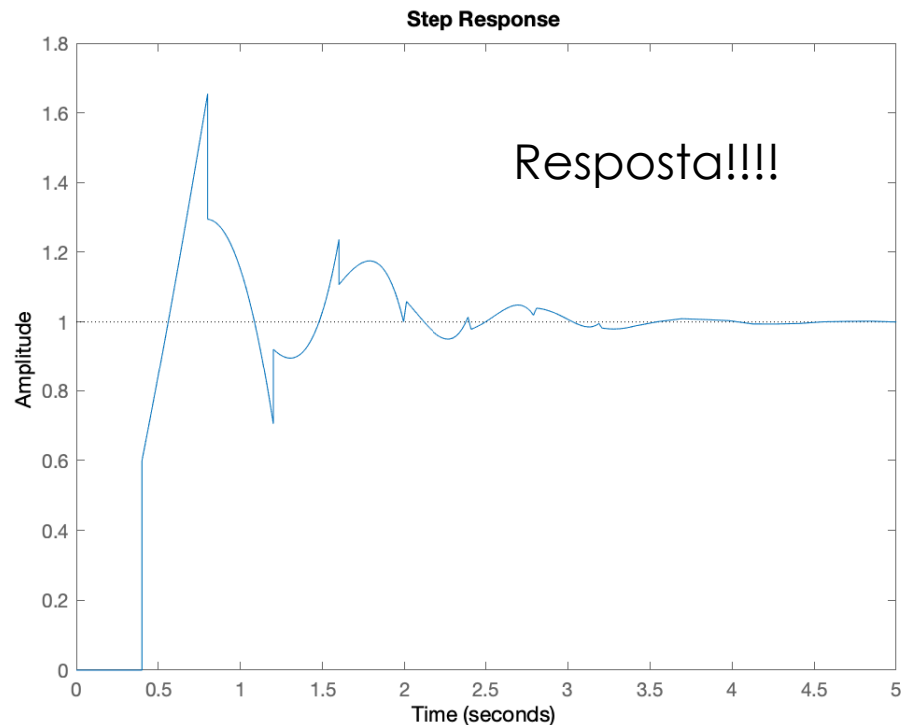


Sintonia de controladores PID

Por que não usar o comando abaixo?

```
m3=feedback(cpid*g,1);
```

```
step(m3)
```



Comentários sobre o método Ziegler-Nichols method

O ganho K_p do controlador é inversamente proporcional ao ganho K da planta.

O ganho K_p do controlador é também inversamente proporcional à relação θ/τ .

O ganho T_i do controlador é relacionado com o tempo morto do processo θ .

Controller	K_p	T_i	T_d
P	$\frac{\tau}{K\theta}$	—	—
PI	$\frac{0.9\tau}{K\theta}$	3.33θ	—
PID	$\frac{1.2\tau}{K\theta}$	2θ	0.5θ

Uso da função pidtuning para esta sintonia

```
function [ C, iae ] = pidtuning(g,'method','zie','type','P');
```

Inputs::

G = função de transferência

type = 'P', 'PI', 'PID' (entre aspas)

method = 'zie', 'chr', 'chr20', 'cohen', 'iae', 'lambda', 'ordem2'

lambda = closed loop time constant, if lambda method is used

Outputs:

$C = K_p(1 + K_i/s)$

iae = integral do erro absoluto



Método CHR

CHIEN, K. L.; HRONES, J. A.; RESWICK, J. B. On the Automatic Control of Generalized Passive Systems. Transactions ASME, v. 74, p. 175-185, 1952.

Duas opções:

- Resposta rápida sem overshoot
- Resposta rápida com 20% overshoot

CHR – sem overshoot

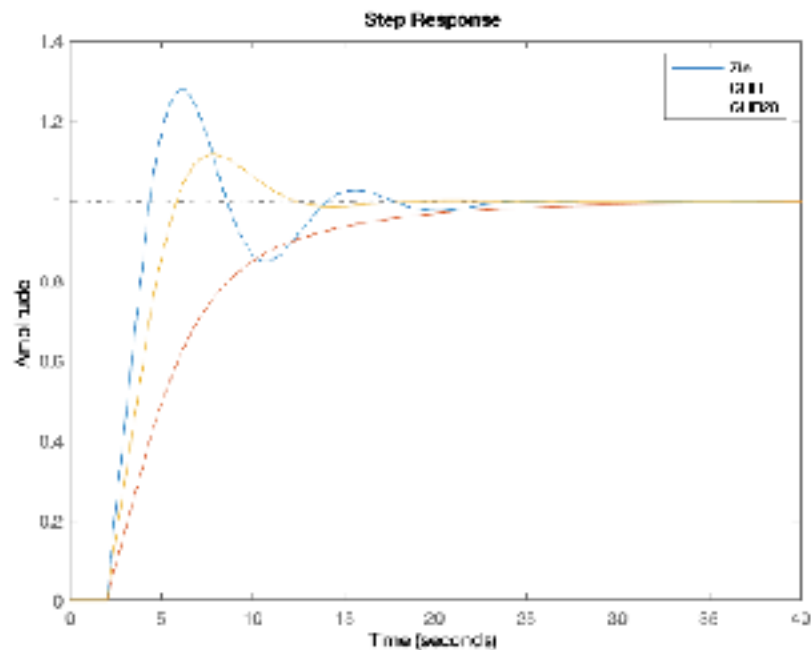
Controller	K_p	T_i	T_d
P	$\frac{0.3\tau}{K\theta}$	–	–
PI	$\frac{0.35\tau}{K\theta}$	1.16τ	–
PID	$\frac{0.6\tau}{K\theta}$	τ	0.5θ

CHR – 20% overshoot

Controller	K_p	T_i	T_d
P	$\frac{0.7\tau}{K\theta}$	—	—
PI	$\frac{0.6\tau}{K\theta}$	τ	—
PID	$\frac{0.95\tau}{K\theta}$	1.4τ	0.47θ

Comparação de 3 sintonias

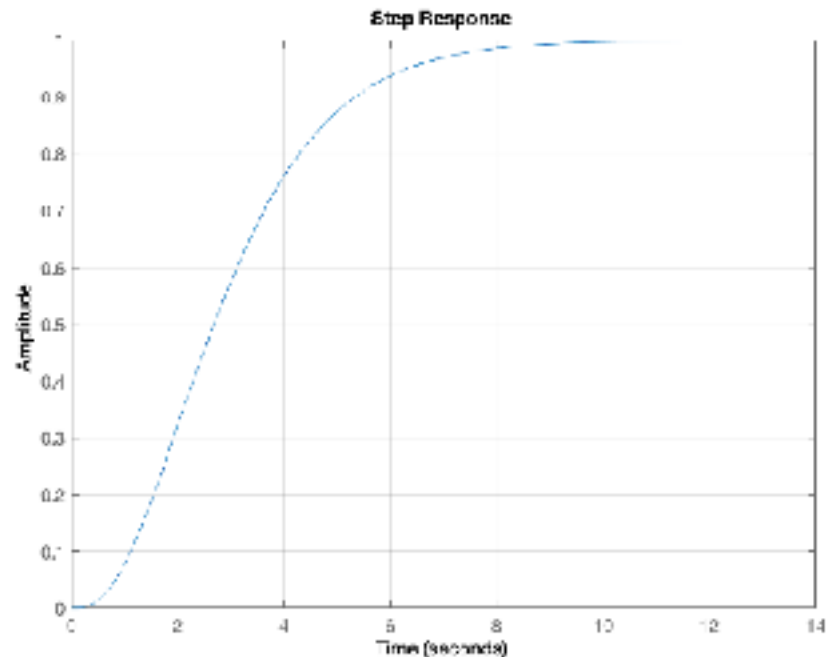
```
c1=sintonia(g,'PI','zie');  
c2=sintonia(g,'PI','chr');  
c3=sintonia(g,'PI','chr20');  
m1=feedback(c1*g,1);  
m2=feedback(c2*g,1);  
m3=feedback(c3*g,1);  
step(m1,m2,m3); legend('Zie','CHR','CHR20');
```



Aplicando métodos de sintonia a sistemas de ordem superior

Se os sistemas de ordem superior são aproximados por sistemas de primeira ordem + atraso de tempo, as regras de ajuste podem ser aplicadas.

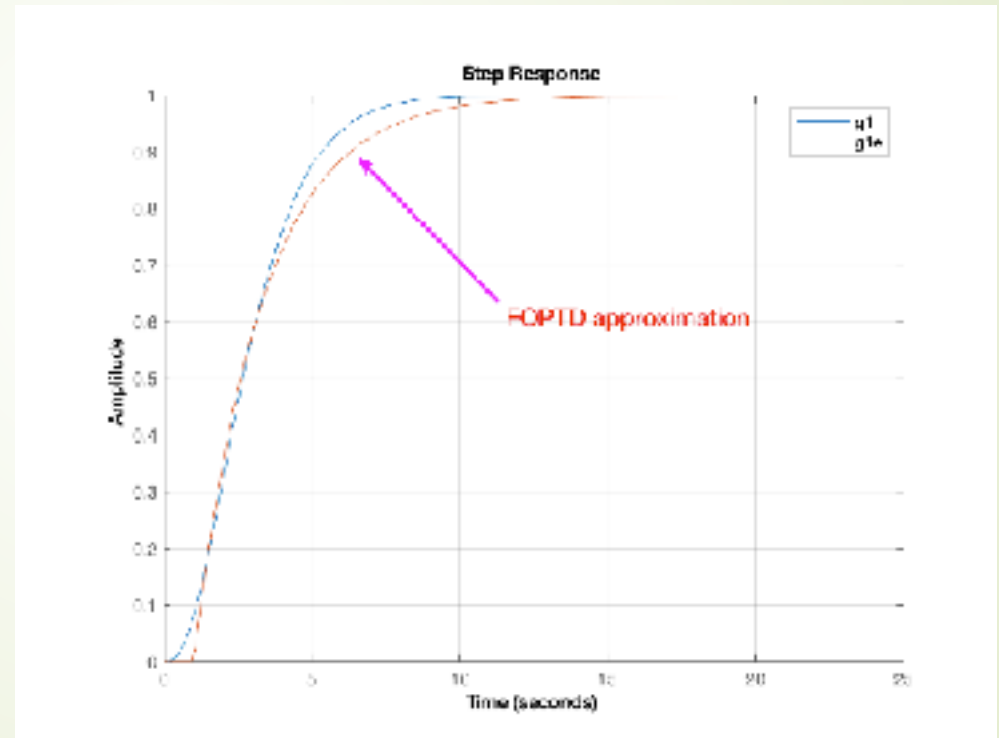
$$g_1(s) = \frac{1}{(s+1)(s+1)(s+1)}$$



Aplicando métodos de sintonia a sistemas de ordem superior

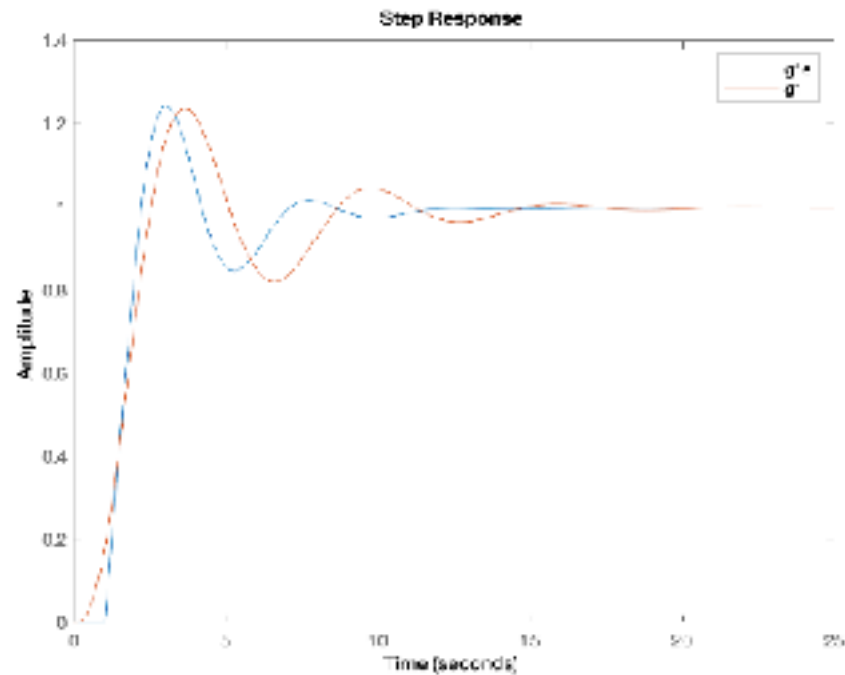
$$g_1(s) = \frac{1}{(s+1)(s+1)(s+1)}$$

$$g_{1e} = \frac{e^{-s}}{2.3s+1}$$



Aplicando métodos de sintonia a sistemas de ordem superior

```
c1=sintonia(g1e,'PI','zie');  
m10=feedback(c1*g1e,1);  
m1=feedback(c1*g1,1); % This is the real test  
step(m10,m1);legend('g1e','g1')
```



Índices de desempenho para os quais o controlador pode ser sintonizado

Integral do erro (IE): integral do sinal de erro no tempo. Este índice não é usual pois erros positivos cancelam erros negativos, podendo mascarar o resultados para respostas subamortecidas (MARLIN, 1995).

$$IE = \int_0^{\infty} e(t) dt$$

Integral do erro absoluto (IAE): integral do valor absoluto do sinal de erro no tempo. É equivalente à soma das áreas acima e abaixo do valor de referência (MARLIN, 1995).

$$IAE = \int_0^{\infty} |e(t)| dt$$

Integral do erro absoluto ponderado no tempo (ITAE): integral do tempo multiplicado pelo valor absoluto do sinal de erro no tempo. Este índice penaliza erros que se mantêm no tempo (MARLIN, 1995).

$$ITAE = \int_0^{\infty} t \cdot |e(t)| dt$$

Integral do erro quadrático (ISE): integral do quadrado do sinal de erro no tempo. Este índice, por definição, penaliza mais, valores maiores do sinal de erro (MARLIN, 1995).

$$ISE = \int_0^{\infty} [e(t)]^2 dt$$

Sintonia baseada na otimização dos critérios de desempenho do ITAE

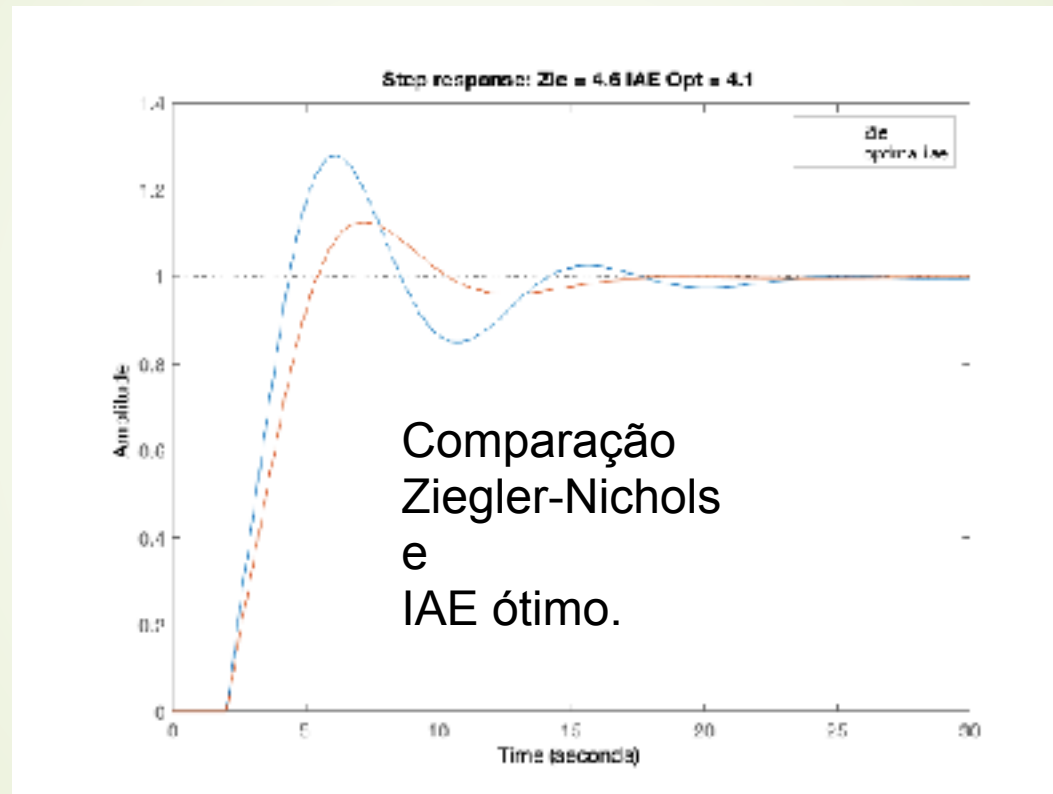
Método	K_c	T_I	T_D
ITAE - s	$\frac{0,965}{K} \cdot \left(\frac{\tau}{\theta}\right)^{0,85}$	$\frac{\tau}{0,796 - 0,1465 \cdot \frac{\theta}{\tau}}$	$0,308 \cdot \tau \cdot \left(\frac{\theta}{\tau}\right)^{0,929}$
ITAE - r	$\frac{1,357}{K} \cdot \left(\frac{\tau}{\theta}\right)^{0,947}$	$\frac{\tau}{0,842} \cdot \left(\frac{\theta}{\tau}\right)^{0,738}$	$0,381 \cdot \tau \cdot \left(\frac{\theta}{\tau}\right)^{0,995}$

Sintonia baseada na otimização dos critérios de desempenho do IAE

$$t = \frac{\tau}{\theta}$$

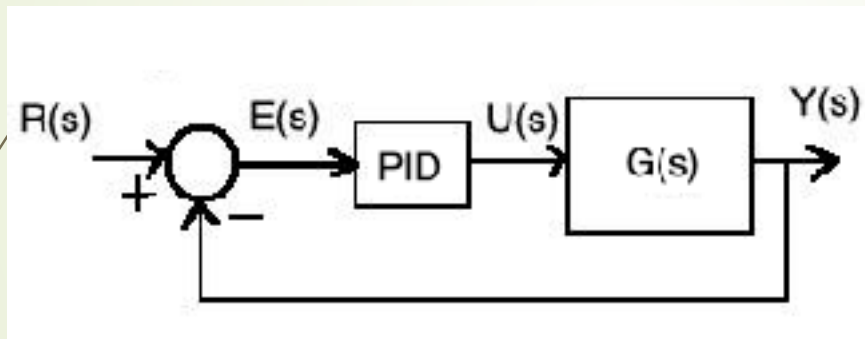
Controller	K_p	T_i	T_d
PI	$(0.7589/K) * (t^{0.861})$	$\tau / (1.02 - 0.323/t)$	—
PID	$(1.086/K) * (t^{0.869})$	$\tau / (0.74 - 0.130/t)$	$0.348\tau(t^{0.914})$

Sintonia baseada na otimização dos critérios de desempenho do IAE



Segundo método de Ziegler-Nichols baseado na resposta em frequência

Neste método, os ganhos K_i e K_d são feitos iguais a zero e o ganho proporcional K_p é aumentado até que a resposta se torne oscilatória quando um degrau é aplicado.



Este ganho, denotado por K_u é anotado, bem como o período de oscilação T_u .

Segundo método de Ziegler-Nichols baseado na resposta em frequência

Os ganhos do controlador P, PI ou PID são dados pela tabela abaixo.

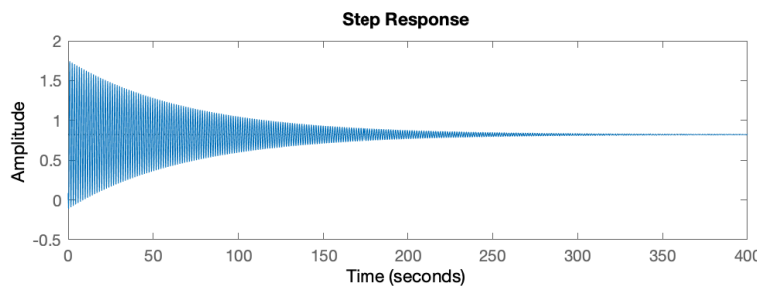
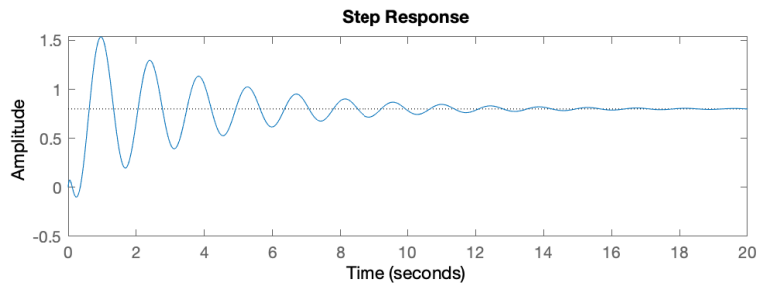
Controller	K/K_u	T_i/T_u	T_d/T_u	T_p/T_u
P	0.5			1.0
PI	0.4	0.8		1.4
PID	0.6	0.5	0.125	0.85

T_p =estimativa da dinâmica dominante

Este ganho, denotado por K_u é anotado, bem como o período de oscilação T_u .

Exemplo: Segundo método de Ziegler-Nichols

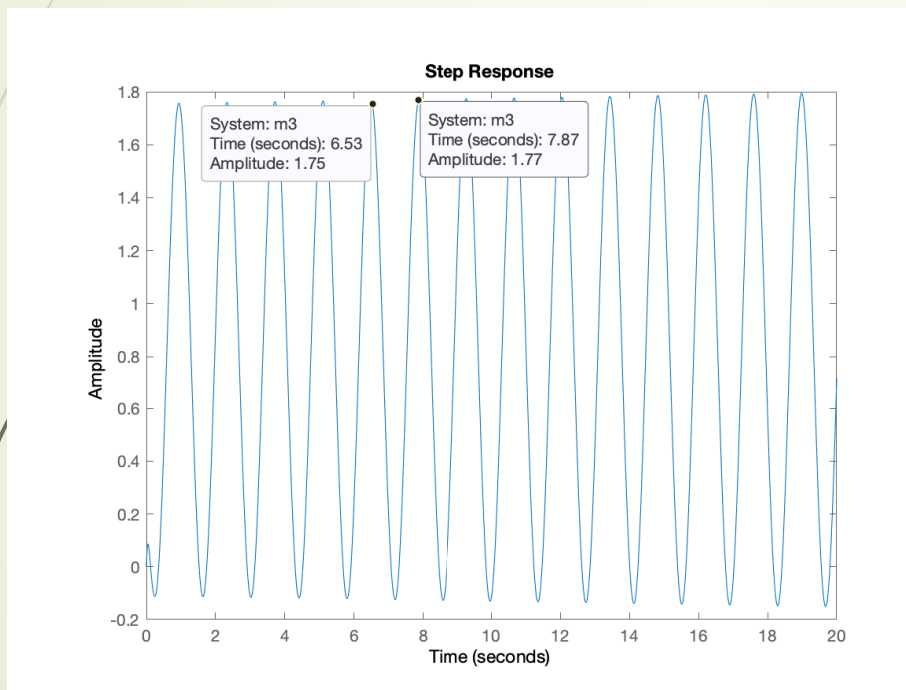
```
g=tf(2,[1 1],'InputDelay',0.4);
```



Superior, $K=2$. Inferior, $K=2.2$. Oscilação com $K=2.32$.

Exemplo: Segundo método de Ziegler-Nichols

```
g=tf(2,[1 1],'InputDelay',0.4);
```



Controlador PI:

$$K = 0.5 * K_u = 1.16$$

$$T_i = 0.8 * T_u = 0.8 * 1.34 = 1.07$$

Oscilação: $K_u=2.32$ e $T_u=1.34s$.

Comparação do primeiro e segundo método:

Primeiro método:

`cpi=pidtuning(g,'method','zie','type','PI')`

$cpi = K_p + K_i / s$, with $K_p = 1.12$, $K_i = 0.845$

Segundo método:

$$K = 0.5 * K_u = 1.16$$

$$T_i = 0.8 * T_u = 0.8 * 1.34 = 1.07 \quad K_i = 1/T_i = 0.93$$

Definição dos parâmetros do controlador PID no Matlab

`C=pid(Kp,Ki,Kd)`

`C =`

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

`C=tf([Kp Ki],[1 0])`

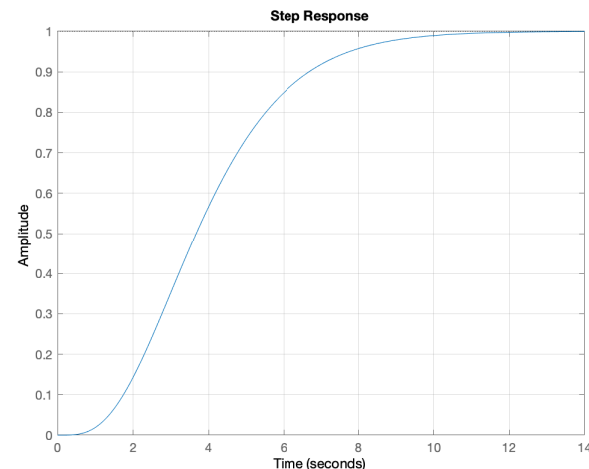
Para o caso de PID série, que é o projetado pelos métodos de sintonia, fazer

`C=pid(Kp,Kp*Ki,Kp*Kd)`

`C=tf([Kp Kp*Ki],[1 0])`

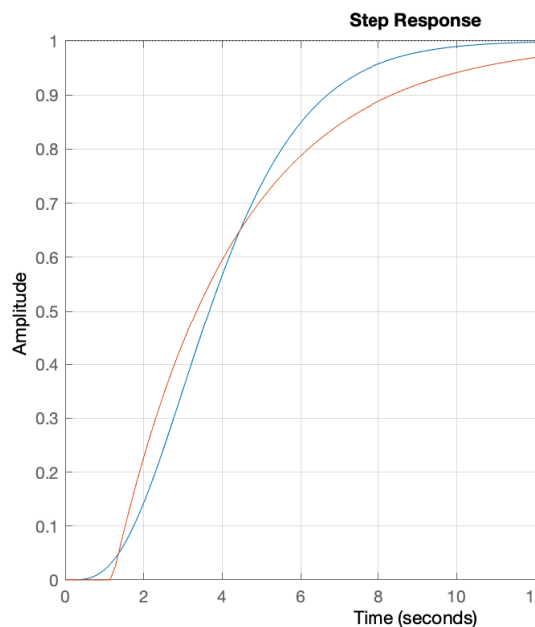
Exemplos no Matlab:

```
g=tf(1,poly(-[1 1 1 1]));  
step(g);shg  
K=1;  
d=1.8; % time delay que afeta os projetos  
tau=4.3-d;  
g1=tf(K,[tau 1],'InputDelay',d);  
g10=pade(g1,2);  
step(g,g1);shg  
mr=tf(1,[tau-1 1],'InputDelay',d);  
c=pidtuning(g1,'method','zie','type','PI');  
m1=feedback(c*g,1);  
c=pidtuning(g1,'method','chr20','type','PI');  
m2=feedback(c*g,1);  
c=pidtuning(g1,'method','iaeot','type','PI');  
m3=feedback(c*g,1);  
step(m1,m2,m3);legend('zie','chr20','iaeot');
```



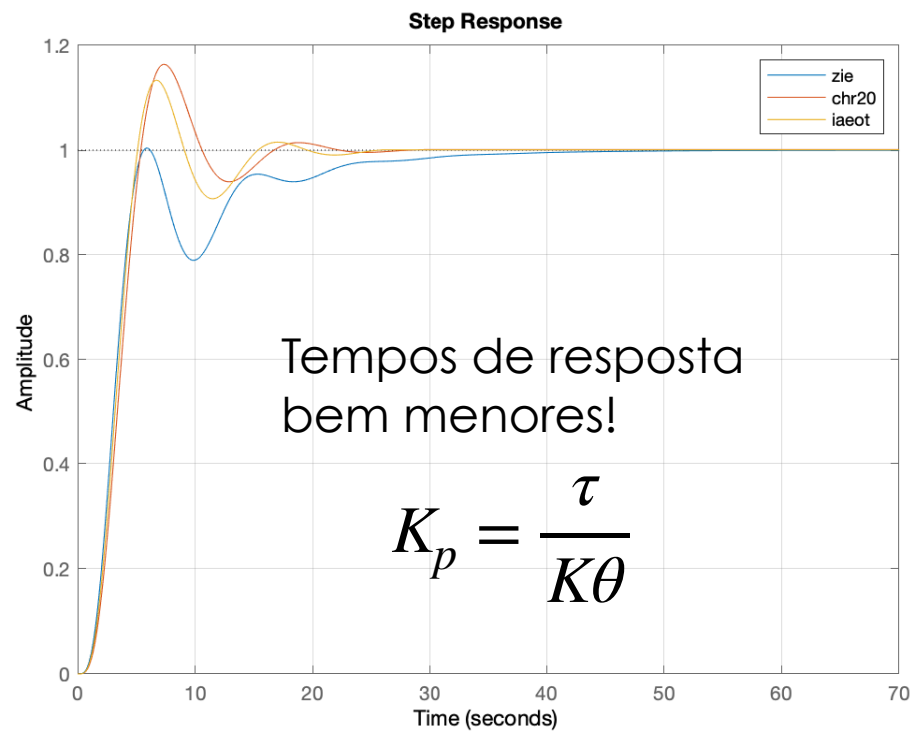
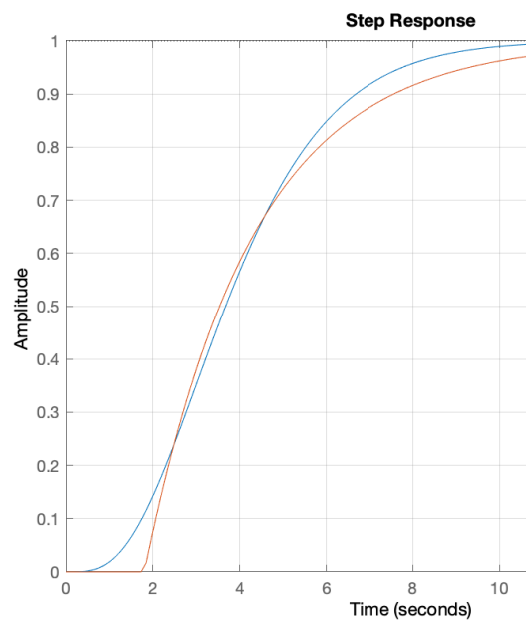
Exemplos no Matlab:

Aproximação usando $\theta = 1.2s$



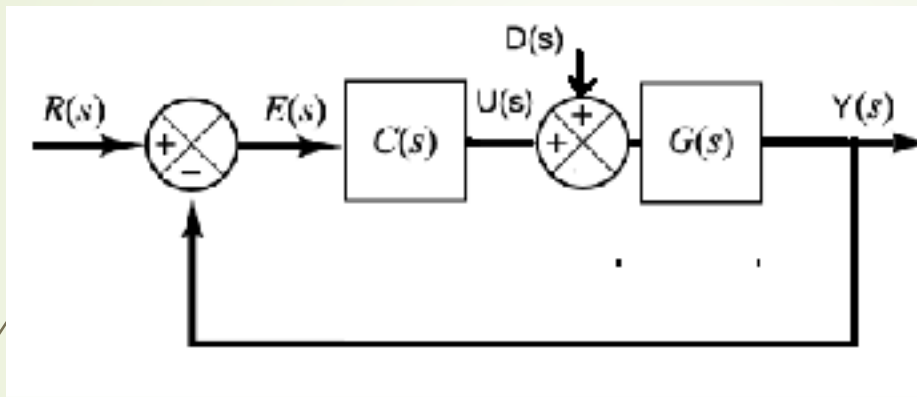
Exemplos no Matlab:

Aproximação usando $\theta = 1.8s$



Exemplos no Matlab:

Vamos analisar agora os sinais $U(s)$ e $Y(s)$ em resposta a degraus em $R(s)$ e $D(s)$

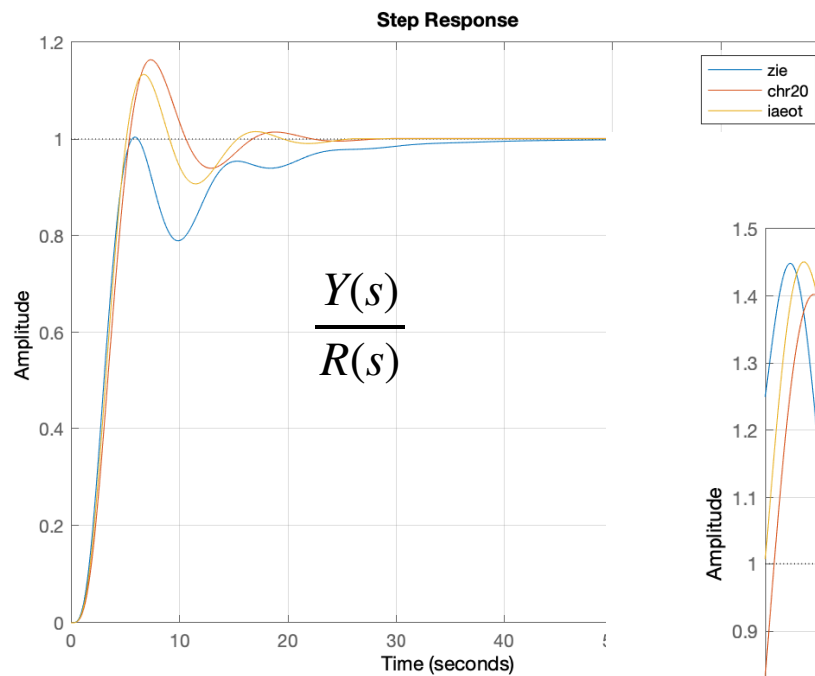


$R(s)$ é a referência a ser seguida
 $D(s)$ é um distúrbio a ser rejeitado.

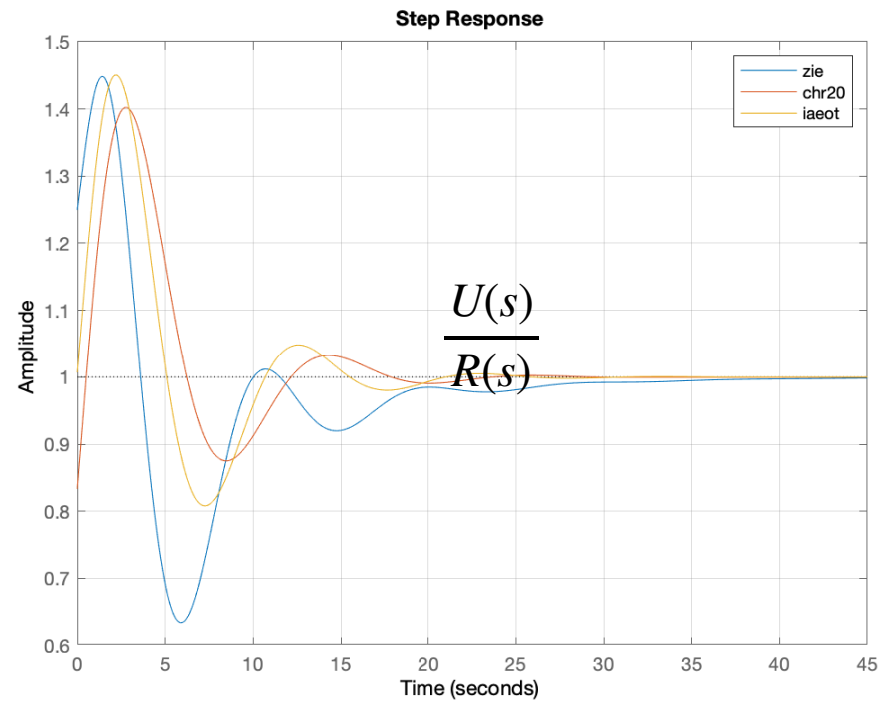
Usaremos a aproximação $\theta = 1.8s$

Exemplos no Matlab:

Saídas

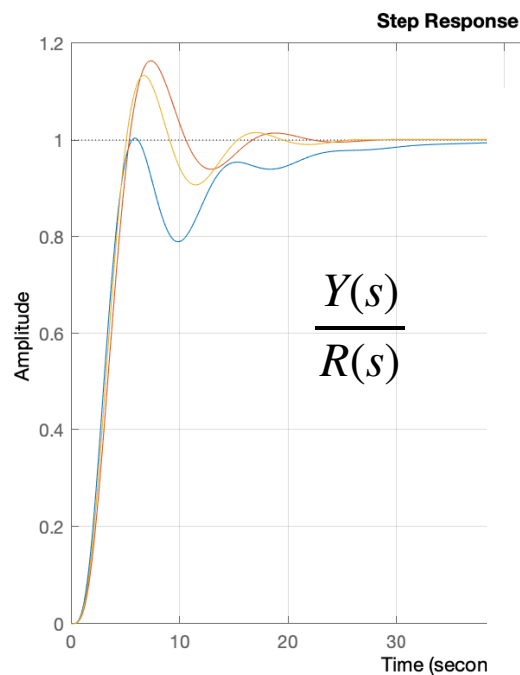


Sinais de controle

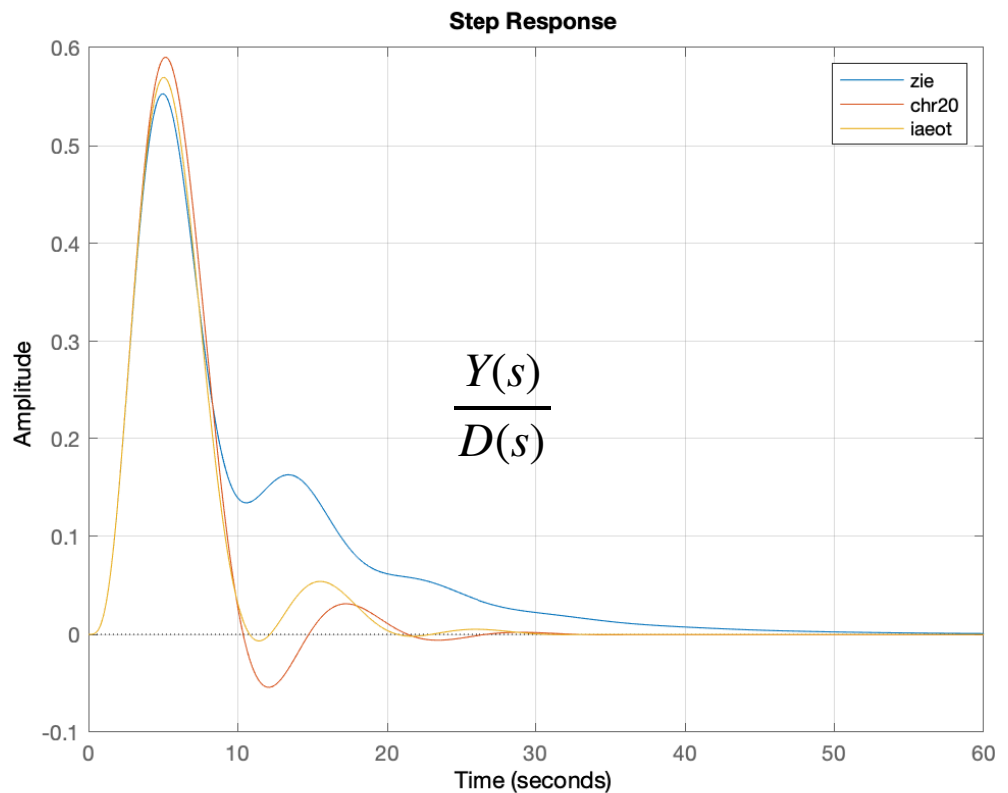


Exemplos no Matlab:

Saídas

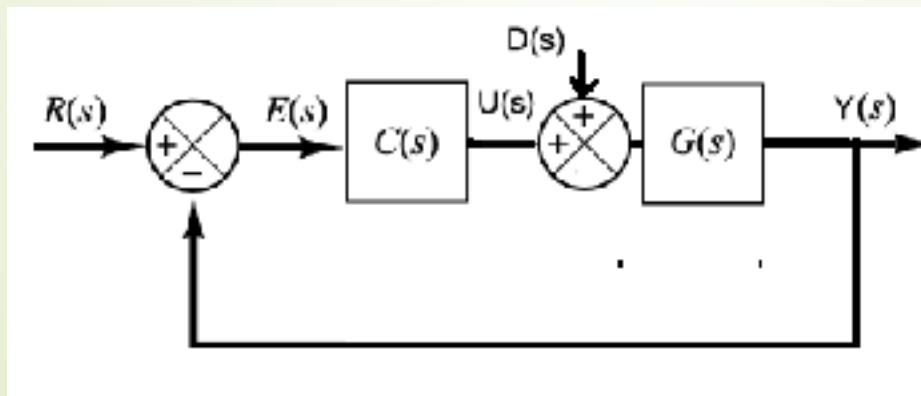


Efeito do distúrbio



Comandos no Matlab para gerar estas respostas

```
c=pidtuning(g,'method','zie','type','PI');  
m1=feedback(c*g,1); % Resposta y ao degrau R  
m1u=feedback(c,g); % Sinal de controle U ao degrau R  
m1d=feedback(g,c); % Resposta y ao degrau no distúrbio D
```





Comandos no Matlab para gerar estas respostas

Para simular o controlador PID com funções de transferência g com tempo morto, melhor aproximar o atraso usando Padé:

```
c=pidtuning(g,'method','zie','type','PI');  
g0=pade(g,1);% Aproximação de Padé de ordem 2  
m1=feedback(c*g0,1);  
step(m1);
```