UPF
Eng. Elétrica

Controle Automático III

Prof. Dr. Eng.
Fernando Passold

ESTUDO DE CASO

Objetivo: testar/projetar controladores clássicos sobre uma mesma planta comparando resultados na forma de uma tabela e gráfico de respostas temporais.

Os controladores à serem testados são:

1) Proporcional;
2) PI
3) PI + Zero
4) PI + Pólos Dominantes
5) Lag (Atraso)
6) Lead (Avanço)
7) PD
8) PID
9) Lead-Lag

Objetivo Geral:

Projetos de Controladores Clássicos, formato digital, usando como ferramenta: Lugar das Raízes.

Resumo sobre Papel/Objetivo de cada Controlador:

1) Proporcional;
2) PI
3) PI + Zero
4) PI + Pólos Dominantes
5) Lag (Atraso)
6) Lead (Avanço)
7) PD
8) PID
9) Lead-Lag

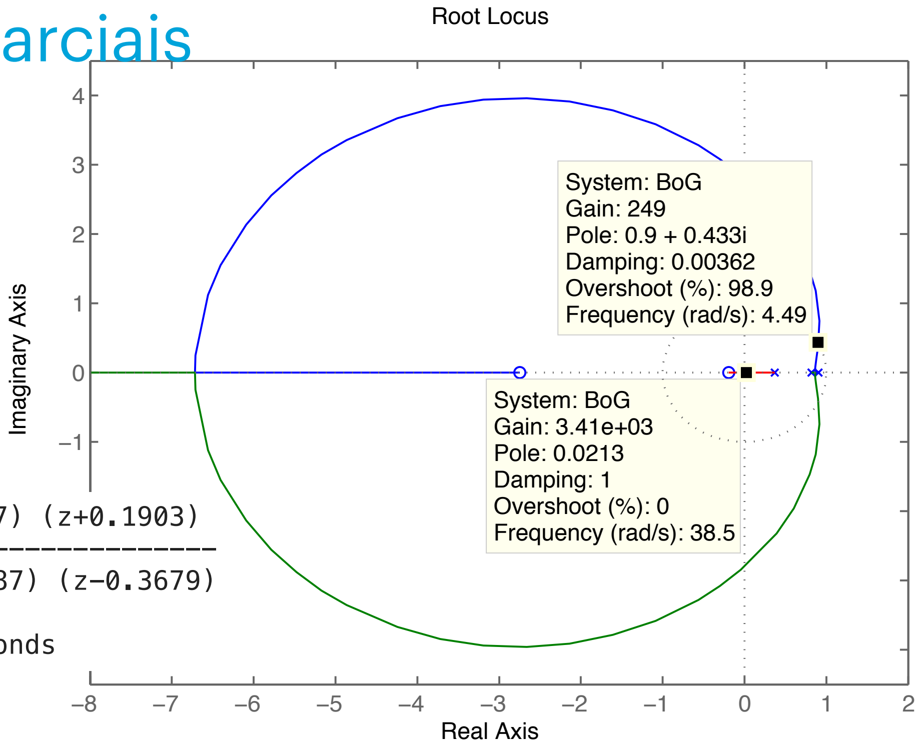
Soluções Parciais

Planta adotada:

```
>> zpk(G)

          1
  ------------------
  (s+10) (s+2) (s+1)
```

BoG(z):

```
  0.00012224 (z+2.747) (z+0.1903)
  --------------------------------
  (z-0.9048) (z-0.8187) (z-0.3679)

Sample time: 0.1 seconds
```



Requisitos de controle: %OS < 5%
ts < metade do tempo de assentamento do controlador proporcional.

Determinando fator de amortecimento, zeta, com base em %OS:

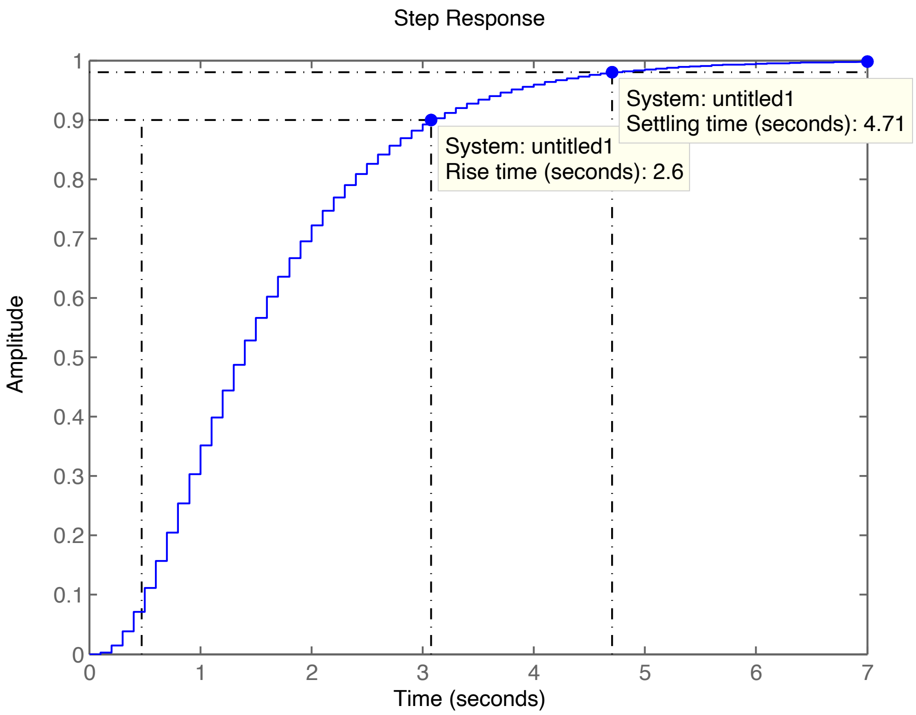
```
>> zeta=(-log(OS/100))/(sqrt(pi^2+(log(OS/100)^2)))
zeta = 0.6901
```

Notar ganho da planta para entrada degrau unitário:

```
>> dcgain(BoG)

    0.0500

>>
```

Note: não é unitário!

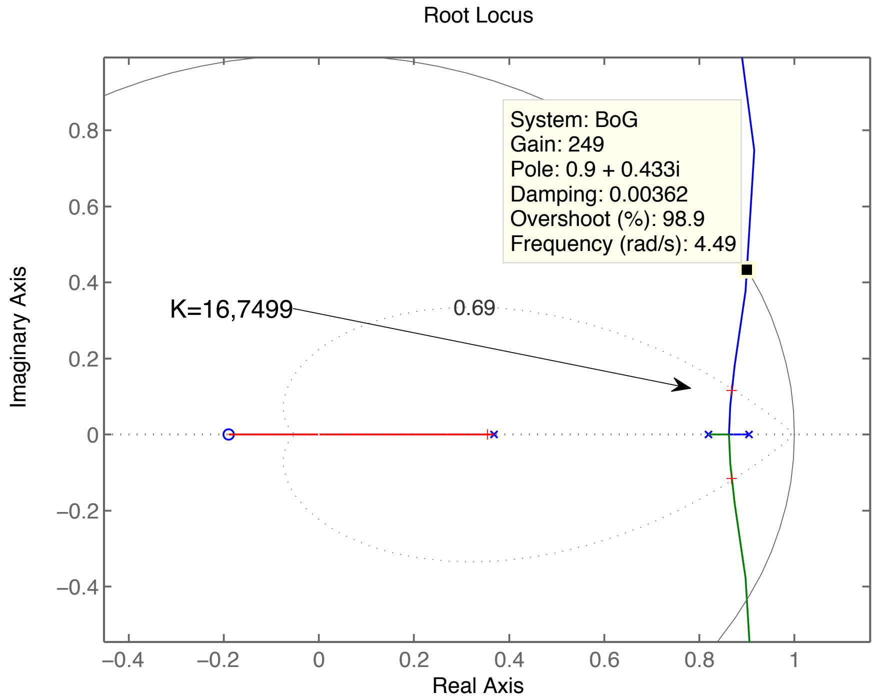
Resposta em Malha-aberta:



1a-Parte: fechar malha de realimentação com controlador Proporcional.

Obtendo RL para BoG(z) e sobrepondo linha com zeta = 0,6901:

K1 = 16.7499



Ganho DC da FTMF ?

```
>> dcgain(ftmf1)

    0.4572
```

Verificando resposta ao degrau para saída y(∞) = 1,0.:

```
>> figure; step(1/
ans*ftmf1)
>> 1/ans

    2.1874 ← Amplitude do degrau!
```

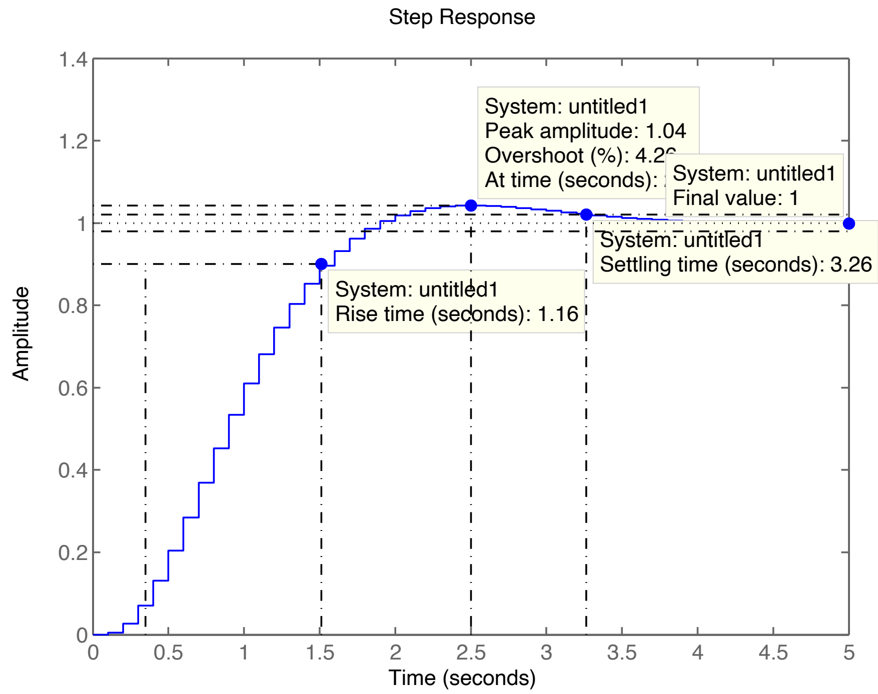
$$BoG(z) = \frac{0.00012224\ (z+2.747)\ (z+0.1903)}{(z-0.9048)\ (z-0.8187)\ (z-0.3679)}$$

Determinar erro de regime permanente —> ?

ts1 = 3,26 segundos

Para próximos controladores:

ts_desejado = 3,26 / 2 = 1,63 (máximo).



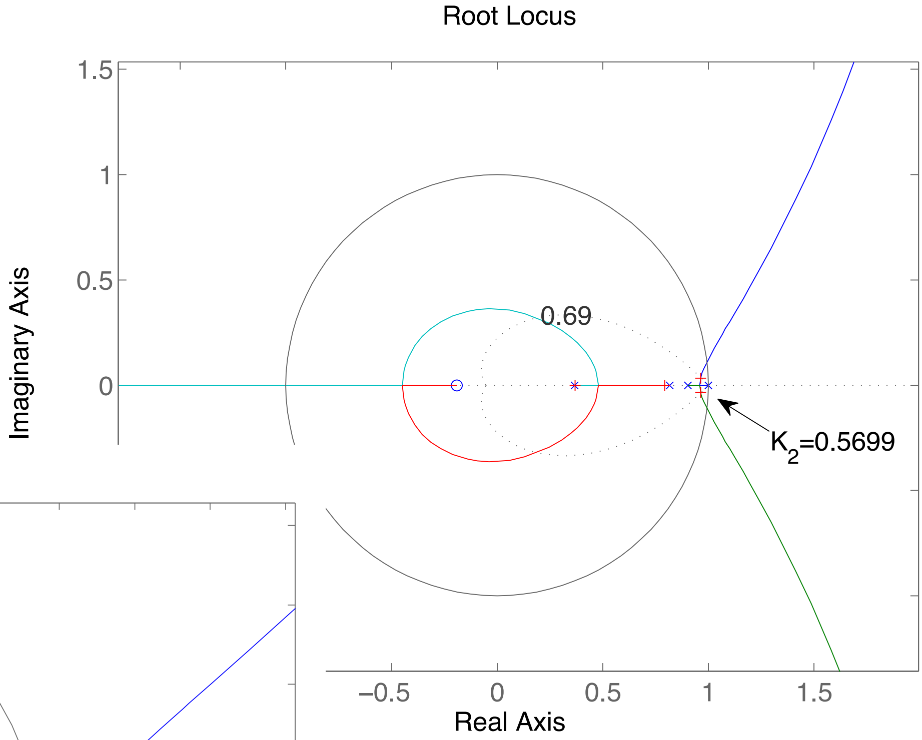
Continua.

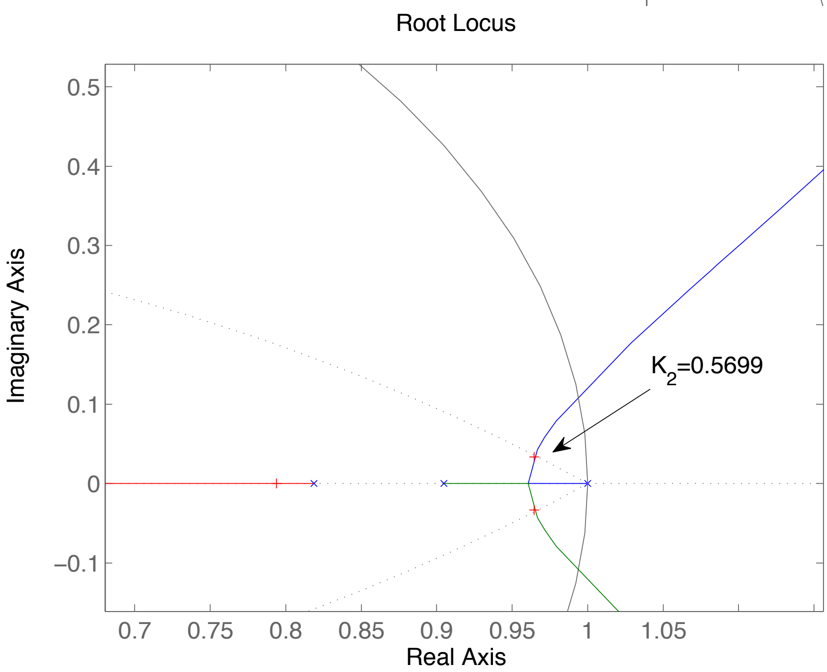
2) Controlador Integrador "puro":

C2(z):

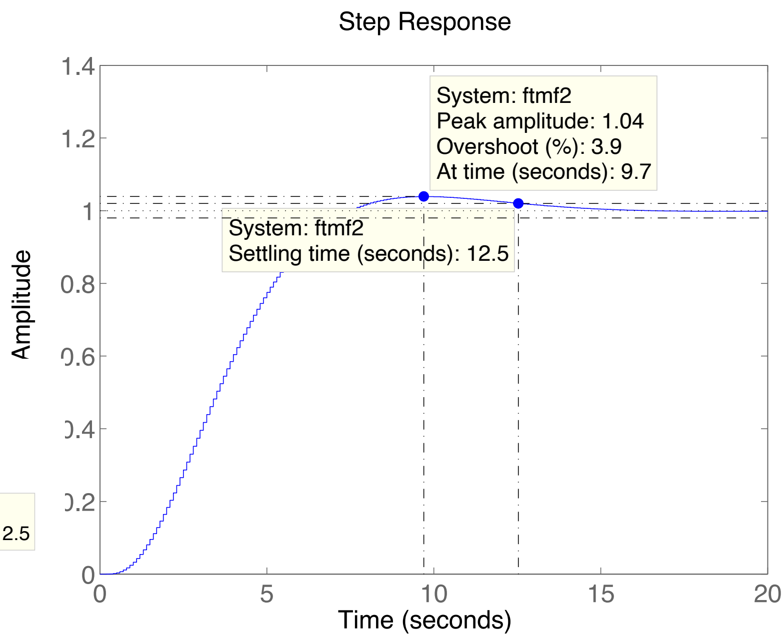
$$\frac{1}{(z-1)}$$

RL resultante:
K2 = 0,5699

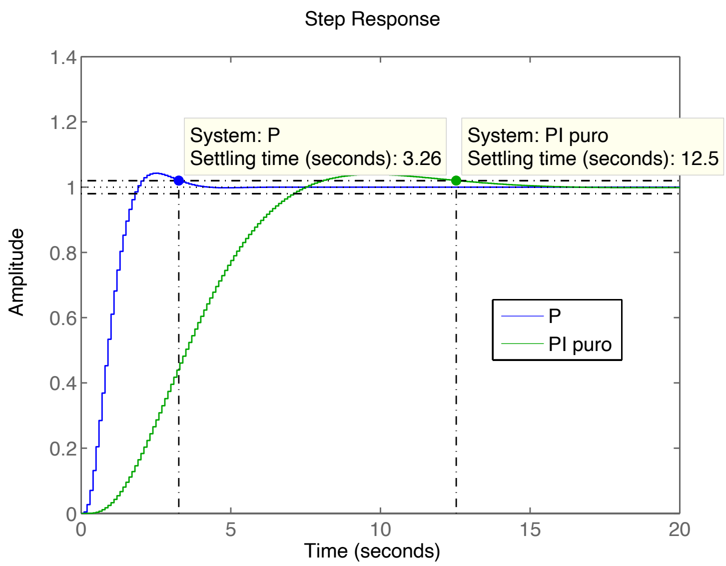




$$BoG(z) = \frac{0.00012224\ (z+2.747)\ (z+0.1903)}{(z-0.9048)\ (z-0.8187)\ (z-0.3679)}$$



Resposta do sistema:
ts = 12,5
%OS = 3,9%



3) Controlador PI = P+I

(zero "extra":)

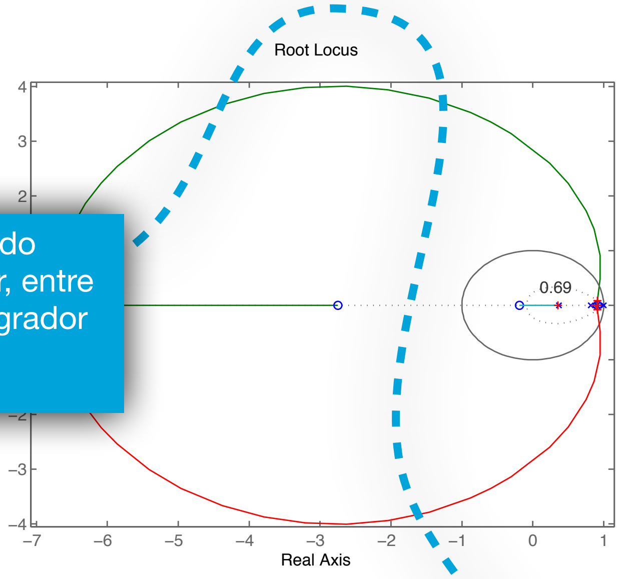
$$C(z) = K_p + \frac{K_i}{(z-1)} = \frac{K_p z - K_p + Ki}{(z-1)}$$

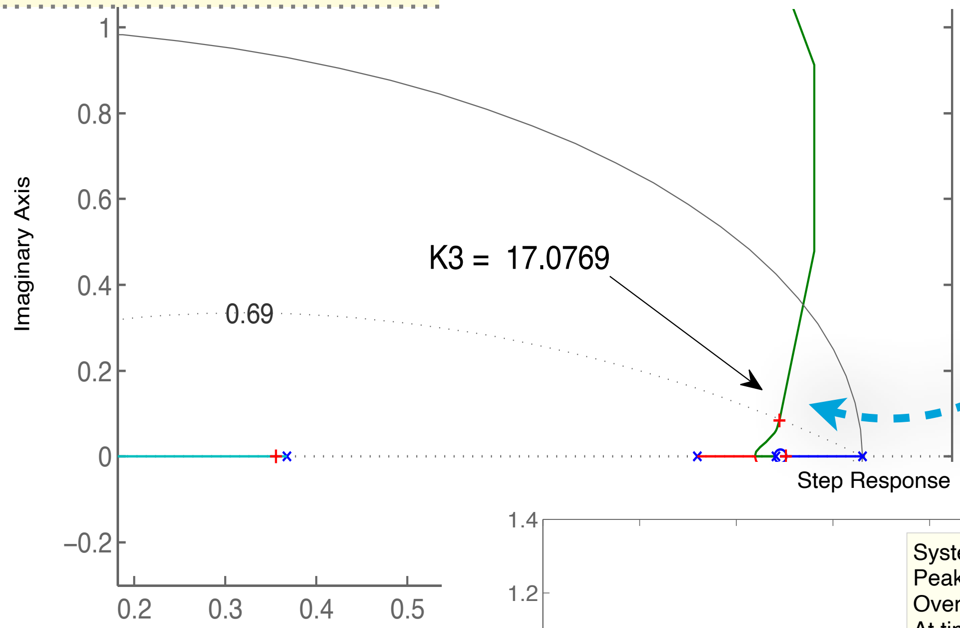
$$C(z) = \frac{K_p\left[z - \left(1 - \frac{K_i}{K_p}\right)\right]}{(z-1)}$$

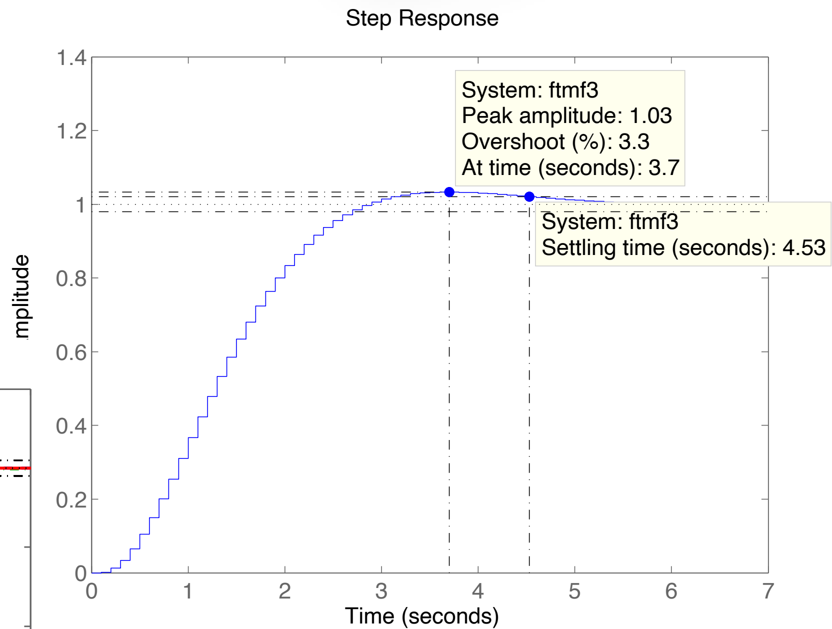
$$C(z) = \frac{K(z-0{,}91)}{(z-1)}$$

Note: zero do controlador, entre o pólo integrador e o pólo

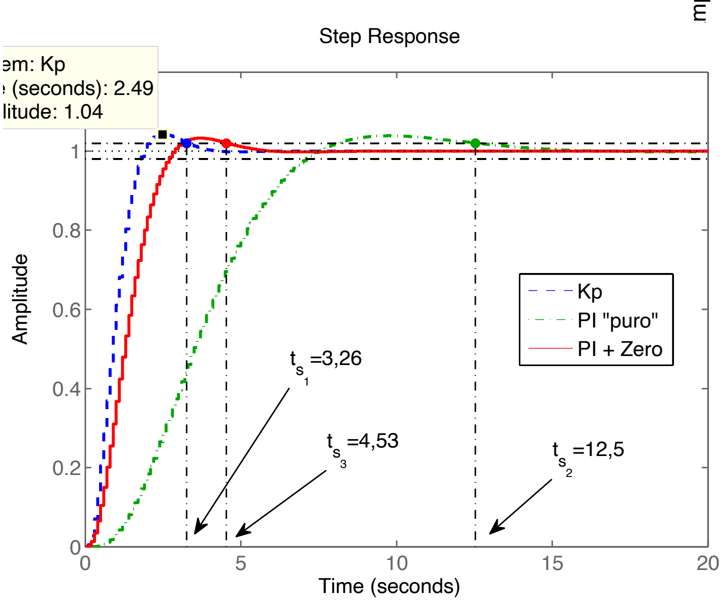
```
              0.00012224 (z+2.747) (z+0.1903)
BoG(z) =  --------------------------------
             (z-0.9048) (z-0.8187) (z-0.3679)
```







K3 = 17,0769

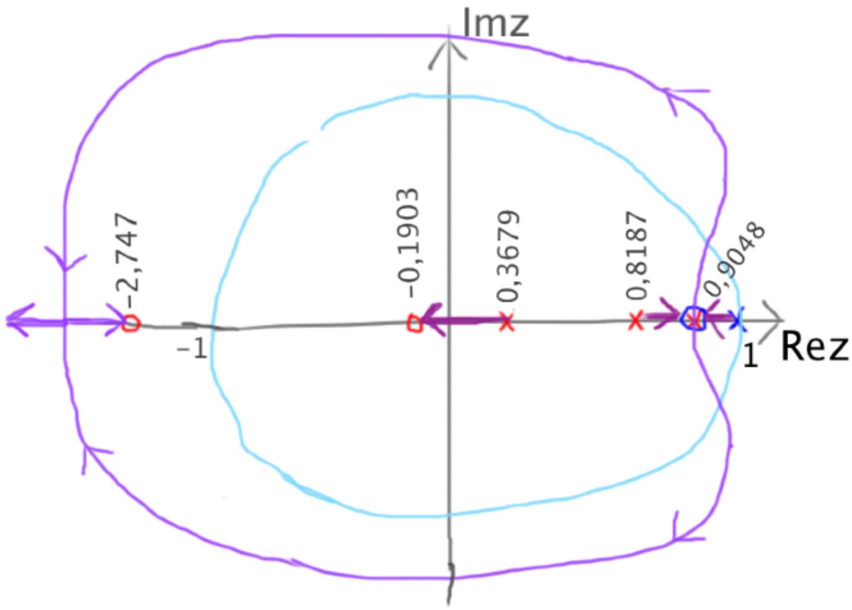


Resposta MF resultante:
ts = 4,53
%OS = 3,3

4) Controlador PI + cancelamento do pólo dominante da planta

Idéia:

$$C4(z) = \frac{K4\ (z - 0{,}9048)}{(z - 1)}$$



$$BoG(z) = \frac{0.00012224\ (z+2.747)\ (z+0.1903)}{(z-0.9048)\ (z-0.8187)\ (z-0.3679)}$$

Fica + rápido ?

Resultados…

K = 16,1225

ts = 11,3 segundos (piora)….

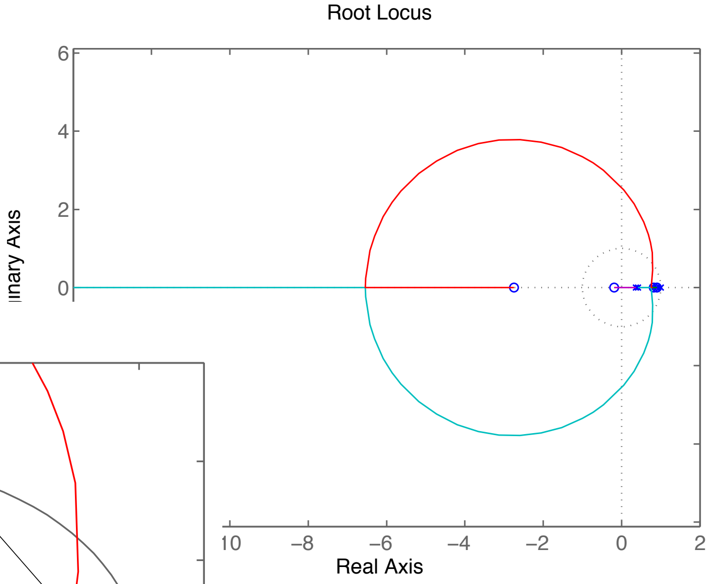
5) Controlador PI com cancelamento de 2 pólos dominantes

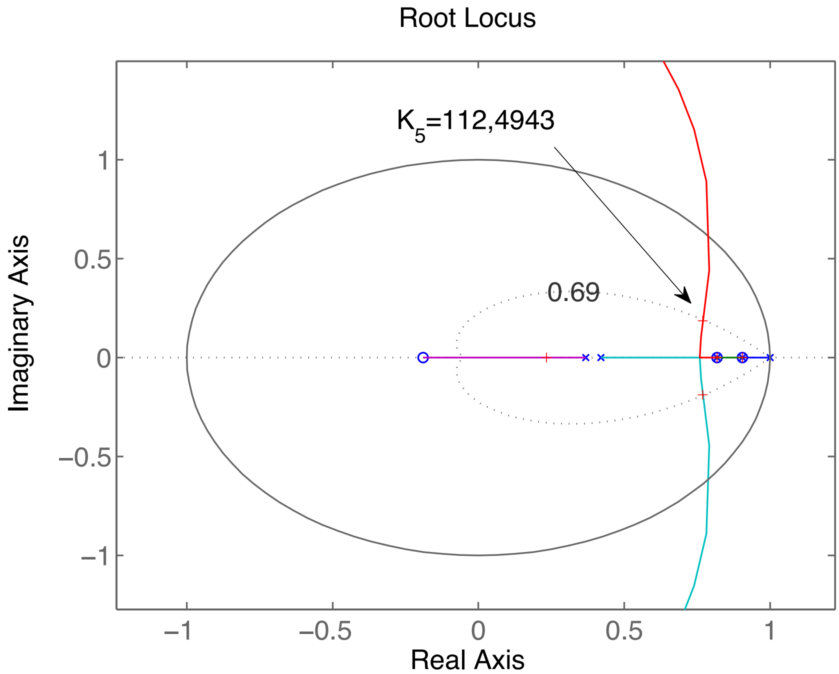
$$C5(z) = \frac{K5 * (z-0.9048)\ (z-0.8187)}{(z-1)\ (z-0.42)}$$

$$BoG(z) = \frac{0.00012224\ (z+2.747)\ (z+0.1903)}{(z-0.9048)\ (z-0.8187)\ (z-0.3679)}$$

Resultados:

K5 = 112.4943

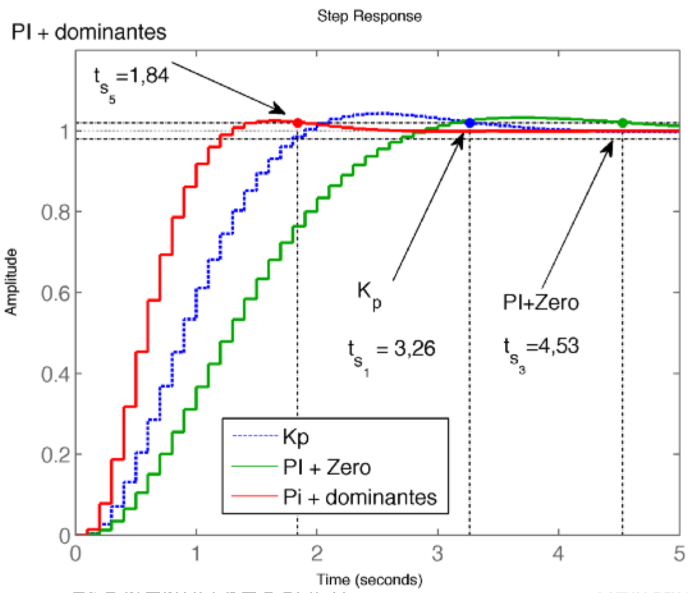


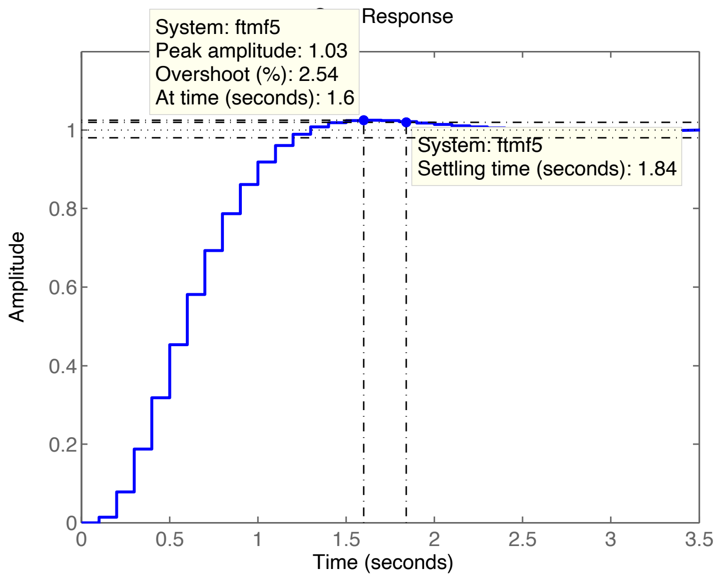


Questão ?
Onde deveria ficar o pólo "extra" de C5(z) ?

Resposta: Calcular contribuição dos ângulos no RL.

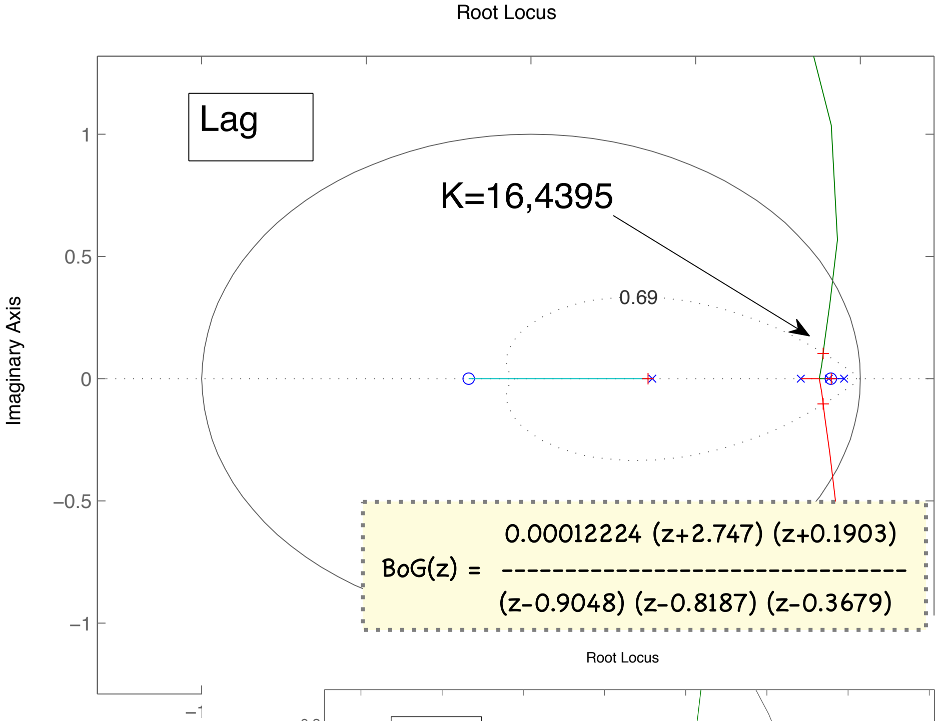
ts = 1,84 segundos.
%OS = 2,54%





6) Controlador por Atraso (Lag)

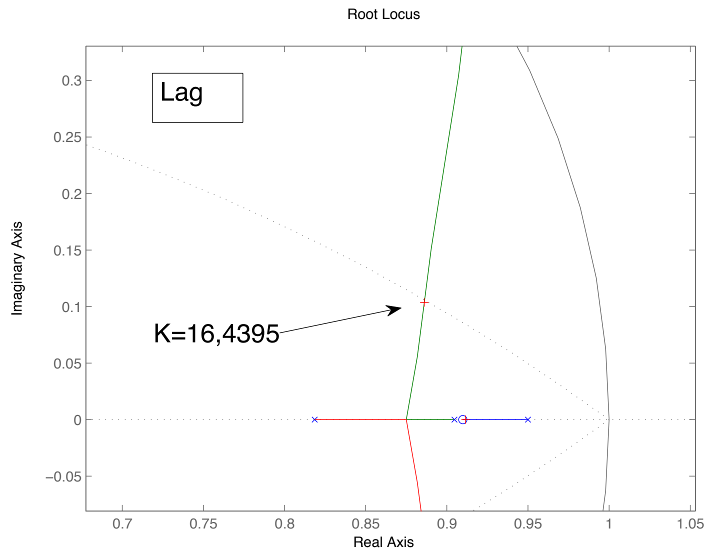
Comentários: localizando zero deste controlador na mesma localização do zero usado no projeto do PI — idéia final é comparar o desempenho deste controlador com os anteriores (PI + Zero, PI + Pólos Dominantes, PI + Lag).



C6(z):

```
 (z-0.91)
---------
 (z-0.95)
```

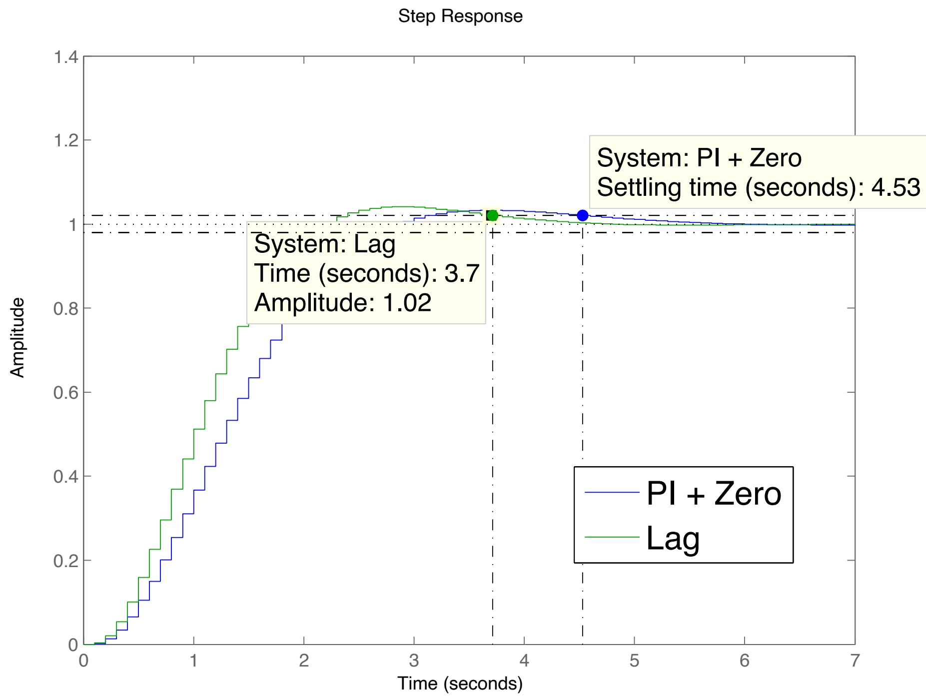
RL resultante:



Resposta resultante:

Detalhes:
>> dcgain(ftmf6)
0.5967

Amplitude do degrau:
>> K6degrau
1.6759



ts_6 = 3,7
%OS = 4,18% (2,8 seg)

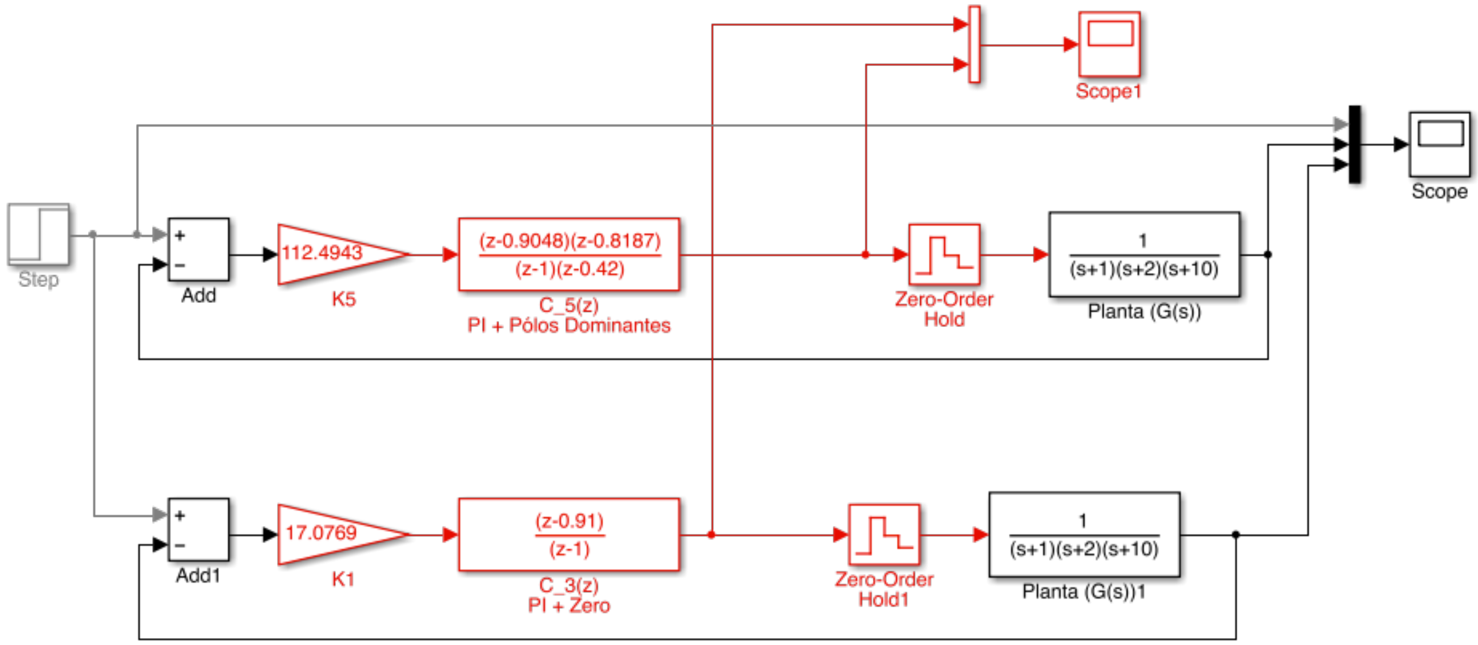
Um pouco + rápido que PI com Zeros (conforme esperado, mas erro nulo em regime permanente?)

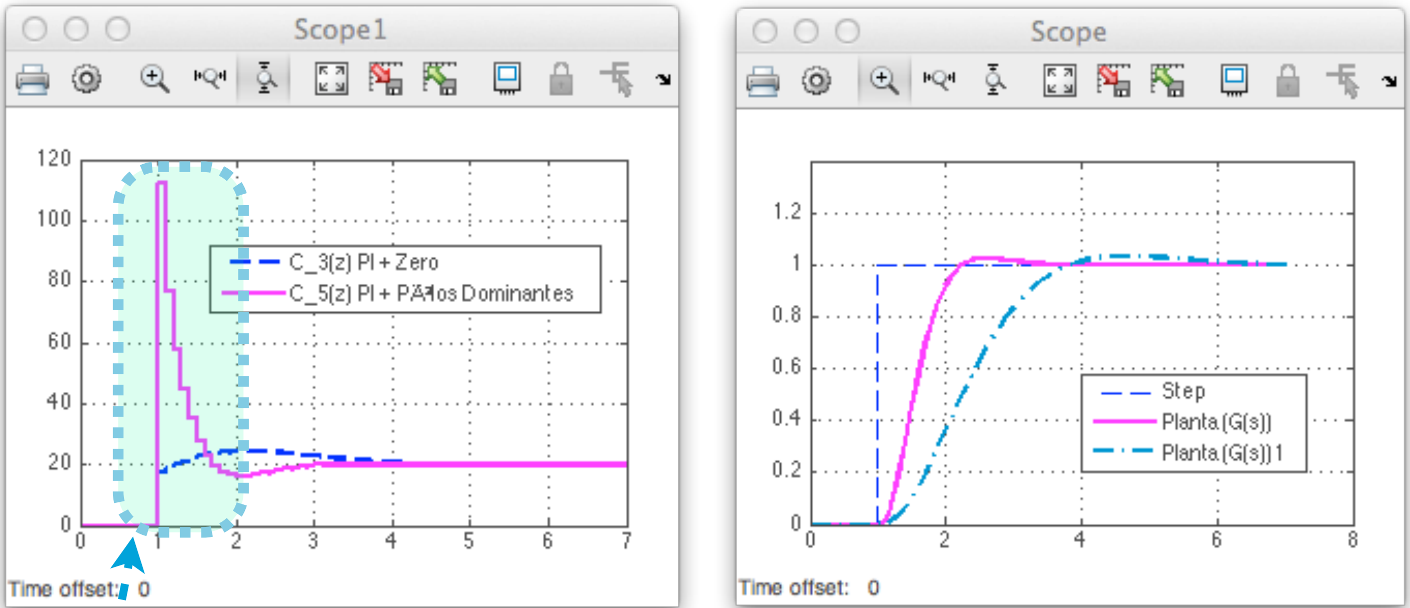
E se o zero do controlador anterior estivesse localizado mais próximo da origem do plano-z ?

“Encerrando" parte dos controladores com ação integral + controlador por atraso (Lag) (integrador aproximado, pólo próximo de z=1), verificando aplicabilidade prática do controlador PI com cancelamento de pólos dominantes:

Comentários:
- altamente provável que ação de controle do PI + pólos dominantes desenvolva amplitudes excessivas, na prática, causando saturação do D/A ou driver de potência na saída do controlador. Comprovando resultados com simulação no Simulink:

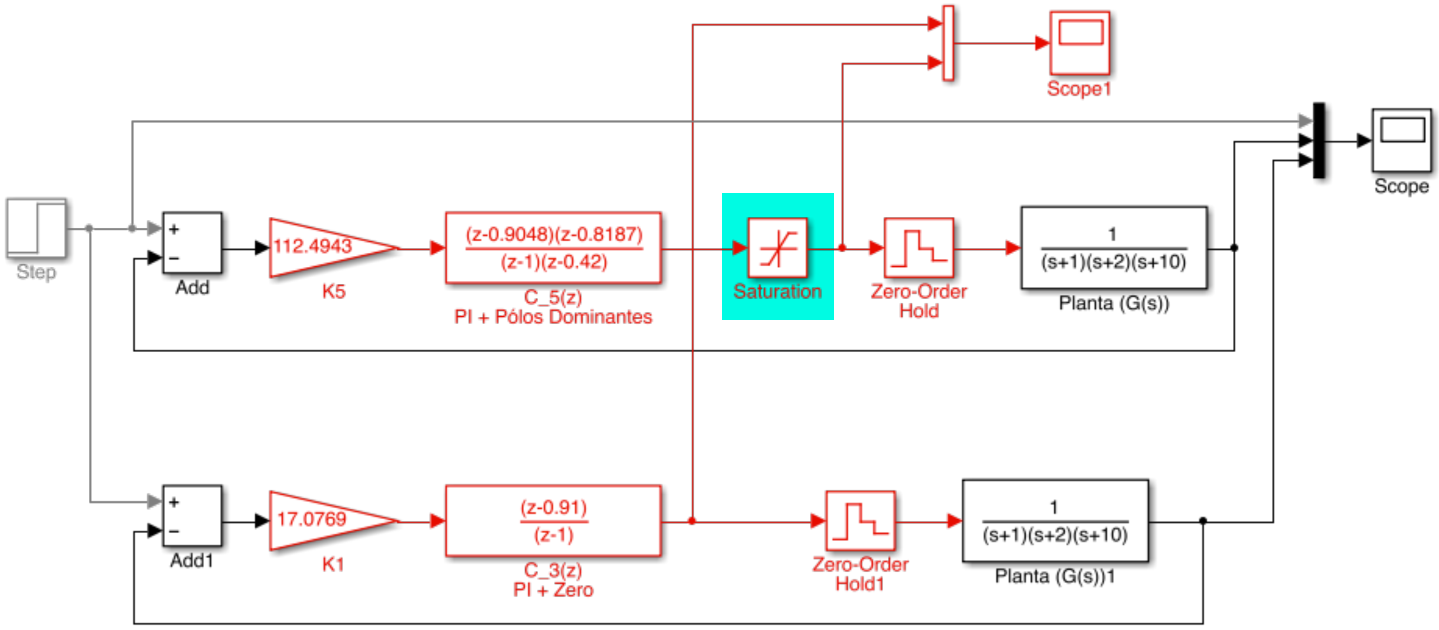
Resultados obtidos:

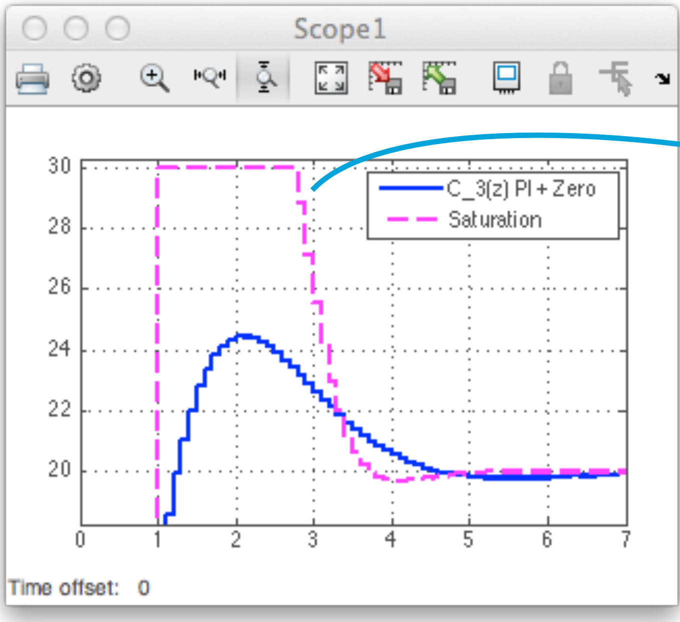




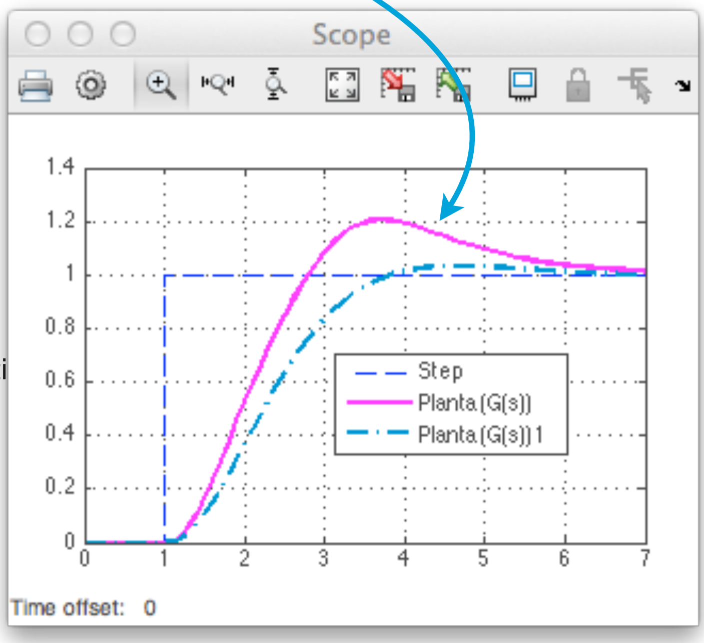
Sinal de controle com amplitudes excessivas (provável saturação do Driver de Potência)

Resultado provável, estipulando saturação do driver de potência em +/- 30,0 (introdução do bloco saturador):





Atraso na resposta



Últi

- Calcular e(∞) = ? (cte ou = 0) ?

Controlador Dead-Beat:

A planta convertida para o formato digital resulta em:

BoG(z)=

0.00012224 (z+2.747) (z+0.1903)

(z-0.9048) (z-0.8187) (z-0.3679)

A idéia do controlador ded-beat é cancelar os pólos e zeros da planta. Mas somente pólos e zeros estáveis, assim C(z) inicialmente fica como:

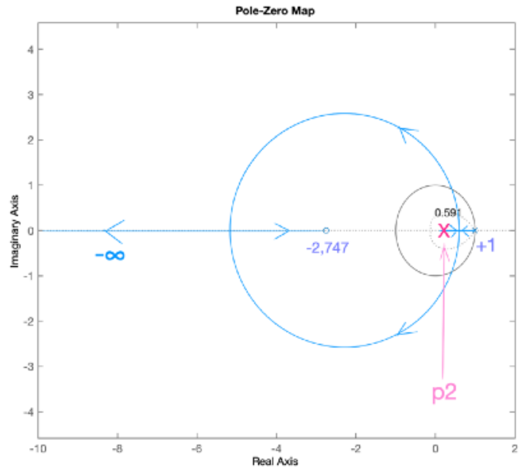
C(z)=

Kc * (z-0.9048) (z-0.8187) (z-0.3679)

(z+0.1903) (z - 1) (z - ??)

$$FTMA(z) = K_c \cdot \frac{0.00012224(z+2.747)\cancel{(z+0.1903)}\cancel{(z-0.9048)}\cancel{(z-0.8187)}\cancel{(z-0.3679)}}{\cancel{(z-0.9048)}\cancel{(z-0.8187)}\cancel{(z-0.3679)}\cancel{(z+0.1903)}(z-1)(z-??)}$$

$$FTMA(z) = K_c \cdot \frac{0.00012224(z+2.747)}{(z-1)(z-p_2)}$$



Notar que como o numerador de C(z) acabou sendo de 3a-ordem, seu denominador deve ser de mesma ordem ou superior. Neste caso, falta definir a posição do pólo extra.

<< Seguem rascunhos de RLs >>

Por fim, uma localização adequada para este terceiro pólo seria em z = -0,5 e o controlador assumiria a forma:

C(z) =

(z-0.9048) (z-0.8187) (z-0.3679)

(z-1) (z+0.5) (z+0.1903)

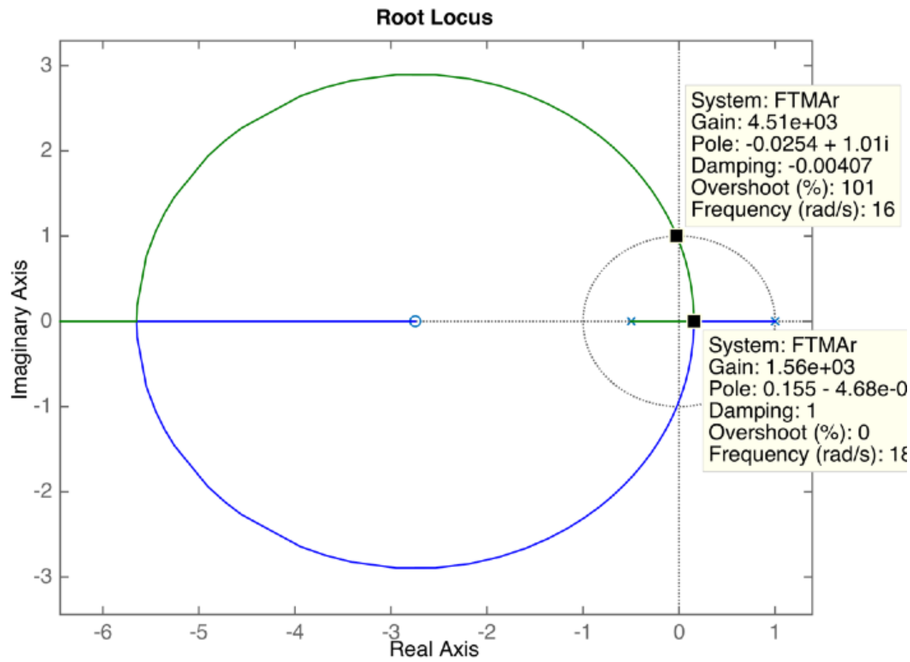
O que resulta na seguinte equação em MA:

FTMA(z) =

0.00012224 (z+2.747)

(z-1) (z+0.5)

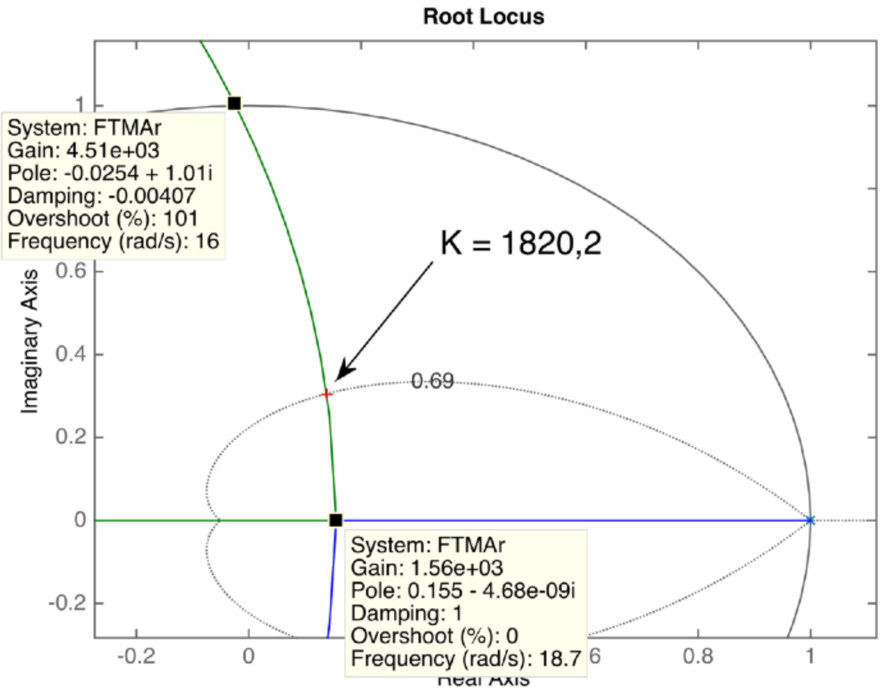
e consequente RL (mostrado na figura ao lado).



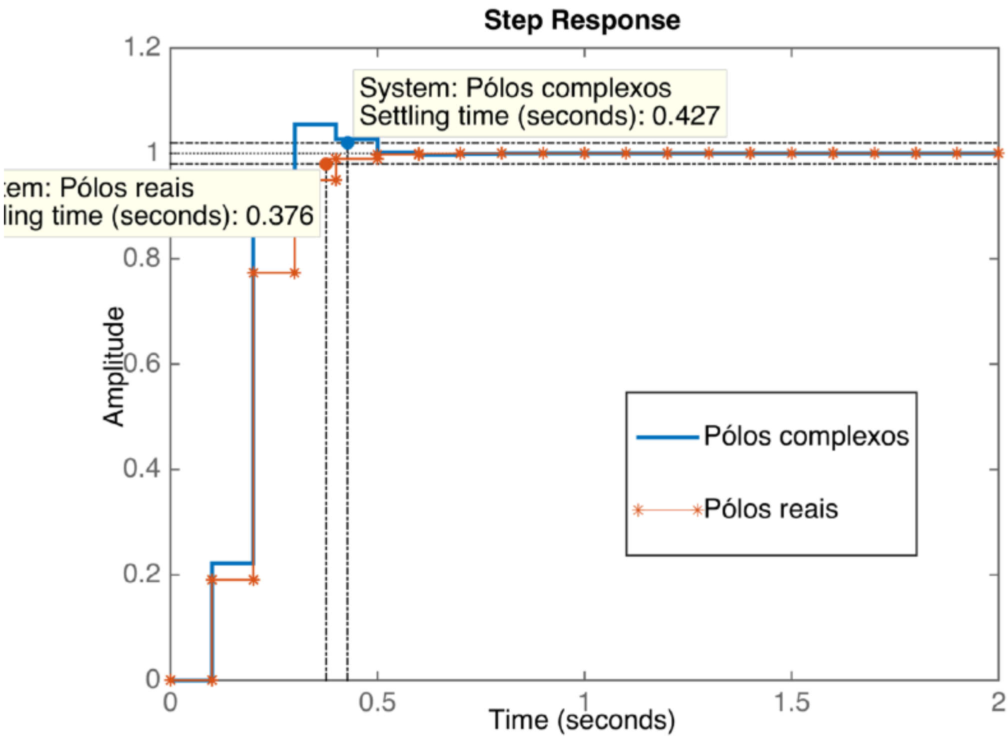
Continua -->

Note que falta definir um ganho. Dois pontos podem ser adotados:

1) pólos reais duplos no ponto de partida do RL (K ≃ 1560), ou;
2) pólos complexos respeitando ζ = 0,6901 (%OS < 5%) e neste caso, K ≃ 1820).

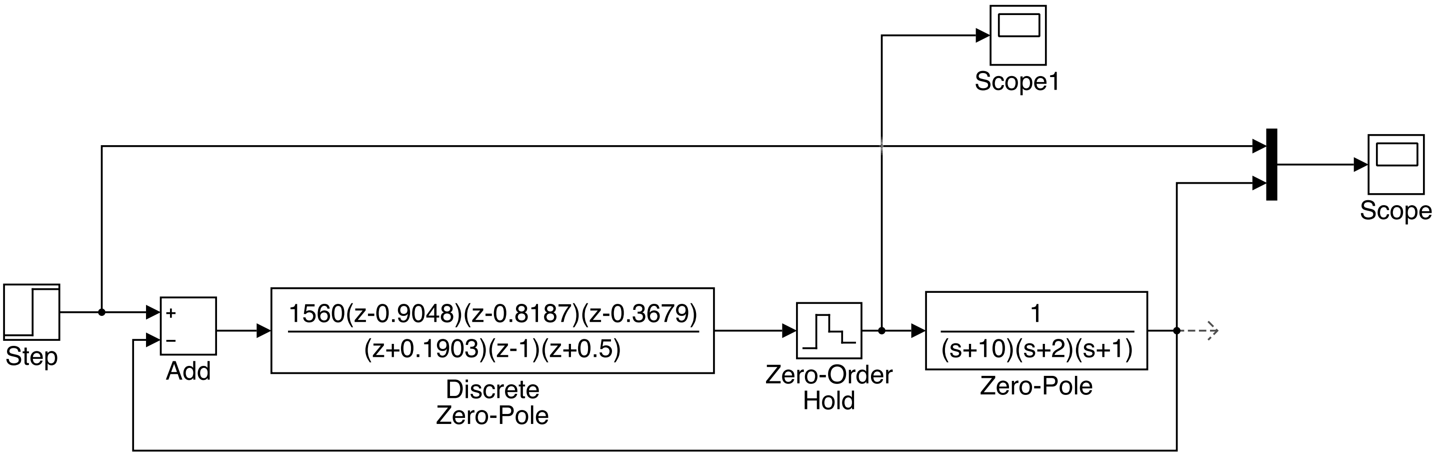


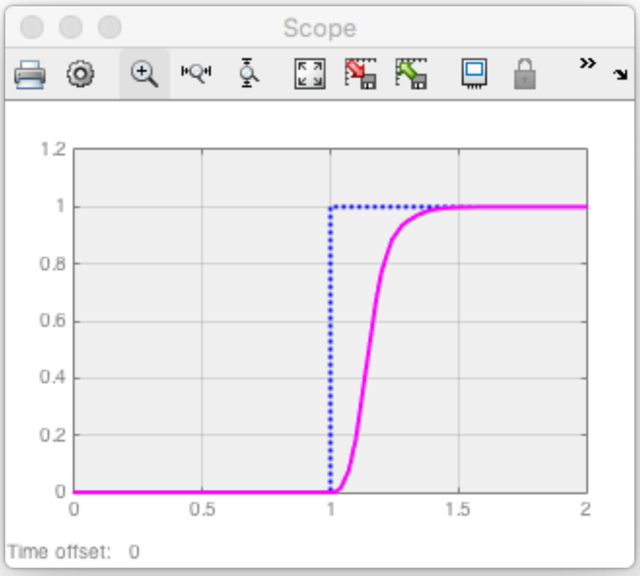
Avaliando as 2 opções, resulta no gráfico mostrado na figura ao lado.



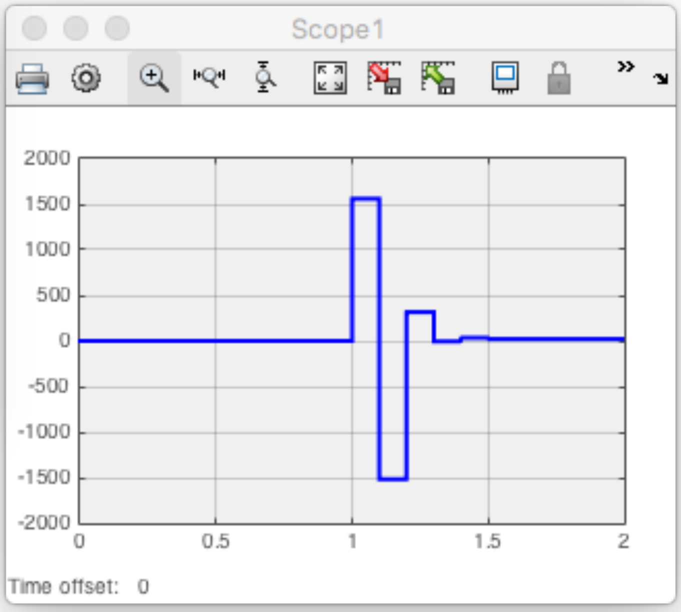
Obs.: O problema não está em se obter um sistema que reage bastante rápido em MF, mas no valor excessivo das amplitudes geradas pelo controlador — seguem simulações realizadas no Simulink:

O sistema simulado usando Simulink resulta em:





Note as elevadas amplitudes desenvolvidas pelo sinal de controle (nos instantes iniciais oscilaram entre mais de +/- 1.500.



Obs.: o gráfico do digrama de blocos (no Simulink) foi exportado do Simulink, usando-se o seguinte comando na janela de comandos do MATLAB:

```
>> print -deps2 -sdead_beat dead_beat_sym.pdf
```

No caso acima, foi gerado um arquivo PDF (vetorizado). Mas outras opções podem ser adotadas como por exemplo: '-dpng' que neste caso, geraria uma figura PNG (resolução melhor que JPG). E notar que '-sXXX' se refere ao diagrama de blocos editado no Simulink com o nome 'XXX'.

7) Controlador por Avanço (de Fase)

Controlador por atraso —> vantagem sobre PD: reduz impacto do ruído frente à diferenciação (o controlador por atraso se aproxima de uma diferenciação).

Desejável tempo de assentamento abaixo de 1,63 segundos, o que significa:

$$t_s = \frac{-\ln\left(0,02\sqrt{1-\zeta^2}\right)}{\zeta\omega_n}$$

para o caso de: $0 < \zeta < 1$, a equação para ts pode ser aproximada para: $t_s = \dfrac{4}{\zeta\omega_n}$

Neste caso, teremos então:
zeta = 0,6901, ts_d = 1,63 e wn resulta então em:

```
>> wn=4/(zeta*ts_d)
wn =    3.5022 (rad/s)
```

ou seja, queremos pólos de MF localizados, no plano-s, na posição: $s = \sigma \pm j\omega_d$
lembrando que:

$$\omega_d = \omega_n \cdot \sqrt{1-\zeta^2}$$

$$\sigma = \omega_n \cdot \zeta$$

temos então:

```
>> sigma=wn*zeta
sigma =    2.4169
>> wd=wn*sqrt(1-zeta^2)
wd =    2.5346
```

Mas como não estamos projetando controlador no plano-s e sim no plano-z, faz-se necessário sua translação do plano-s para o plano-z usando a definição da transformada-Z:

$$z = e^{-Ts}$$

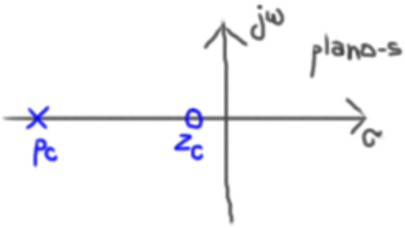
os pólos dominantes de MF deveriam passar por:

```
>> polos_MFs=sigma+i*wd
polos_MFs =    2.4169 + 2.5346i
>> polos_MFz=exp(-T*polos_MFs)
polos_MFz =    0.7602 - 0.1969i
```

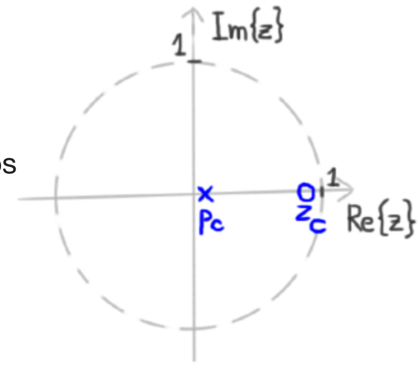
Temos agora que recordar o formato de um controlador de avanço que é caracterizado pela equação:

$$C(z) = \frac{s+z_c}{s+p_c}$$

onde no plano-s: $p_c \to -\infty$ e seu zero fica próximo da origem do plano-s (aprox. de efeito derivativo).



Mas no plano-z, ficaria com seu zero próximo do círculo unitário e seu pólo bastante próximo da origem do plano-z.

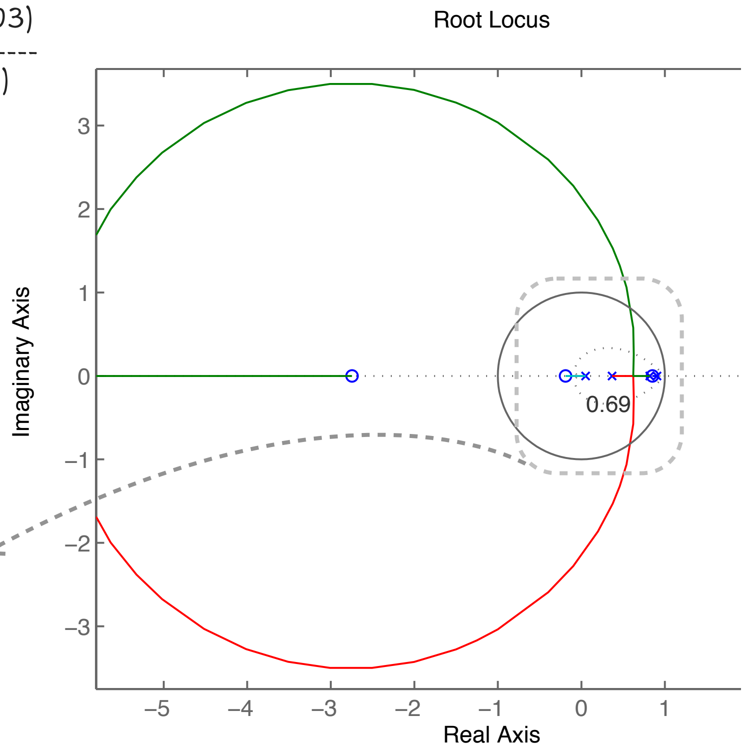


Num primeiro momento, sem se importar com a melhor posição para os pólos e zeros do controlador em relação à planta, podemos fazer:

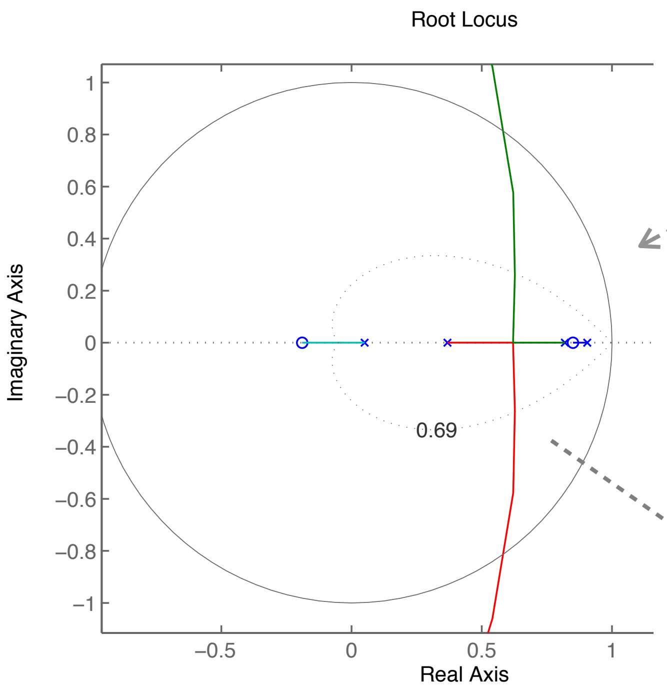
$$C7(z) = \frac{(z-0.85)}{(z-0.05)}$$

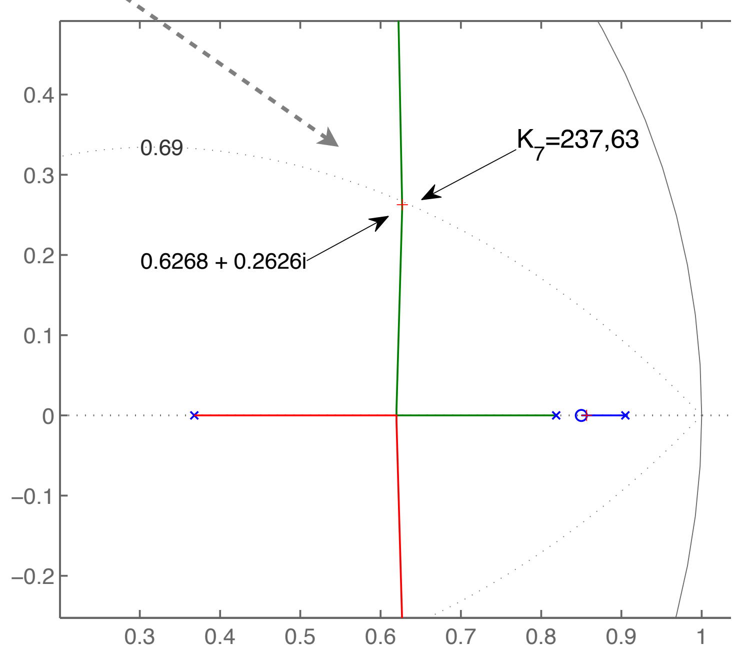
O que resultaria numa FTMA(z):

$$\frac{0.00012224\,(z+2.747)\,(z-0.85)\,(z+0.1903)}{(z-0.9048)\,(z-0.8187)\,(z-0.3679)\,(z-0.05)}$$



e correspondente RL como:



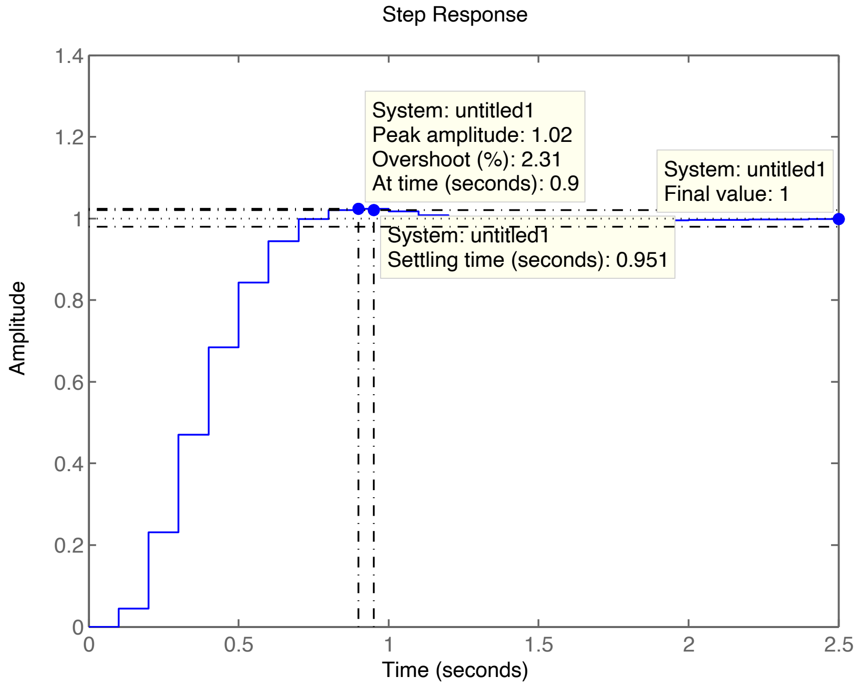


Com
K7 = 237.6297

e pólos MF dominantes em:
0.6268 + 0.2626i

Notar que o desejado era:
polos_MFz = 0.7602 - 0.1969i

O Controlador anterior resulta na resposta:



Notando que:
>> dcgain(ftmf7)
ans = 0.6523

ou seja, que para fazer y(∞) = 1,0; deve ser aplicado um degrau de amplitude igual à:

>> K7degrau=1/ans
K7degrau = 1.5330

resultando num ts = 0,951
o ts_d = 1,63.
E %OS = 2,31%

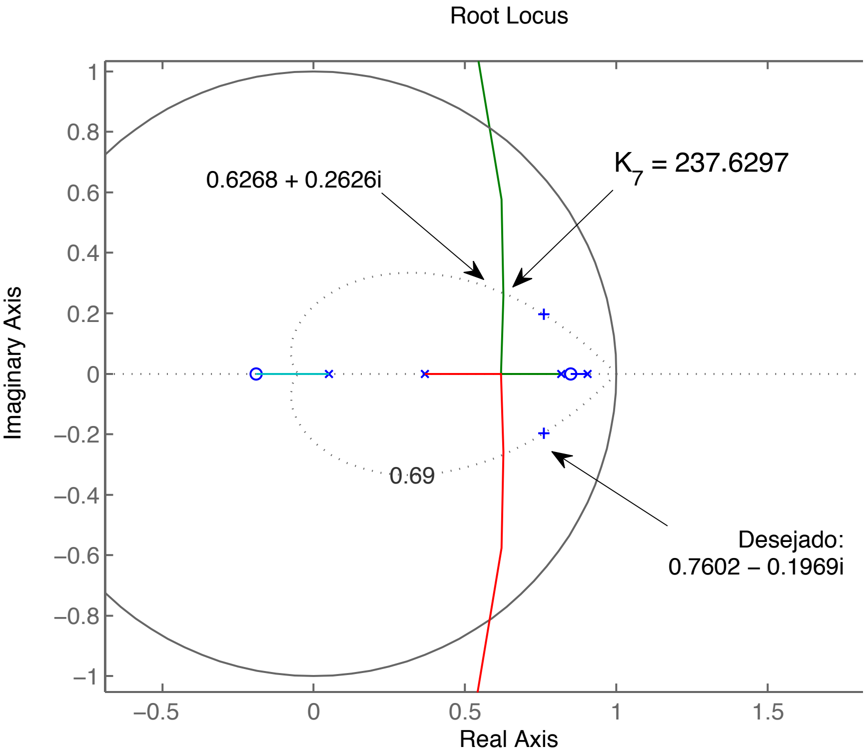
=> Verificar se ação de controle não desenvolve amplitudes elevadas demais ?

Com
K7 = 237.6297
e pólos MF dominantes em:
0.6268 + 0.2626i

Notar que o desejado era:
polos_MFz =

0.7602 - 0.1969i

Isto pode ser “melhorado" calculando a contribuição dos ângulos, arbitrando a localização inicial para o zero do controlador próximo da origem do plano-z.



Como a planta possui uma raiz bastante próxima da origem do plano-z, um pólo em z = 0,3679, podemos arbitrar uma posição do pólo do controlador em z = 0,15 (este pólo será atraído pelo zero da planta em z = -0,19 e o pólo da planta em z = 0,3679 "caminhará" na direção do outro pólo da planta em z = 0,8187). Esta nova composição resultaria no seguinte RL:

Calculando as contribuições dos ângulos (script “`angulos.m`”), obtemos:

```
>> numc7=1;
>> denc7=[1 -0.15];
>> C7=tf(numc7,denc7,T);
>> zpk(C7)
ans =
     1
  --------
  (z-0.15)
>> ftma7=C7*BoG;
>> [num,den,aux]=tfdata(ftma7,'v');
>> angulos

open_poles =
    0.9048
    0.8187
    0.3679
    0.1500

open_zeros =
   -2.7471
   -0.1903

New settiling time (desired): ? 1.655
Natural damping osc. frequency, wn = 3.5022
(rad/s)
```

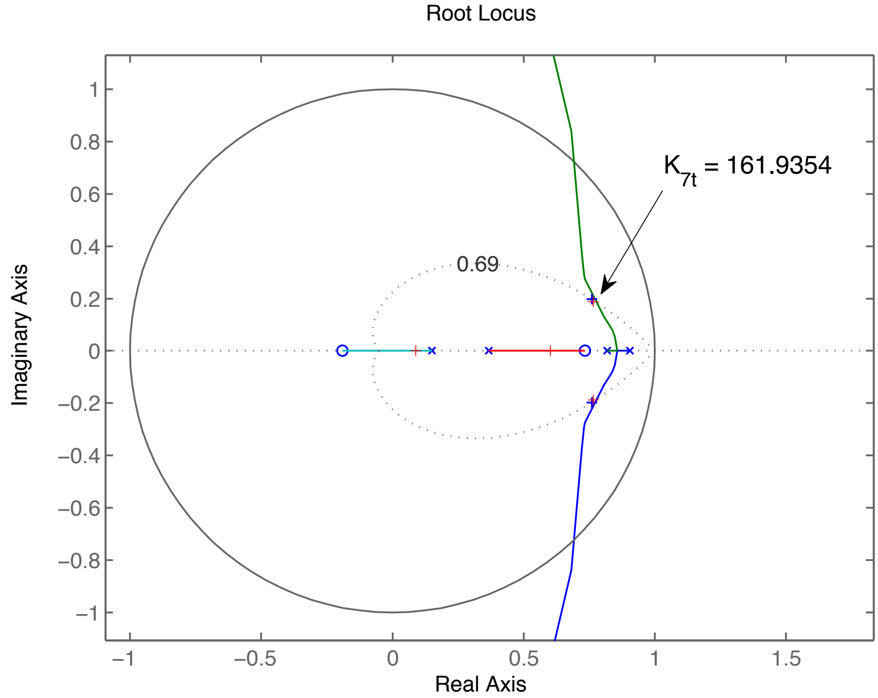
```
Localização do pólo de MF no plano-s:
s = 2.4169 +/- j2.5346
Localização do pólo de MF no plano-z:
z = 0.7602 +/- j0.1969
Angle Contribution of each pole of the open
loop system
 p1 = 0.9048 --> 126.30^o
 p2 = 0.8187 --> 106.55^o
 p3 = 0.3679 --> 26.65^o
 p4 = 0.1500 --> 17.89^o
Sum of angular poles positions: 277.39^o
Angle Contribution of each zero of the open
loop system
 z1 = -2.7471 --> 3.21^o
 z2 = -0.1903 --> 11.70^o
Sum of angular zeros positions: 14.92^o
Final Resulting angle for the extra Lead pole/
zero: 82.4679^o
Final position for the extra Lead zero:
0.7342
>>
>> pc
pc =    0.7342
>>
```

Testando:

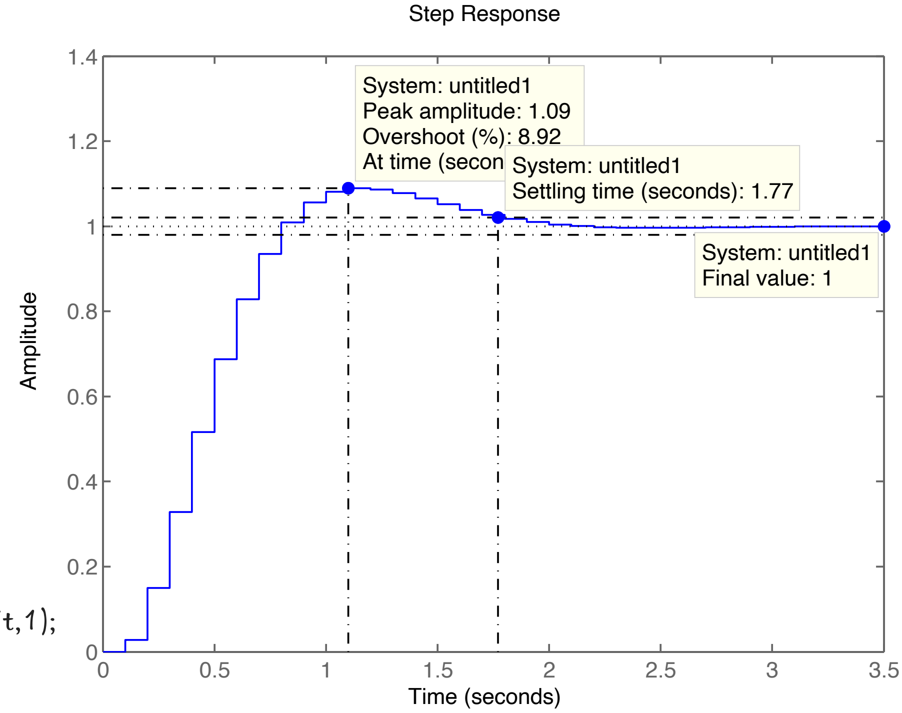
```
>> numc7t=[1 -pc];
>> denc7t=denc7;
>> C7t=tf(numc7t,denc7t,T);
>> zpk(C7t)

ans =

  (z-0.7342)
  ----------
   (z-0.15)

Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.

>> ftma7t=C7t*BoG;
>> figure; rlocus(ftma7t)
```

```
>> hold on
>> zgrid(zeta,wn/T)
>> polos_MFz

polos_MFz =
   0.7602 - 0.1969i

>> plot([polos_MFz 0.7602+0.1969i],'b+') %
sobrepõe no RL os pólos de MF desejados!
>> K7t=rlocfind(ftma7t)
Select a point in the graphics window

selected_point =
   0.7686 + 0.1895i

K7t =  161.9354
>>
```

O que resulta no seguinte RL mostrada na figura a seguir:



E na resposta temporal mostrada a seguir:



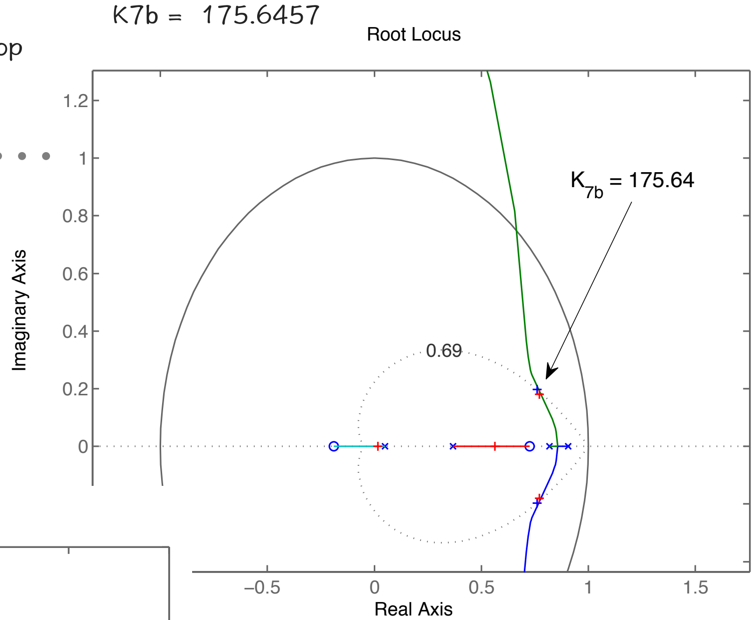

```
>> ftmf7t=feedback(K7t*ftma7t,1);
>> dcgain(ftmf7t)
ans =     0.7169
```

Obs.: Necessário aplicar degrau de amplitude igual à: 1,3949

```
>> K7degraut=1/ans
K7degraut =     1.3949

>> >> figure; step(K7degraut*ftmf7t)
```

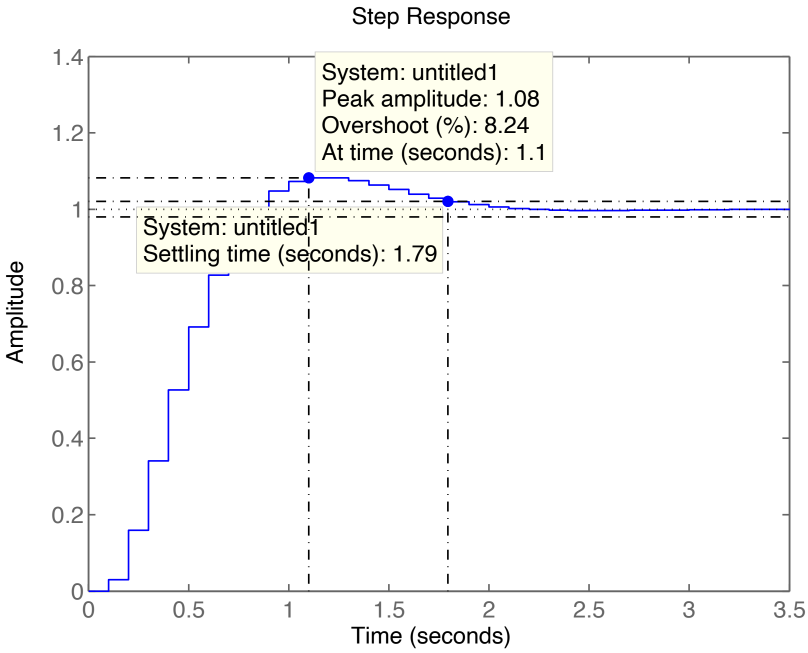
Se o pólo do Controlador foi colocado em z = 0,05; teremos:

```
>> angulos
open_poles =
    0.9048
    0.8187
    0.3679
    0.0500

open_zeros =
   -2.7471
   -0.1903

New settiling time (desired): ? 1.6550
Natural damping osc. frequency, wn = 3.5022
(rad/s)
Localização do pólo de MF no plano-s:
s = 2.4169 +/- j2.5346
Localização do pólo de MF no plano-z:
z = 0.7602 +/- j0.1969
Angle Contribution of each pole of the open loop
system
 p1 = 0.9048 --> 126.30^o
 p2 = 0.8187 --> 106.55^o
 p3 = 0.3679 --> 26.65^o
 p4 = 0.0500 --> 15.50^o
Sum of angular poles positions: 275.00^o
Angle Contribution of each zero of the open loop
system
 z1 = -2.7471 --> 3.21^o
 z2 = -0.1903 --> 11.70^o
Sum of angular zeros positions: 14.92^o
Final Resulting angle for the extra Lead pole/zero:
80.0796^o
Are you enter the [p]ole or [z]ero of C(z): ? z
Ok, Evaluating the final position for the extra
Lead pole:
Final position for the extra Lead pole: 0.7258
>> numc7b=[1 -0.7258];
>> denc7b=denc7t;
>> C7b=tf(numc7b,denc7b,T);
>> zpk(C7b)
ans =
  (z-0.7258)
  ----------
   (z-0.05)

>> K7b=rlocfind(ftma7b)
Select a point in the graphics window

selected_point =
   0.7745 + 0.1828i

K7b =  175.6457
```



Resultado final:

```
>> ftmf7b=feedback(K7b*ftma7b,1);
>> dcgain(ftmf7b)
ans =     0.7171

>> K7bdegrau=1/ans
K7bdegrau =     1.3945
```



Continua...

Projeto de Controlador de Avanço-Atraso usando transformação bilinear.

Note que neste caso, o controlador é projetado inicialmente no plano-s, usando Diagramas de Bode e depois é convertido para seu formato digital usando o método de Tustin.

Não é objetivo deste trabalho, avaliar novos métodos de projetos de controladores de avanço-atraso no plano-s usando diagrama de Bode. Neste caso, será usado um *script* organizado previamente no MATLAB que permite automatizar o projeto deste tipo de controlador.

A planta é dada por:

```
                  1
G(s) = ------------------
       (s+10) (s+2) (s+1)
```

Note: é um sistema do tipo 0 (sem integradores), então exibe erro não nulo para entrada degrau. A fim de restringir (limitar) a quantidade de incógnitas para determinação dos parâmetros do controlador lead-lag, definimos um valor máximo tolerável de erro em regime permanente e então definimos a constante de erro de posição, Kp. Neste caso:

$$e_{step}(\infty) = \lim_{s\to 0} sE(s) = \lim_{s\to 0} \frac{s(1/s)}{1+G(s)} = \frac{1}{1+\lim_{s\to 0} G(s)} = \frac{1}{1+K_p}$$

Se erro_degrau(∞)=5%; como y(∞)=1,0; implica erro_degrau(∞)=0.05.

Kp original da planta: Kp = 1/(10*2*1) = 0.0500 (curiosidade).

Mas Kp para sistema FTMA(s) ficaria em:

$$K_p = \frac{1-e}{e}$$

ou:

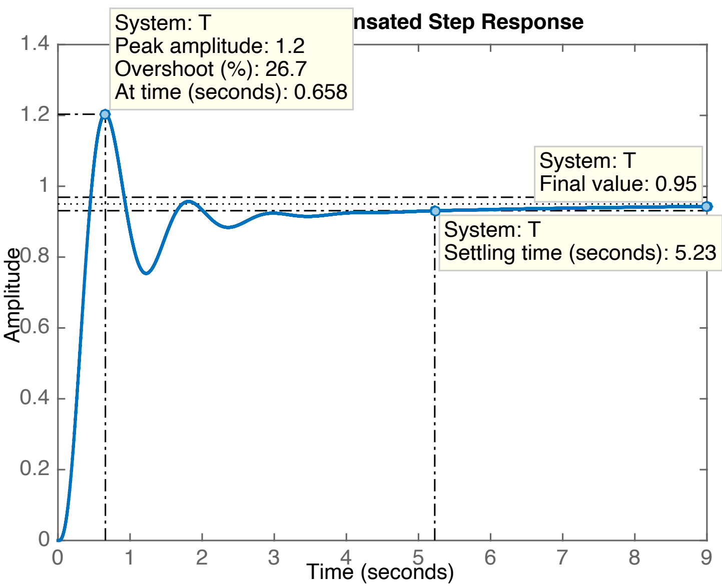
```
>> Kp=(1-0.05)/0.05
Kp =    19
```

Rodando então o script: “*bode_lag_lead.m*” obtêm-se os seguintes resultados:

```
>> bode_lag_lead
Input %OS: ? 5
Input peak time, Tp: ? 1.2
Type value of Kp/v/a (static error) : ? 19
Value of K to consider desired Kv, K =
K =   380
G(s) considering Kv (e->infy):

G =
        380
  ------------------
  (s+10) (s+2) (s+1)
 Continuous-time zero/pole/gain model.

Calculating damping ratio for the required %OS, zeta:
z =    0.6901

Determining Phase margin required based on zeta (and %OS), Pm:
Pmreq =    64.6253

Based on previous zeta, determining the natural frequency:
wn =    3.6175
```

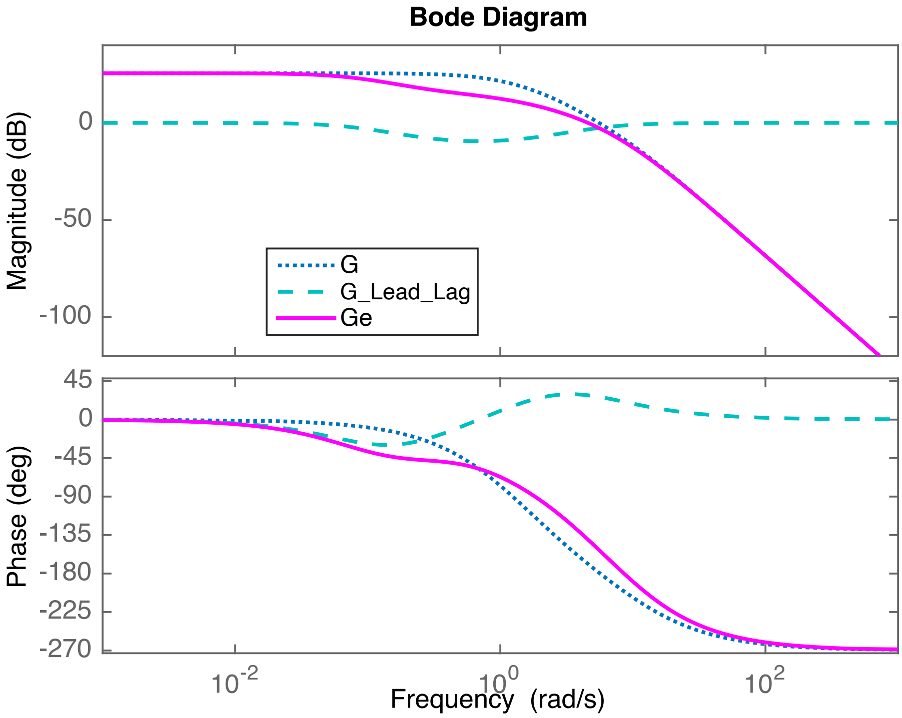
```
Required bandwidth:
wBW =    3.7044

Determining a break frecuency for lead near wBW:
wpm =    2.9635

Phase for the new w_BW, wpm):
P = -143.8466

Gain for new w_BW, wpm):
M =    3.2582

Considering the new PM_req with Lead compensator:
Pmreqc =   33.4719

Determining Beta for the Lead compensator:
beta =    0.2891

Determining the zero position of the Lag compensator:
zclag =    0.2964

Determining the pole position of the Lag compensator:
pclag =    0.0857

Determining the K of the Lag compensator (0 db at low frequencies):
Kclag =    0.2891

Lag compensator, Glag(s):
Glag =
  0.28905 (s+0.2964)
  ------------------
     (s+0.08566)
Continuous-time zero/pole/gain model.

ans = Lead compensator

Glead =
  3.4596 (s+1.593)
  ----------------
     (s+5.512)
Continuous-time zero/pole/gain model.


ans = Lag-Lead Compensated Ge(s)
Ge =
         380 (s+0.2964) (s+1.593)
  ----------------------------------------
  (s+10) (s+5.512) (s+2) (s+1) (s+0.08566)
Continuous-time zero/pole/gain model.

Kv =     0
>>
```

Que permite alcançar a seguinte resposta para entrada degrau:



Resultados gráficos:
%OS = 26,7%
tp = 0,658
ts = 5,23
Erro em regime permanente: 0.95

Por curiosidade, o diagrama de Bode relacionado com o projeto segue na figura ao lado.

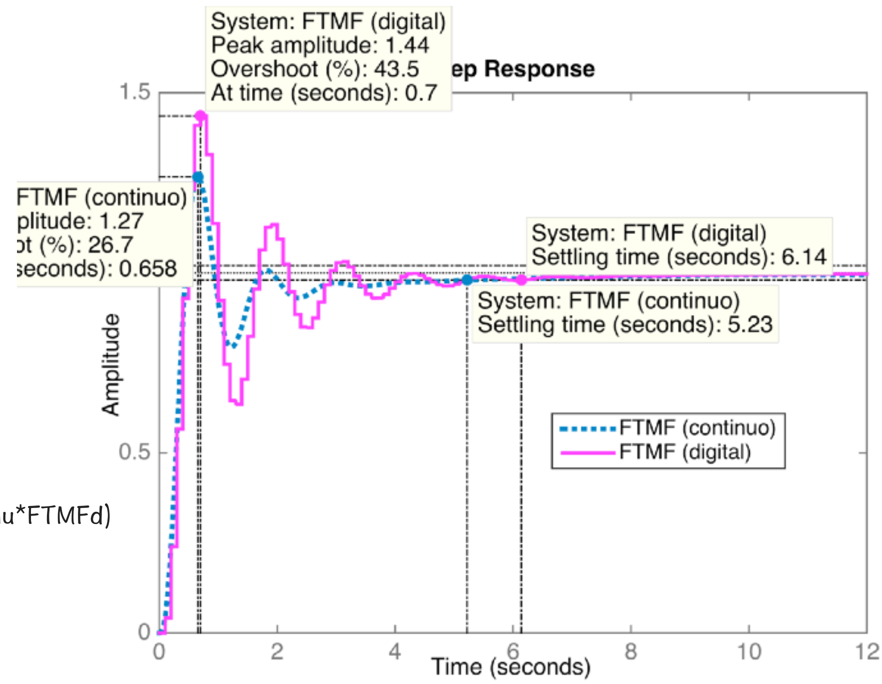


Mas estamos interessados na versão digital do controlador. Então necessitamos realizar uma transformação bilinear do plano-s para o plano-z usando neste caso o método de Tustin:

```
>> G_Lead_Lag=K*Glag*Glead;
>> zpk(G_Lead_Lag)
ans =
  380 (s+0.2964) (s+1.593)
  ------------------------
   (s+0.08566) (s+5.512)
Continuous-time zero/pole/gain model.
>> T=0.1;
>> G_lead_lag_d=c2d(G_Lead_Lag,T,'tustin');
>> zpk(G_lead_lag_d)

ans =
  325 (z-0.9708) (z-0.8524)
  -------------------------
    (z-0.9915) (z-0.5679)
Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.
>>
```

Simulando, resulta em:

```
>> FTMAd=G_lead_lag_d*BoG;
>> FTMFd=feedback(FTMAd,1);
>> dcgain(FTMFd)
ans =     0.9500

>> K_degrau=1/ans;
>> FTMF=feedback(Ge,1);
>> figure; step(K_degrau*FTMF,K_degrau*FTMFd)
>>
```



Projeto de Controlador PID

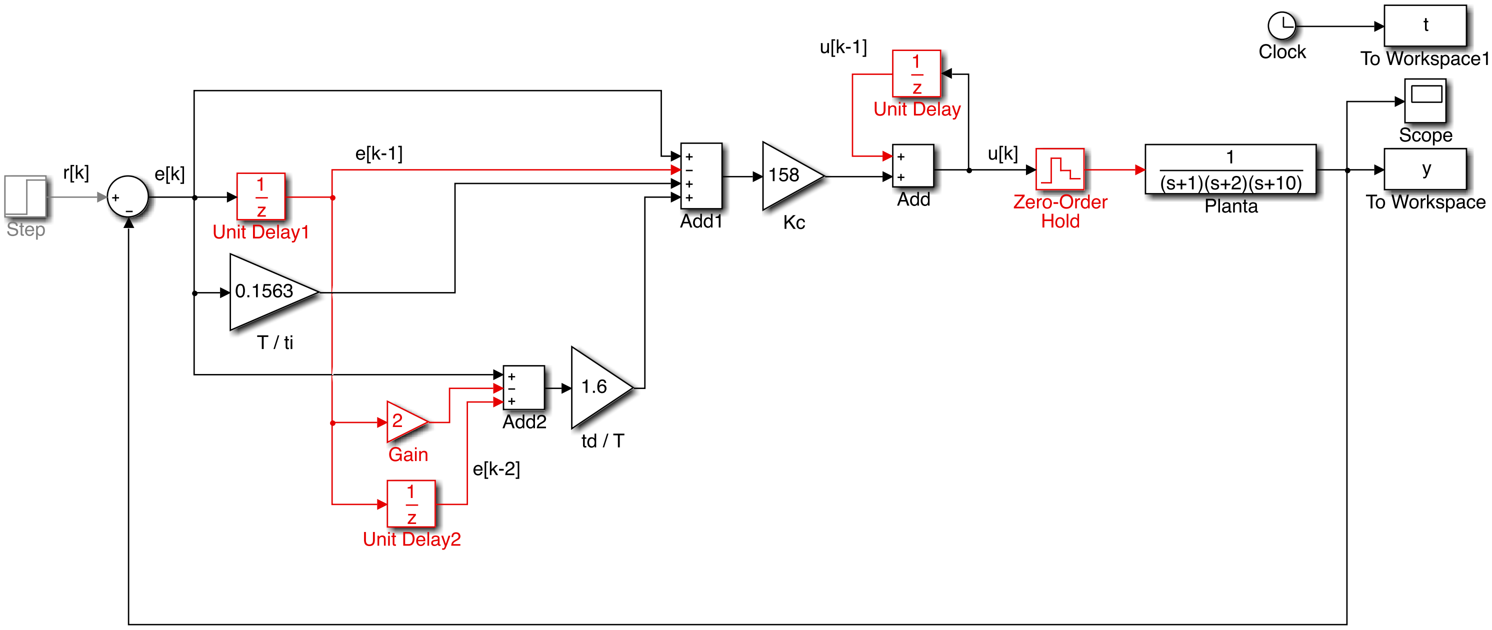
A planta é dada por:

$$G(s) = \frac{1}{(s+1)(s+2)(s+10)}$$

e que seja usado período de amostragem, T = 0,1 segundos.

$$u[k] = u[k-1] + K_c\left[(e[k] - e[k-1]) + \frac{T}{\tau_i}e[k] + \frac{\tau_d}{T}(e[k] - 2e[k-1] + e[k-2])\right]$$

A equação (de diferenças) do PID no formato de velocidade é dado por:



Este conjunto introduzido no Matlab/Simulink fica como:

<Arquivo: planta_PID_velocity.slx >

Note que antes de tentar simular este sistema, é necessário se realizar sua sintonia. Para tanto, devemos determinar ***Ku*** (ultimate gain) e ***Tu*** (período da oscilação). Isto pode ser obtido usando-se o método de Yuri, ou de forma mais simples (mas mais imprecisa), traçando o gráfico do lugar das raízes e simulando o sistema para capturar *Tu*.

```
>> num=1;
>> den=poly([-1 -2 -10]);
>> G=tf(num,den);
>> T=0.1;
>> BoG=c2d(G, T);
>> zpk(BoG)
  0.00012224 (z+2.747) (z+0.1903)
  --------------------------------
  (z-0.9048) (z-0.8187) (z-0.3679)
>> rlocus(BoG);
```

Que resulta no gráfico mostrado na próxima página.

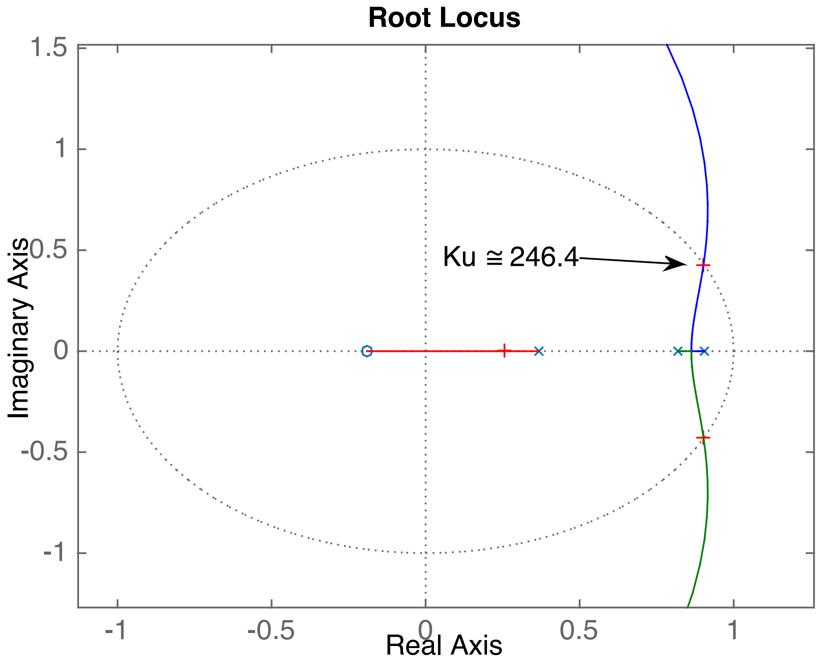
RL de BoG(z):

```
>> Ku=rlocfind(BoG)
```

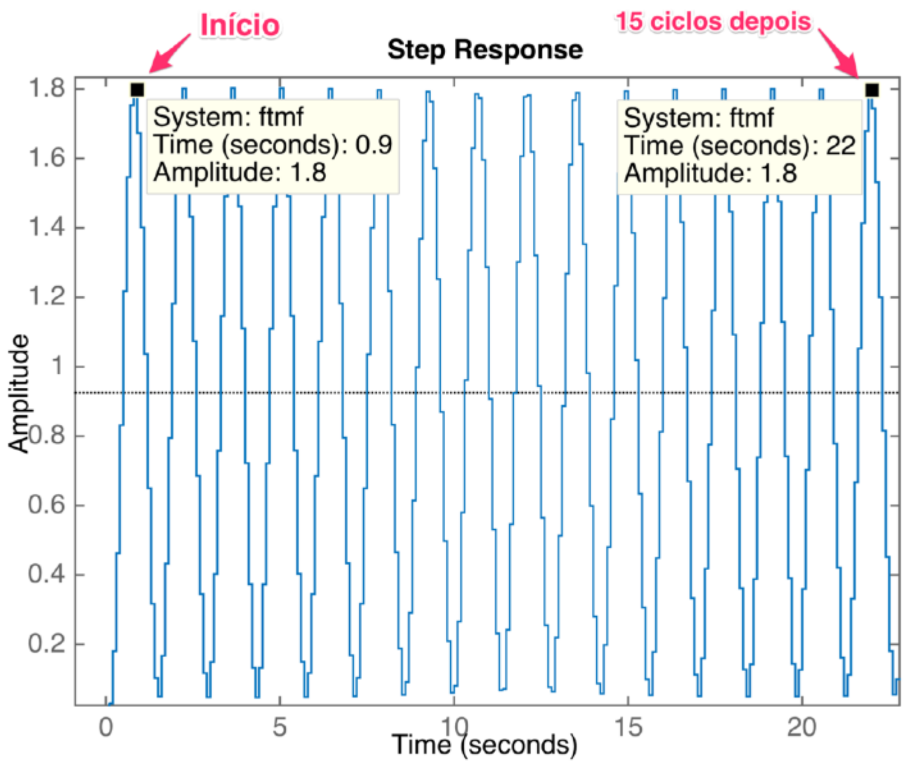
Após algumas tentativas, determina-se que Ku é aproximadamente:

Ku = 246.4000

Fechando-se uma malha de controle usando este valor para o ganho proporcional verificamos o sistema oscilando para uma entrada degrau:




```
>> ftmf=feedback(Ku*BoG,1);
>> figure; step(ftmf)

>> Tu = (22-0.9)/15
>> Tu =     1.4067
```

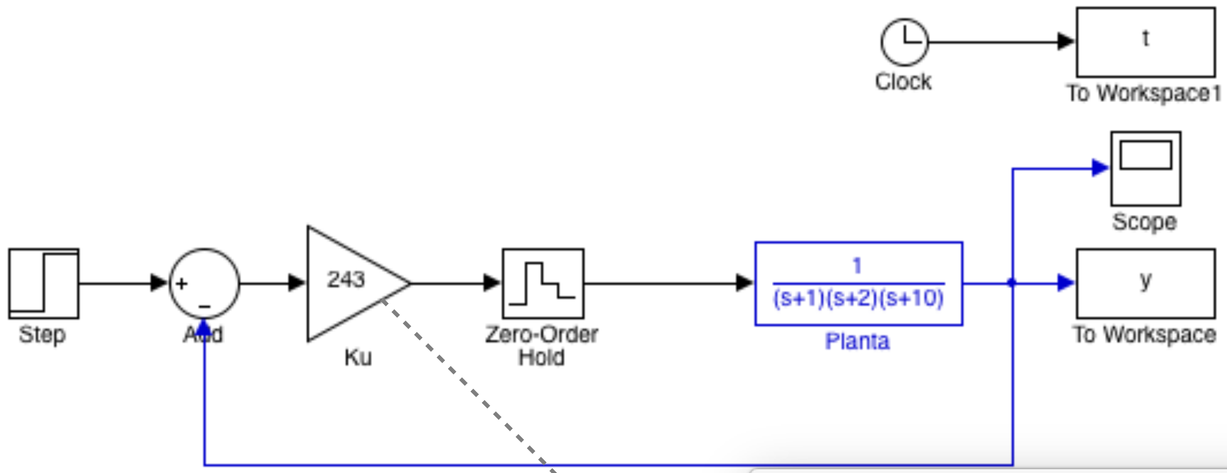


Mas uma simulado usando Matlab/Simulink resulta em:

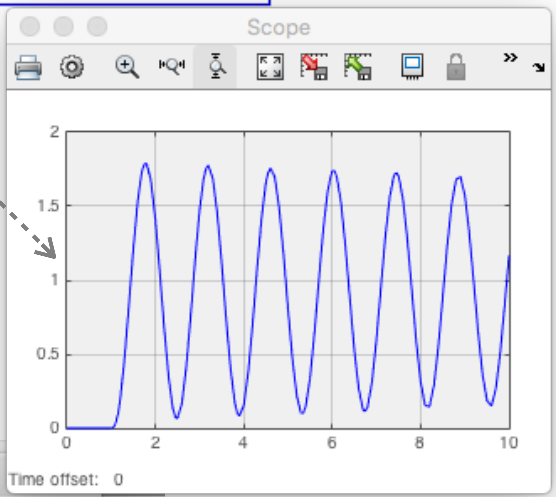
Segue na próxima página diagrama em blocos no Simulink: <Arquivo: planta_P_Ku.slx >

Arquivo: planta_P_Ku.slx:

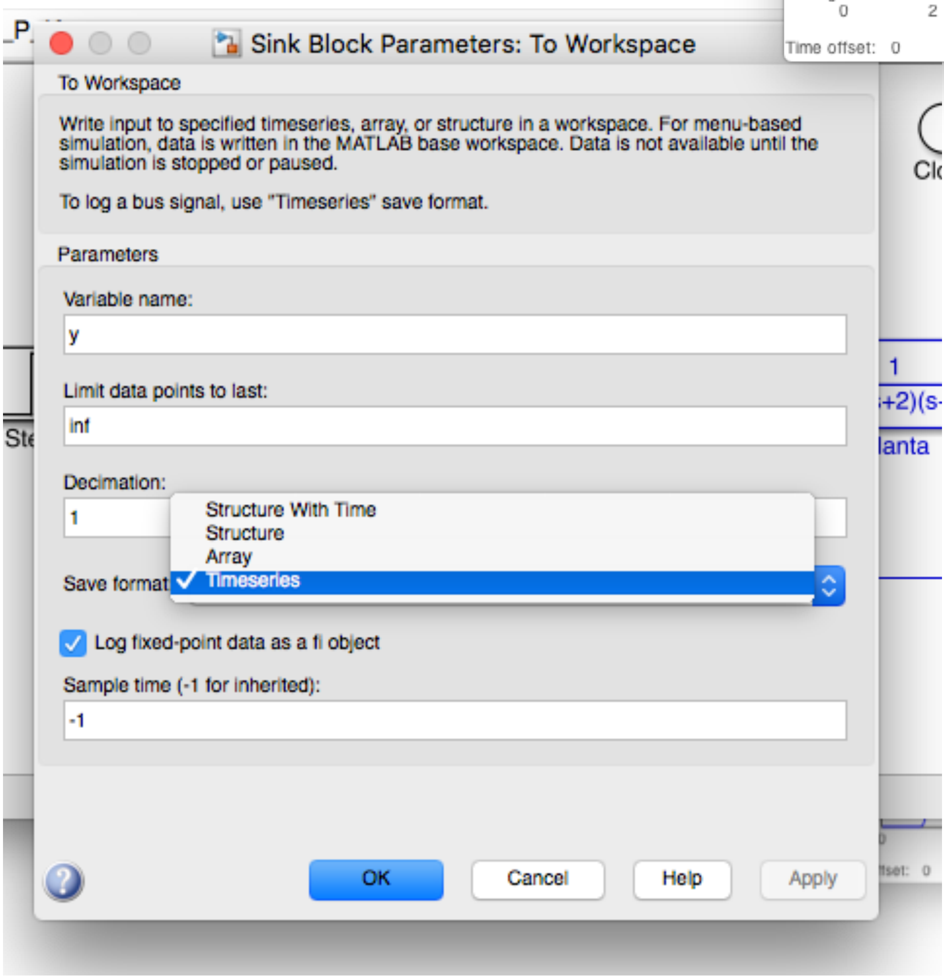
```
>> open planta_P_Ku % abre diagrama em blocos no Simulink
```



Notar que o bloco “Scope" pode ser útil para rapidamente visualizar o resultado de uma simulação, mas não permite coletar dados para determinar o período de oscilação, motivo pelo qual, estão sendo usados os blocos “To Workspace”.



Mas O bloco “To Workspace” permite ser ajustado em 1 de 4 opções diferentes para o tipo de dado sendo guardado:



<-- Detalhe para o formato de dados selecionado para o bloco “**To Worksapce**”:

Se for usado o formato (novo) “Timeseries”, um objeto específico contendo dados é gerado no Matlab, neste caso:

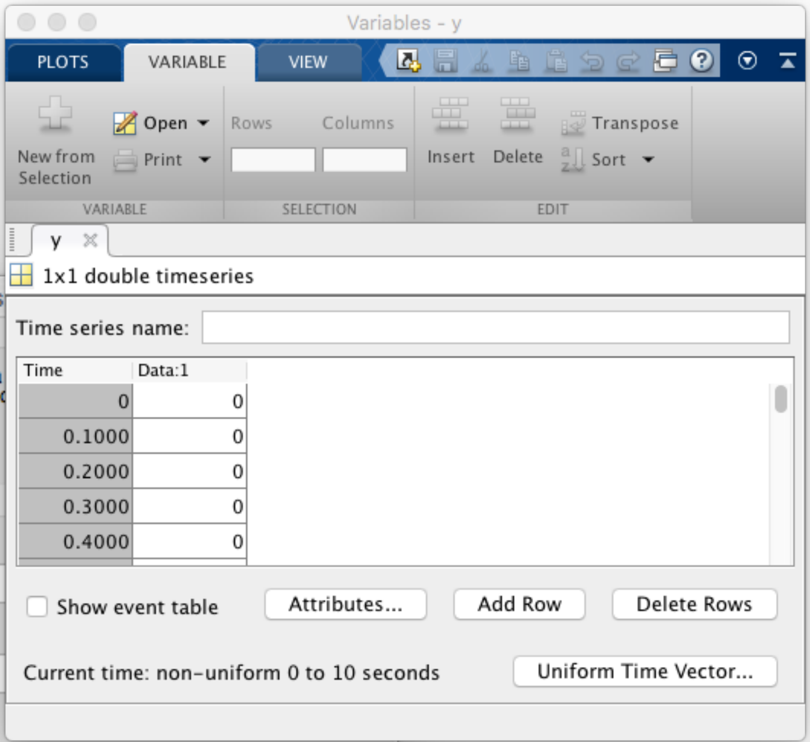
```
>> whos y
  Name      Size            Bytes  Class         Attributes
  y         1x1             2964  timeseries
>> y
  timeseries
  Common Properties:
            Name: ''
            Time: [131x1 double]
        TimeInfo: [1x1 tsdata.timemetadata]
            Data: [131x1 double]
        DataInfo: [1x1 tsdata.datametadata]

  More properties, Methods
>>
```

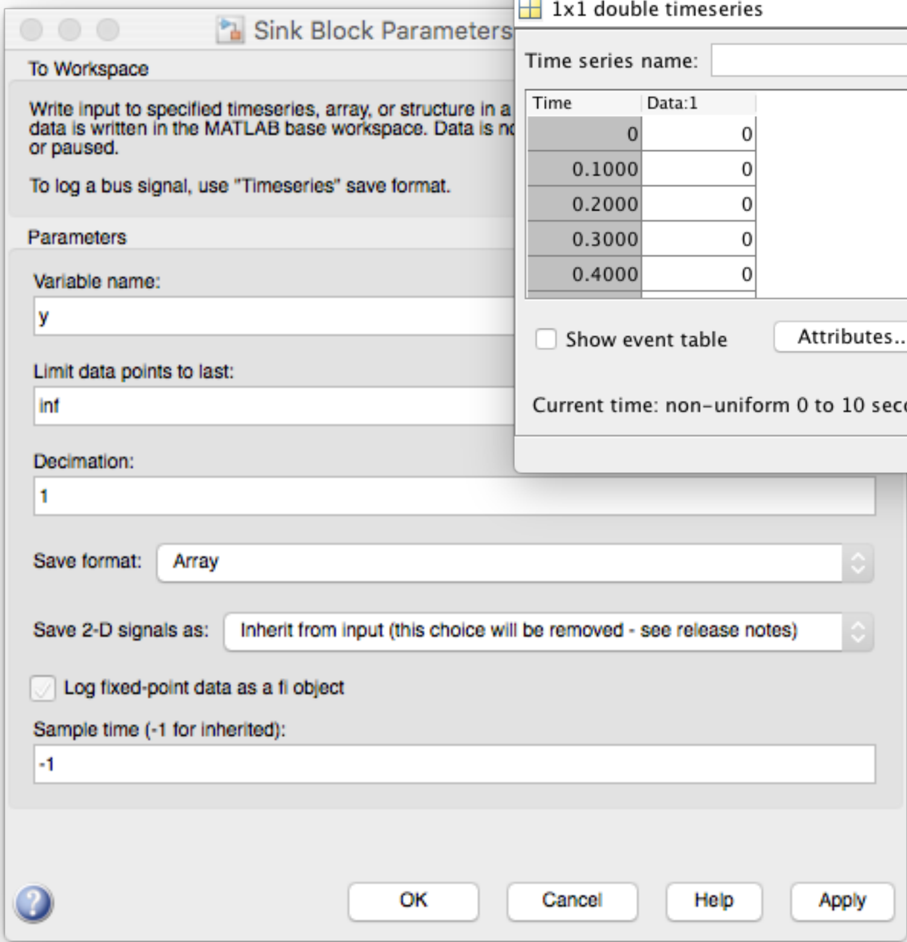
Gerar gráficos usando este formato de dados é simples, basta fazer: >> plot(y).

Se por acaso for desejado inspecionar o conteúdo interno de dados no Matlab do tipo ‘timeseries’, basta num caso como este fazer:

```
>> open y
```

Deve abrir uma janela como a mostrada ao lado.





Notar que o formato “Array” está para ser eliminado em versões futuras do Maltab (notar quadro de texto “Sace 2-D signal as”) na figura ao lado.

De posse dos dados levantados anteriormente com respeito a simulação para determinar Ku e Tu obtemos:

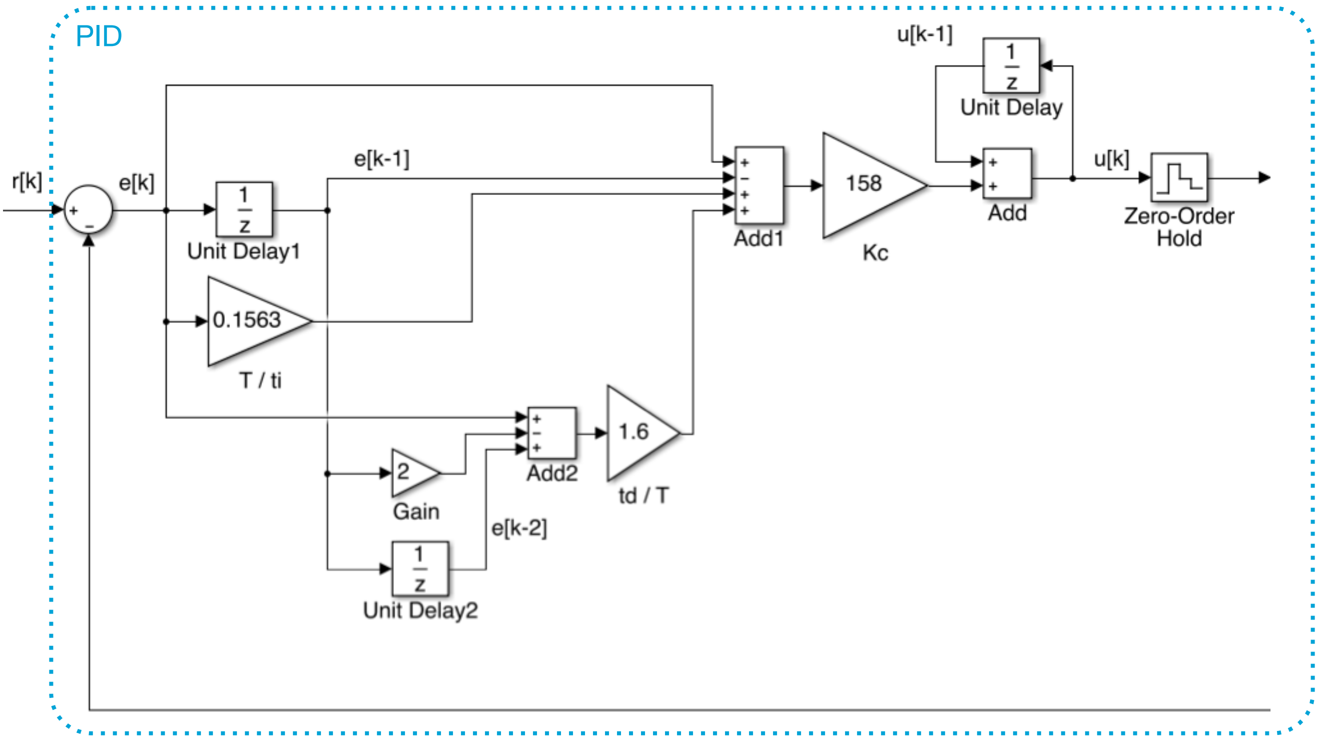
```
>> Ku
Ku =  264.4000
>> Tu
Tu =    1.4067
```

Usando a tabela de ZH para determinar Kc, Ti e Td obtemos:
PID —> Kc = 0,6 Ku; Ti = Tu/2; Td = Tu/8;

Então:

```
>> Kc=0.6*Ku
Kc =  158.6400
>> Ti=Tu/2
Ti =    0.7033
>> Td=Tu/8
Td =    0.1758
```

Mas.. os valores acima não podem ser aplicados diretamente num PID digital (apesar de poderem ser usados num PID no plano-s) porque faltou considerar o período de amostragem T:



Lembrando da equação do PID (formato de velocidade):

$$u[k] = u[k-1] + K_c \left[(e[k] - e[k-1]) + \frac{T}{\tau_i} e[k] + \frac{\tau_d}{T} (e[k] - 2e[k-1] + e[k-2]) \right]$$

A ponderação associado com a integração fica:

```
>> T/Ti =    0.1422
```

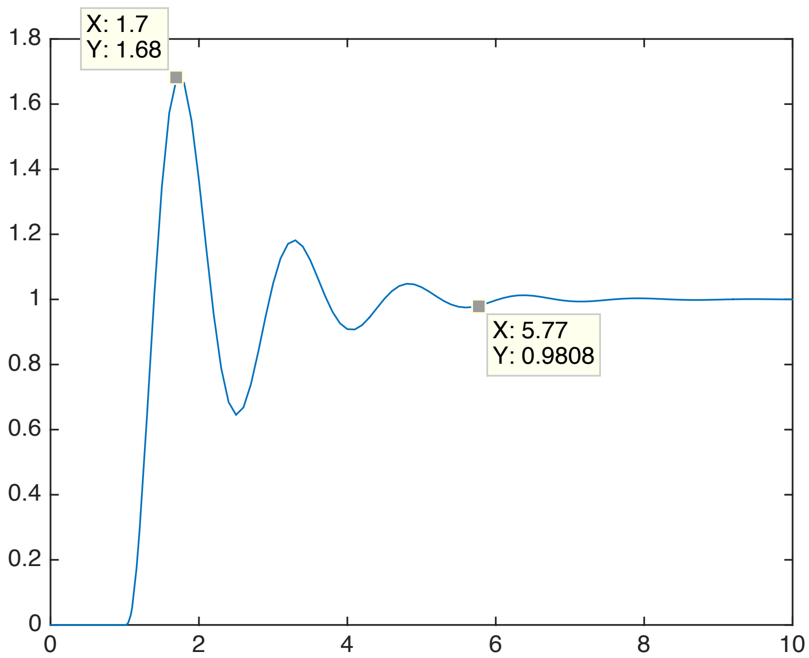
e a ponderação para o bloco associado com a derivada fica:

```
>> Td/T
ans =    1.7583
```

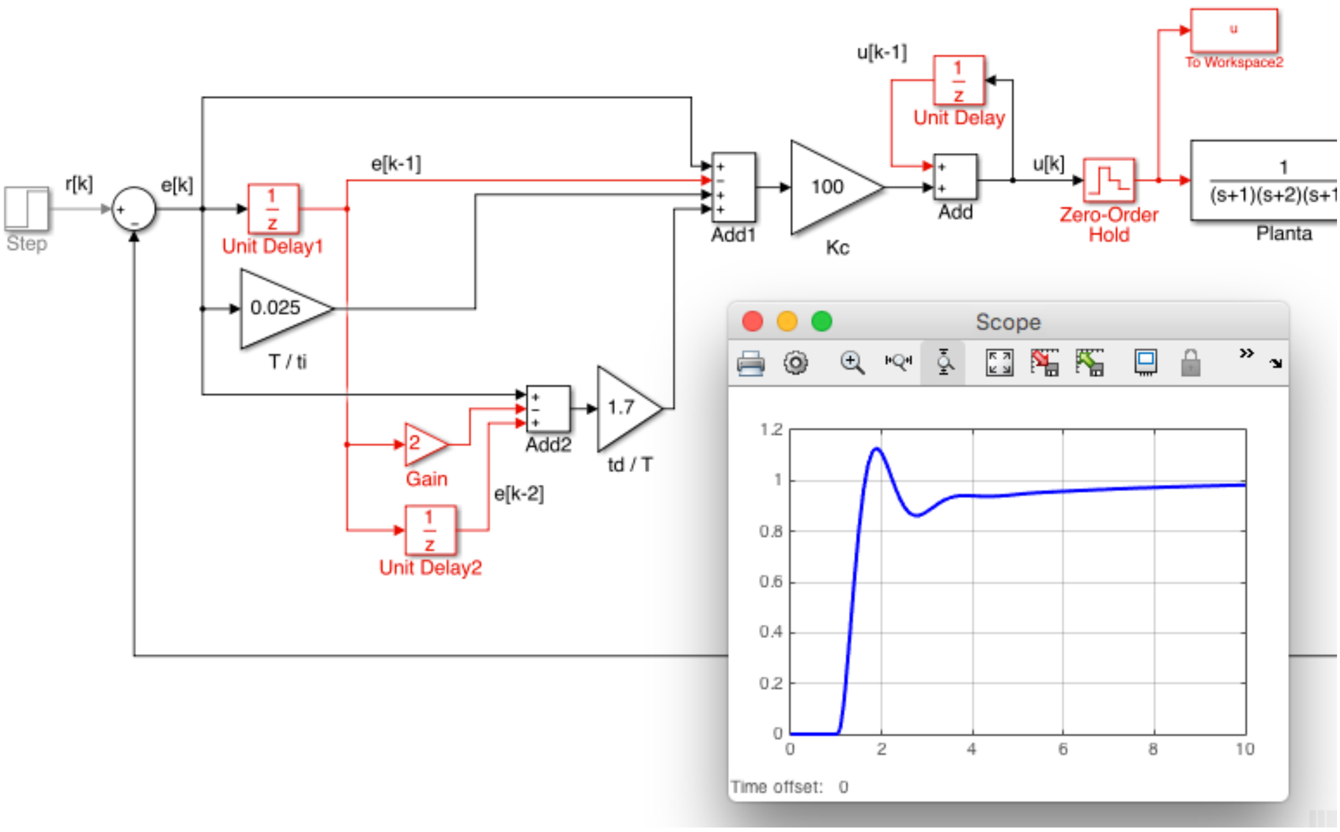
Os valores anteriores levam ao seguinte resultado:

Um overshoot de 68% no instante de tempo, t = 1,7 segundos

e um tempo de ajuste (2%) de, ts = 4,4 segundos.



Obviamente os valores inicialmente determinados para Kc, Ti e Td devem ser modificados para alcançar um melhor resultado:



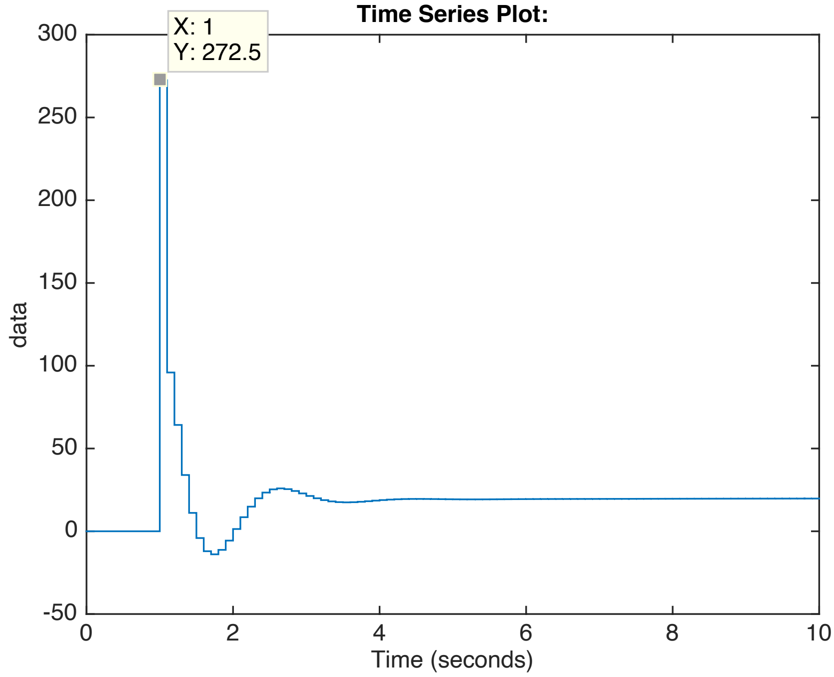
Mas repare nas amplitudes geradas para o sinal de controle (próxima página) -->

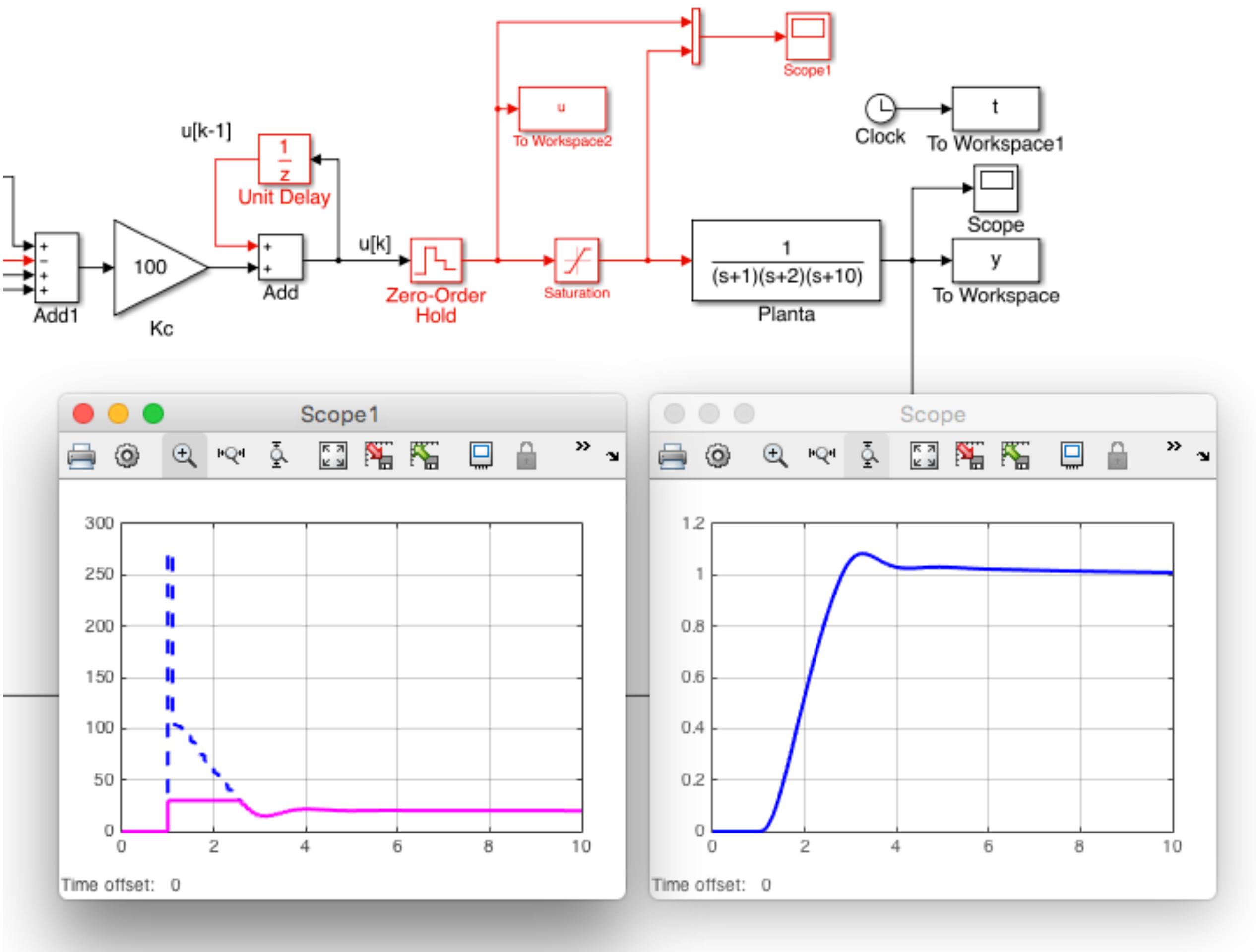
Amplitudes geradas para o sinal de controle (figura ao lado):

Note o valor elevado de amplitude gerado pelo PID.

Considerando o projeto de um controlador realizado anteriormente (por Atraso), a amplitude máxima do sinal de controle que o driver da plata suporta é |30,0|.

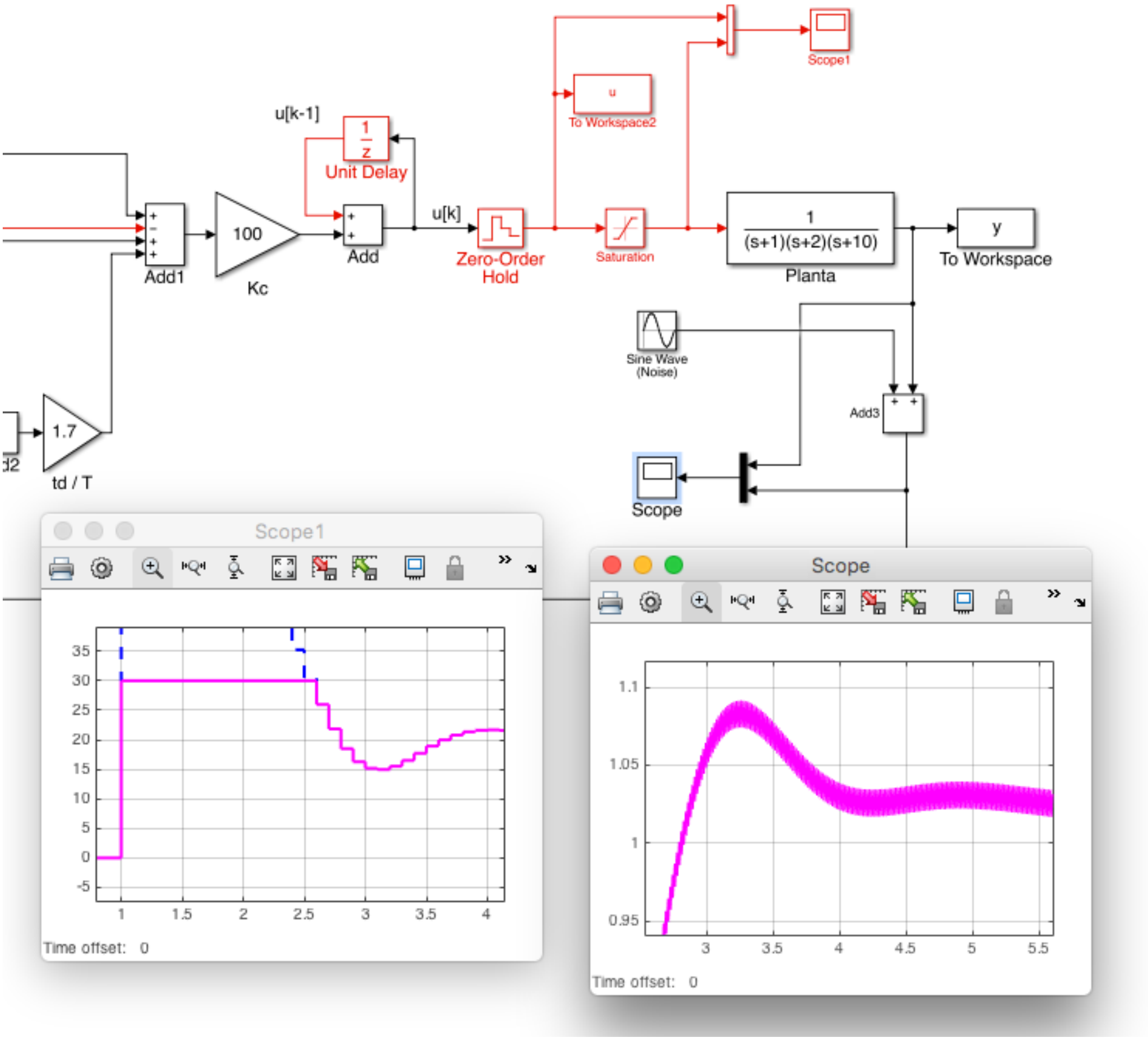
Considerando este fato, introduzimos um bloco de saturação na saída do PID e verificamos o que sucede:



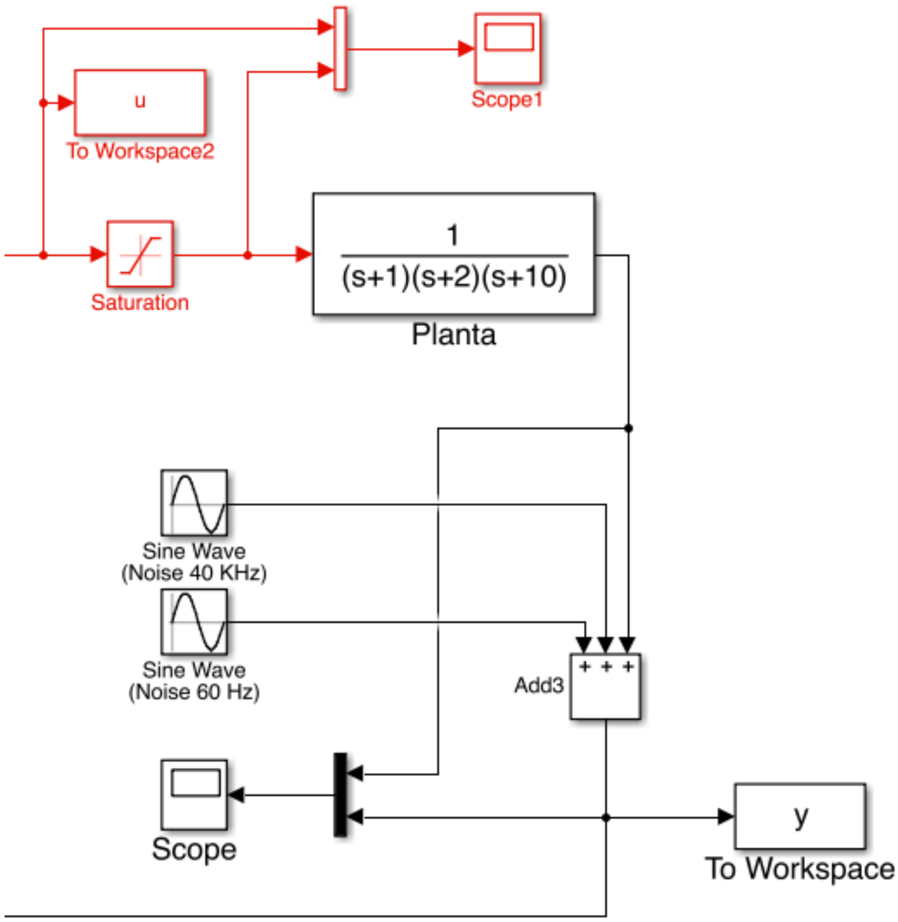


Continua -->

Mas a amplitude exagerado o sinal de controle pode não ser o único problema.
Suponha que agregado ao sinal de saída do sistema está sobreposto um ruído (senóide de 60Hz com amplitude 0,01 — ou seja 1% da amplitude em regime permanente da planta).

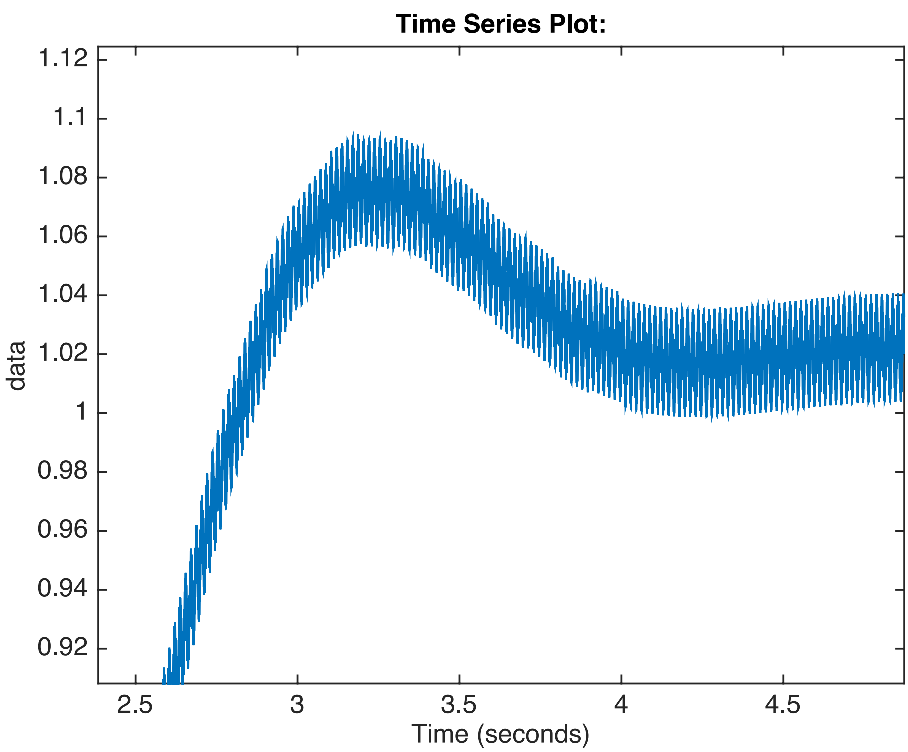


Continua -->



Suponha agora a mesma amplitude + mais um senoide de 40 KHz (ruído na rede provocado por fontes chateadas sem filtro de RF):

A saída do sistema se modifica para:



ANEXO1:

Script usado no MATLAB para calcular posição do zero com base em contribuição angular:
Projeto do Controlador por Avanço de Fase (Lead Compensator):

angulos.m:

```
% programa para determinar contribuição de ângulos
% Fernando Passold, em 16/out/2015
% Baseado em "example_9_4.m" de /UCV/Control/ (2009)
% Lead Compensator Desing (NISE)
% Parâmetros de entrada:
%   num = polinômio da FTMA(z), sem o zero de C(z)
%   den = polinômio da FTMA(z), incluindo o pólo de C(z)
%   zeta = fator de amortecimento desejado
%   T = período de amostragem adotado
% Observação:
%   sigma (plano-s) e new_sigma (plano-z) é calculado em função do ts desejado
% Sugestão, usar:
%   >> [num,den,aux]=tfdata(tf_FTMA,'v')
%
open_poles = roots(den)
[num_poles aux] = size(open_poles);
open_zeros = roots(num)
[num_zeros aux] = size(open_zeros);
new_t_s = input('New settiling time (desired): ? ');
wn=4/(zeta*new_t_s);
fprintf('Natural damping osc. frequency, wn = %.4f (rad/s)\n', wn);
sigma=wn*zeta;              % parte real do polo dominante no plano-s !
wd=wn*sqrt(1-zeta^2);    % parte imag do polo dominante no plano-s !
fprintf('Localização do pólo de MF no plano-s:\ns = %.4f +/- j%.4f\n', sigma,wd)
% falta calcular posição desejada para pólo de MF no plano-z
% usando definição da transformada-Z:
polos_MFs = sigma + j*wd;
polos_MFz = exp(-T*polos_MFs);
new_sigma = real(polos_MFz);
new_omega = abs(imag(polos_MFz));
fprintf('MF pole location on plan-z:\nz = %.4f +/- j%.4f\n', ..
   new_sigma,new_omega)

sump = 0;
fprintf('Angle Contribution of each pole of the open loop system:\n')
for i=1:num_poles
   % open_poles(i)
   % x(i) = -new_sigma - real(open_poles(i));
   x(i) = real(open_poles(i)) -new_sigma;
```

```
    y(i) = new_omega;
    th(i) = pi - atan2(y(i), x(i)); % determinando la contribuición de cada polo
    sump = sump + th(i);
    % th(i)*180/pi
    % fprintf('No. Pole : Pole : Angle Contribution\n')
    fprintf (' p%1i = %.4f --> %.2f^o\n', i,open_poles(i),th(i)*180/pi)
end
fprintf('Sum of angular poles positions: %.2f^o\n', sump*180/pi)

fprintf('Angle Contribution of each zero of the open loop system:\n')
sumz=0;
for i=1:num_zeros
    % open_poles(i)
    % x(i) = -new_sigma - real(open_poles(i));
    x(i) = real(open_zeros(i)) -new_sigma;
    y(i) = new_omega;
    th(i) = pi - atan2(y(i), x(i)); % determinando la contribuición de cada polo
    sumz = sumz + th(i);
    % th(i)*180/pi
    % fprintf('No. Pole : Pole : Angle Contribution\n')
    fprintf (' z%1i = %.4f --> %.2f^o\n', i,open_zeros(i),th(i)*180/pi)
end
fprintf('Sum of angular zeros positions: %.2f^o\n', sumz*180/pi)

final_angle = abs(pi - sump + sumz);
fprintf('Final Resulting angle for the extra Lead pole/zero: %.4f^o\n', final_angle*180/pi)

% pode-se determinar agora a posição para o zero ou pólo do Lead
fprintf('Ok, Evaluating the final position for the extra Lead pole/zero:\n')
pc=-(new_omega/tan(final_angle)-new_sigma);
fprintf('Final position for the extra Lead pole/zero: %.4f\n', pc)
% Seguiria nova figura gráfica plotando contribuição dos ângulos...
% Baseado no Exemplo 9-4 do NISE (adaptado do plano-s)
```

ANEXO2:
Script no MATLAB para auxiliar no projeto do controlador por avanço-atraso:

bode_gain_adjust.m

```
% Nise, N.S.
% Control Systems Engineering, 3rd ed.
% John Wiley & Sons, New York, NY, 10158-0012
%
% Control Systems Engineering Toolbox Version 3.0
% Copyright © 2000 by John Wiley & Sons, Inc.
% Chapter 11: Design via Frequency Response
%
% (ch11p1) Example 11.1: We can design via gain adjustment on the Bode plot using
% MATLAB. You will input the desired percent overshoot from the keyboard. MATLAB
% will calculate the required phase margin and then search the Bode plot for that
% phase margin. The magnitude at the phase-margin frequency is the reciprocal of
% the required gain. MATLAB will then plot a step response for that gain. Let us
% look at Example 11.1 in the text.

% disp('(ch11p1) Example 11.1')       % Display label.
clear all
close all
numg=[1];                          % Define numerator of G(s).
deng=poly([-1 -2 -10]);            % Define denominator of G(s).
G=tf(numg,deng);                   % Create G(s).
zpk(G)                             % display G(s)
pos=input('Type %OS ?: ');         % Input desired percent overshoot.
z=(-log(pos/100))/(sqrt(pi^2+log(pos/100)^2));            % Calculate required damping ratio.
fprintf('\nRequired damping ratio: %6.4f\n', z)
Pm=atan(2*z/(sqrt(-2*z^2+sqrt(1+4*z^4))))*(180/pi);     % Calculate required phase margin.
fprintf('Required phase margin, Pm = %7.4f°\n', Pm)

K=1;
fprintf('Input K (for start magnitud, K=%5.2f): ', K)
aux=input('? ');
if aux~=''
  K=aux;
end

w=0.1:0.01:100;                    % Set range of frequency from 0.01 to 1000 in steps of 0.01.
[Mag,P]=bode(K*numg,deng,w);       % Get Bode data.

figure;                            % Plot Bode diagram
subplot(2,1,1)
```

```
semilogx(w, 20.*log10(Mag));
%grid
aux=['Bode Diagram (K= ' num2str(K,'%5.2f') ')'];
title(aux)
ylabel('Magnitude (dB)');
subplot(2,1,2)
semilogx(w, P);
%grid
ylabel('Phase (deg)')
xlabel('Frequency (rad/sec)')

Ph=-180+Pm;                    % Calculate required phase angle.
fprintf('Required phase angle: %7.2f°\n', Ph)
u=length(P);
for k=1:1:u;                   % Search Bode data for required phase angle.
  if P(k)-Ph<=0;               % If required phase angle is found, find the value of
    M=Mag(k);                  % magnitude at the same frequency.
    fprintf('Found at w = %5.2f (rad/s)\n', w(k))
    fprintf('with magnitude = %5.2f dB (%5.2g)\n', 20*log10(M), M)
    new_K=1/M;                 % Calculate the required gain.
    subplot(2,1,2)
    hold on
    semilogx([w(k) w(k)], [-180 0], 'm:')
    semilogx([w(1) w(u)], [P(k) P(k)], 'm:')
    aux=[num2str(w(k),'%3.2f') ' rad/s'];
    text(log10(w(k)),0, aux)
    subplot(2,1,1)
    hold on
    semilogx([w(k) w(k)], [0 20.*log10(Mag(k))], 'm:')
    semilogx([w(1) w(u)], [20.*log10(Mag(k)) 20.*log10(Mag(k))], 'm:')
    aux=[num2str(20*log10(M),'%5.2f') ' dB'];
    text(log10(w(k)), 20.*log10(Mag(k)), aux )
    break                      % Stop the loop.
  end                          % End if.
end                            % End for.
fprintf('Then, required K = %6.2f\n', new_K)
k_final=K*new_K;
fprintf('Then, final required K = %6.2f\n', k_final)
T=feedback(k_final*G,1);               % Find T(s) using the calculated K.
figure; step(T)                        % Generate a step response.
title(['Closed-Loop Step Response for K= ',num2str(k_final)])       % Add title to step response.
% verificando diagrama de Bode compensado
figure; bode(G,k_final*G)
```

ANEXO2:

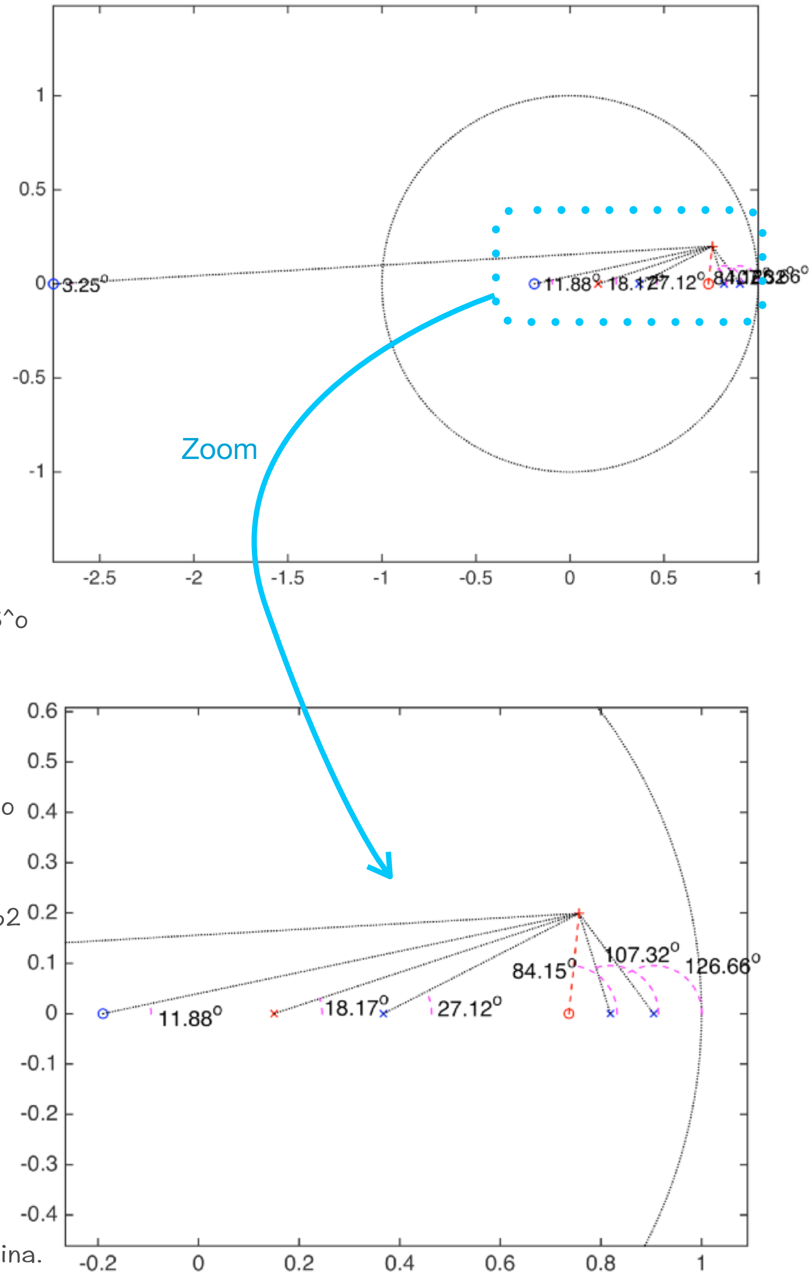
Revisão do script "**angulos.m**", do Projeto do Compensador por Avanço de Fase. Nesta versão, o novo script mostra didaticamente como ocorre a contribuição angular.

Note que para entrar com novas plantas, é necessário modificar parte do código de "**angulos2.m**", especificamente as variáveis `num` e `den` que registram o polinômio do numerador e denominador da planta à ser controlada, ainda no plano-s, e se for desejado modificar também o período de amostragem, a alocação `T=0.1` deve ser modificada também.

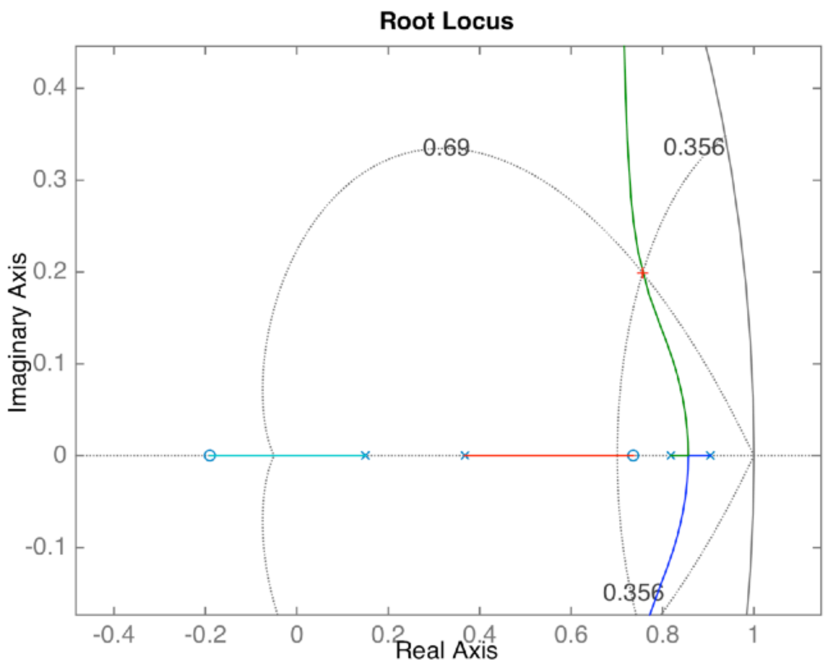
Segue exemplo de uso:

```
>> angulos2
Projeto Controlador por Avanço da Fase

Planta (no domínio-s:
         1
  ------------------
  (s+10) (s+2) (s+1)
Continuous-time zero/pole/gain model.

T =    0.1000

Planta digitalizada, BoG(z):
  0.00012224 (z+2.747) (z+0.1903)
  --------------------------------
  (z-0.9048) (z-0.8187) (z-0.3679)
Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.

Entre com overshoot máximo desejado (%OS), em %: ? 5
zeta (fator de amortecimento) deve ser: 0.6901
Entre com tempo de assentamento desejado, ts: ? 1.63
Resulta na frequencia de oscilação natural, wn =  3.5560 (rad/s)
Os pólos de MF (no plano-s) deveriam ficar localizados em:
2.4540 +/- j2.5735
Localização dos pólos de MF no plano-z:
z = 0.7566 +/- j0.1991
Indique a posição do pólo do controlador (plano-z): ? 0.15
O controlador ficaria algo como (sem o zero ainda):
     1
  --------
  (z-0.15)
Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.

FTMA(z) temporária:
       0.00012224 (z+2.747) (z+0.1903)
  -----------------------------------------
  (z-0.9048) (z-0.8187) (z-0.3679) (z-0.15)
Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.
```

```
open_poles =
    0.9048
    0.8187
    0.3679
    0.1500
open_zeros =
   -2.7471
   -0.1903

Contribuição angular de cada pólo no plano-z:
 p1 = 0.9048 --> 126.66^o
 p2 = 0.8187 --> 107.32^o
 p3 = 0.3679 --> 27.12^o
 p4 = 0.1500 --> 18.17^o
Somatório contribuição angular dos pólos: 279.28^o
Pause... [enter]
Contribuição angular de cada zero no plano-z:
 z1 = -2.7471 --> 3.25^o
 z2 = -0.1903 --> 11.88^o
Somatório contribuição angular dos zeros: 15.13^o
Ângulo final para o zero de C(z): 84.1486^o
Ok, determinando a posição para o zero de C(z):
Final position for the extra Lead pole/zero: 0.7362
O controlador por avanço de fase fica:
  (z-0.7362)
  ----------
   (z-0.15)
Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.
>>
```



Este script gera os gráficos mostrados nesta página.



Obs.: o script "angulos2.m", exige a function (script) "arc.m" para funcionar.

Seguem os códigos:

angulos2.m:

```
% programa para determinar contribuição de ângulos
% Fernando Passold, em 16/out/2015
% Baseado em "example_9_4.m" de /UCV/Control/ (2009)
% Lead Compensator Desing (NISE)
%
fprintf('Projeto Controlador por Avanço da Fase\n\n');
% entrando com dados da planta, no plano-s (transformada de Laplace):
num=1;
den=poly([-10 -2 -1]);
G=tf(num,den);
fprintf('Planta (no domínio-s:\n');
zpk(G)
T=0.1 % informando período de amostragem
BoG=c2d(G, T);
fprintf('Planta digitalizada, BoG(z):\n');
zpk(BoG)
[numd, dend, aux] = tfdata(BoG,'v');
OS=input('Entre com overshoot máximo desejado (%OS), em %: ? ');
% calculando fator de amortecimento em função de %OS:
zeta=(-log(OS/100))/(sqrt(pi^2+(log(OS/100)^2)));
fprintf('zeta (fator de amortecimento) deve ser: %6.4f\n', zeta)
ts_d=input('Entre com tempo de assentamento desejado, ts: ? ');
% determinando wn em função de zeta e ts:
wn=4/(zeta*ts_d);
fprintf('Resulta na frequencia de oscilação natural, wn = %7.4f (rad/s)\n', wn);
% calculando a posição desejado para pólos de malha-fechada:
% polos_MF em sigma +- jwd
sigma=wn*zeta;
wd=wn*sqrt(1-zeta^2);
fprintf('Os pólos de MF (no plano-s) deveriam ficar localizados em:\n%6.4f +/- j%6.4f\n', ...
    sigma, wd);
% mas nosso controlador está sendo projetado no mundo "digital" (plano-z),
% então aplicando a definição da transformada Z: z = e^{-Ts}:
polo_MFs=sigma - j*wd;
polo_MFz=exp(-T*polo_MFs);
new_sigma = real(polo_MFz);
new_omega = abs(imag(polo_MFz));
fprintf('Localização dos pólos de MF no plano-z:\nz = %.4f +/- j%.4f\n', ...
    new_sigma, new_omega)

% usuário deve agora arbitrar uma posição para o pólo do controlador por
% avanço de fase:
polo_c=input('Indique a posição do pólo do controlador (plano-z): ? ');
% montando controlador digital (sem o zero, à ser calculando usando
% contribuição angular
num_c=1;
den_c=poly(polo_c);
fprintf('O controlador ficaria algo como (sem o zero ainda):\n');
C=tf(num_c, den_c, T);
```

```
zpk(C)
fprintf('FTMA(z) temporária:\n');
ftma=C*BoG;
zpk(ftma)
[num,den,aux]=tfdata(ftma,'v');
% determinando número de pólos de MA:
open_poles = roots(den)
[num_poles aux] = size(open_poles);
% determinando número de zeros de MA:
open_zeros = roots(num)
[num_zeros aux] = size(open_zeros);

% Iniciando cálculos e gráficos relacionados com contribuição angular:
figure;
% plotando o círculo unitário (referência):
th=0: (2*pi)/360 : 2*pi;
x=1*cos(th);
y=1*sin(th);
plot(x,y,'k:');
axis ('equal');
hold on
% plotando em cor azul pólos e zeros de BoG(z)
[num_BoG, den_BoG, aux] = tfdata(BoG, 'v');
polos_BoG=roots(den_BoG);
zeros_BoG=roots(num_BoG);
plot(real(zeros_BoG), imag(zeros_BoG),'bo'); % zeros de BoG(z)
plot(real(polos_BoG), imag(polos_BoG),'bx'); % pólos de BoG(z)
% plotando em cor vermelha o pólo de C(z):
plot(real(polo_c), 0,'rx');
% plotando posição desejado para pólo de MF:
plot(real(polo_MFz), imag(polo_MFz),'r+');

% Iniciando cálculos das contribuições angulares.

% linha pontilhada ligando pólos de MA ao pólo de MF
for i=1:num_poles,
    plot(real([open_poles(i) polo_MFz]), imag([open_poles(i) polo_MFz]),'k:') % traça linhas pontilhadas
ligando cada pólo de MA ao pólo desejado em MF
end
sump = 0; % soma angulos dos pólos (contribuição angular pólos)
fprintf('Contribuição angular de cada pólo no plano-z:\n')
sum_aux=0;
for i=1:num_poles
    x(i) = real(polo_MFz) - real(open_poles(i));
    y(i) = imag(polo_MFz) - imag(open_poles(i));
    theta(i) = atan2(y(i), x(i)); % determinando la contribuición de cada polo
    theta_deg(i)=(theta(i)*180)/pi; % angulo em graus
    sump = sump + theta(i);
    fprintf (' p%1i = %.4f --> %.2f^o\n', i, open_poles(i), theta_deg(i))
    sum_aux=sum_aux+sqrt(x(i)^2+y(i)^2); % calcula distâncias entre pólos
```

```
end
avg=(sum_aux/num_poles)/4; % valor médio raio arco das crontibuições angulares
% plotando as contribuições angulares:
for i=1:num_poles,
    p1=[real(open_poles(i))+avg; 0]; % matriz 2 x 1; [x; y]
    p2=[real(open_poles(i))+avg*cos(theta(i)); avg*sin(theta(i)) ];
    center=[real(open_poles(i)); 0];
    arc(p1, p2, center);
    aux=num2str(theta_deg(i),'%7.2f'); % transforma número em string
    text(real(open_poles(i))+(avg/2)*cos( (theta(i)/1.5) ), ...
        (avg/2)*sin( (theta(i)/1.5) ), [aux '^o']);
end
fprintf('Somatório contribuição angular dos pólos: %.2f^o\n', sump*180/pi)

n=input('Pause...','s');

% linha pontilhada ligando zeros de MA ao pólo de MF
for i=1:num_zeros,
    plot(real([open_zeros(i) polo_MFz]), imag([open_zeros(i) polo_MFz]),'k:') % traça linhas pontilhadas
ligando cada pólo de MA ao pólo desejado em MF
end
fprintf('Contribuição angular de cada zero no plano-z:\n')
sumz=0;
for i=1:num_zeros
    x(i) = real(polo_MFz) - real(open_zeros(i));
    y(i) = imag(polo_MFz) - imag(open_zeros(i));% new_omega;
    theta(i) = atan2(y(i), x(i)); % determinando la contribuición de cada zero
    theta_deg(i)=(theta(i)*180)/pi; % angulo em graus
    sumz = sumz + theta(i);
    fprintf (' z%1i = %.4f --> %.2f^o\n', i, open_zeros(i), theta_deg(i))
end
% plotando as contribuições angulares dos zeros
for i=1:num_zeros,
    p1=[real(open_zeros(i))+avg; 0]; % matriz 2 x 1; [x; y]
    p2=[real(open_zeros(i))+avg*cos(theta(i)); avg*sin(theta(i)) ];
    center=[real(open_zeros(i)); 0];
    arc(p1, p2, center);
    aux=num2str(theta_deg(i),'%7.2f'); % transforma número em string
    text(real(open_zeros(i))+(avg/2)*cos( (theta(i)/1.5) ), ...
        (avg/2)*sin( (theta(i)/1.5) ), [aux '^o']);
end
fprintf('Somatório contribuição angular dos zeros: %.2f^o\n', sumz*180/pi)

final_angle = abs(pi - sump + sumz);
fprintf('Ângulo final para o zero de C(z): %.4f^o\n', final_angle*180/pi)

% pode-se determinar agora a posição para o zero ou pólo do Lead
fprintf('Ok, determinando a posição para o zero de C(z):\n')
zc=-(new_omega/tan(final_angle)-new_sigma);
fprintf('Final position for the extra Lead pole/zero: %.4f\n', zc)
```

```
% plotando o gráfico com o zero resultante:
plot(real(zc), imag(zc),'ro');
plot(real([zc polo_MFz]), imag([zc polo_MFz]),'r--')
p1=[real(zc)+avg; 0]; % matriz 2 x 1; [x; y]
p2=[real(zc)+avg*cos(final_angle); avg*sin(final_angle) ];
center=[real(zc); 0];
arc(p1, p2, center);
aux=num2str( (final_angle*180/pi) ,'%7.2f'); % transforma número em string
text(real(zc)+(avg/2)*cos( (final_angle/1.5) ), ...
(avg/2)*sin( (final_angle/1.5) ), [aux '^o']);
num_c=poly(zc);
C=tf(num_c, den_c, T);
fprintf('O controlador por avanço de fase fica:\n');
zpk(C)
% Verificando...
figure;
ftma=C*BoG;
rlocus(ftma);
hold on
zgrid(zeta, wn*T);
% plotando posição desejado para pólo de MF:
plot(real(polo_MFz), imag(polo_MFz),'r+');
```

arc.m:

```
% function arc.m
% Fernando Passold, em 25/05/2016
% baseado em: http://www.mathworks.com/matlabcentral/newsreader/view_thread/278048
%
% Parâmetros de entrada:
%   p1 = ponto de partida do arco;
%   p2 = ponto de chegada do arco;
%   center = ponto (centro) do arco;
% Detalhe: p1, p2, center envolvem vetor de 2 dimensões, contendo
% por exemplo:
%    p1(1)=coordenada X (ou parte real de número complexo)
%    p1(2)=coordenada Y (ou parte imaginária de número complexo)
% esta função só gera saída gráfica, então se
% pressupõe que já foi enviado antes comando como
%    'hold on'
function arc(p1,p2,center)
   n=20; % numero de pontos dentro do arco
   v1=p1-center;
   v2=p2-center;
   % v3=[0 -1;1 0]*v1; % v1 rotated 90 degrees CCW
   c = det([v1,v2]); % "cross product" of v1 and v2
   v3 = [0, -c; c, 0]*v1; % v3 lies in plane of v1 and v2 and is orthog. to v1
   % a = linspace(0, mod(atan2(det([v1, v2]), dot(v1, v2)), 2*pi)); % Angle range
   a = linspace(0, atan2(abs(c), dot(v1,v2)), n); % Angle range
   % Note the absence of the 'abs' function in 'atan2'
```

```
    % v = v1*cos(a)+v3*sin(a);
    v = v1*cos(a)+((norm(v1)/norm(v3))*v3)*sin(a); % Arc, center at (0,0)
    plot(v(1,:)+center(1),v(2,:)+center(2),'m--'); % Plot arc, centered at 'center'
    axis equal
end
```