Controle Automático III (Aula de 11/04/2017)

# Estudo de Caso: Projeto de Controladores por Atraso de Fase

Prof. Fernando Passold

---

# Comandos usados no MATLAB

(aula de 11/04/2017; Conteúdo revisado em 23/04/2019)

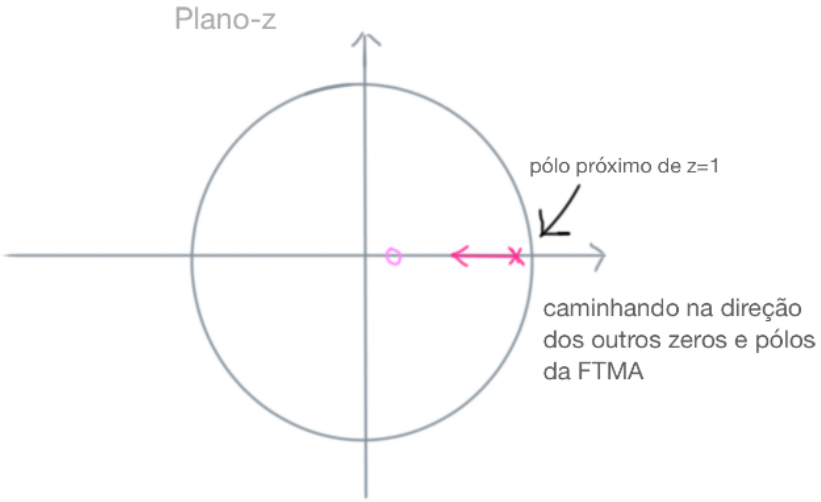
```
>> pwd % para confirmar o diretório atual de trabalho onde estão salvos os últimos arquivos de dados criados
ans =
/Volumes/Macintosh HD/Users/fernandopassold/Documents/MATLAB
>> diary 'aula_11abr2017.txt' % eventualmente um arquivo texto com comandos usados neste aula pode ser criado (opcional)
>> dir *.mat % lista arquivos de dados já criados anteriormente com comando 'save'
02nov2015.mat
03out2015.mat
...
planta.mat
planta_K_PI.mat
```

```
>> load planta_K_PI % carregando arquivo de dados com informações da planta e outras - não é necessário informar a extensão do arquivo (*.mat)
```

# Qual é a ideia do Controlador por Atraso de Fase?

Seria um "caso especial do "controlador PI com zero", ou quase um controlador PI + zero, onde seu pólo ao invés de ficar localizado em z=1, fica localizado próximo de z=1, desempenhando um papel próximo ao de um integrador.

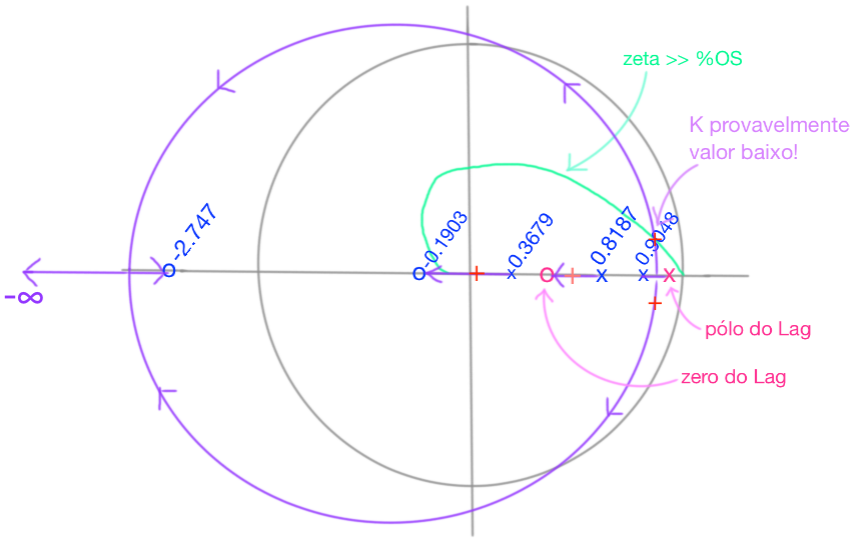
Como ficaria o RL com a introdução de um Controlador por Atraso? Segue um esboço do RL:




Nota-se que: zero < pólo para que continue sendo um controlador por atraso de fase.

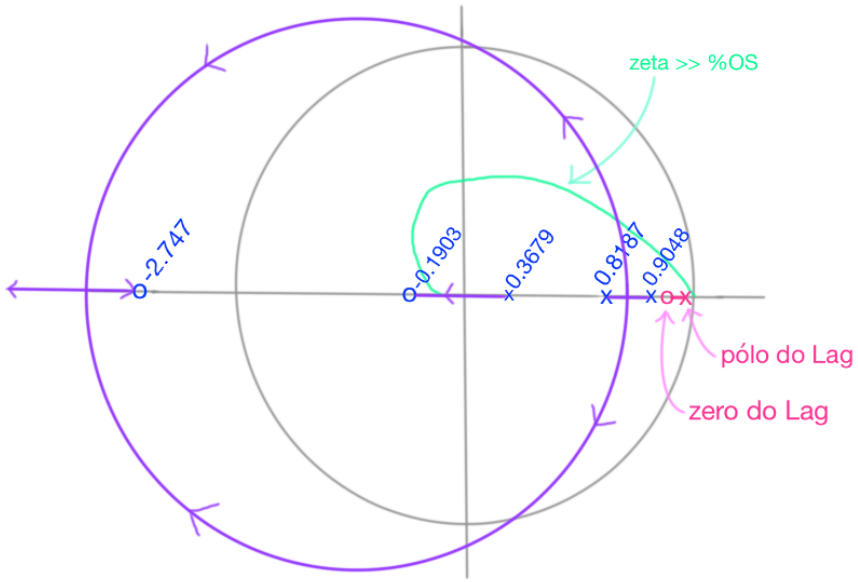
```
>> % Verificando os dados da planta para seguir raciocinando projeto do controlador por atraso de fase:
>> zpk(BoG)
ans =
  0.00012224 (z+2.747) (z+0.1903)
  -------------------------------
  (z-0.9048) (z-0.8187) (z-0.3679)
Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.
>>
```

Raciocinando como o RL poderia ser modificado para esta planta em particular: opções 1 e 2:




**Opção 1**: zero entre os pólos (da planta): 0,3679 e 0,8187. Detalhes: notar que o ponto de partida dos pólos de MA, z ≈ 1 (Lag) e z = 0,9018 ainda ocorre algo afastado da origem do plano-z e algo mais próximo do círculo unitário o que deve implicar em maior t_s (tempo de assentamento).

**Outra opção (2)**: forçar que o ponto de partida ocorra mais para dentro da origem do plano-z:



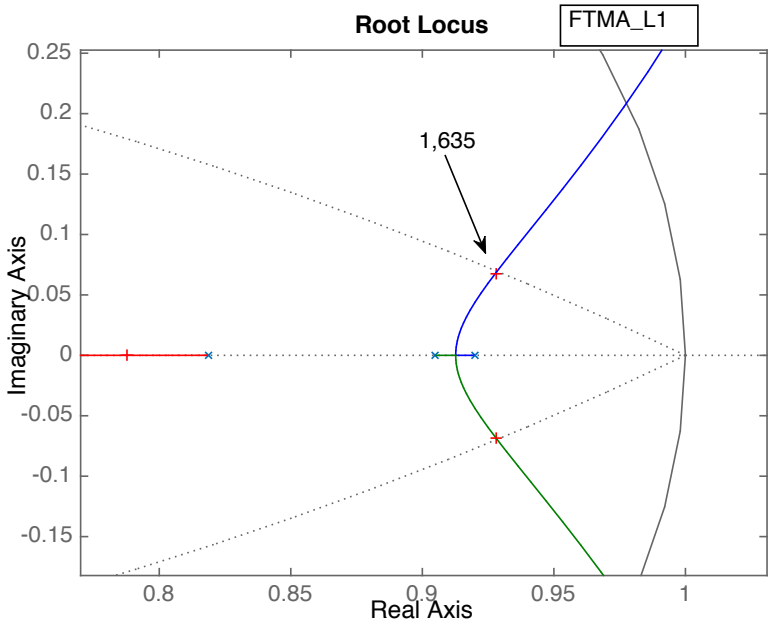

**Opção 2**: note que neste caso, o ponto de partida vai ocorrer um pouco mais próximo da origem do círculo unitário, o que deve implicar em menor t_s em comparação ao controlador da opção 1.

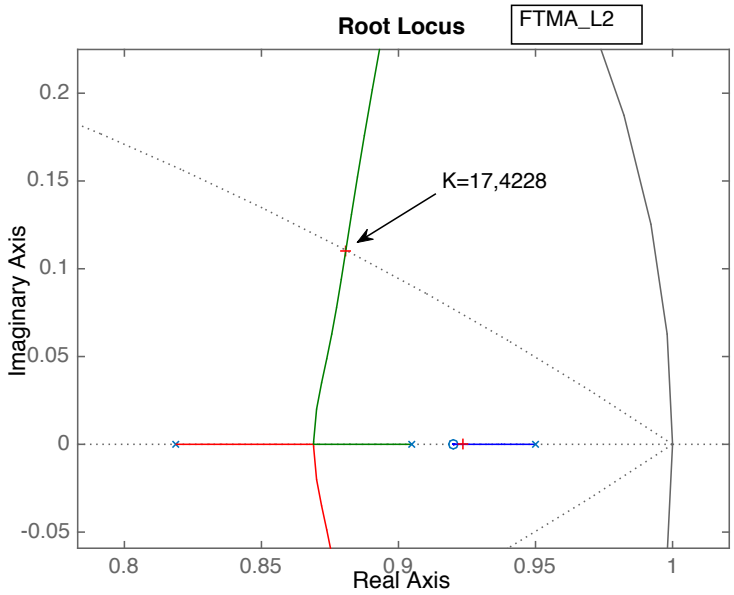
Usando MATLAB:

```
>> % entrando com dados da primeira versão do controlador de atraso de fase:
>> C_Lag1=tf([1 -0.5], [1 -0.92], T);
>> zpk(C_Lag1)
ans =
  (z-0.5)
  --------
  (z-0.92)
Sample time: 0.1 seconds
```

```
Discrete-time zero/pole/gain model.
>> % entrando com dados da segunda versão do controlador de atraso de fase:
>> C_Lag2=tf([1 -0.92], [1 -0.95], T);
>> zpk(C_Lag2)
ans =
  (z-0.92)
  --------
  (z-0.95)
Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.
>> % Incorporando os controladores à planta (BoG(z))
>> FTMA_L1=C_Lag1*BoG;
>> FTMA_L2=C_Lag2*BoG;
>> rlocus(FTMA_L1); % levantando o RLocus com o primeiro controlador
>> figure; rlocus(FTMA_L2)
>> figure(1);
>> zgrid(zeta,0);
```

Os comandos ‘rlocus(.)’ anteriores geram os seguintes gráficos:

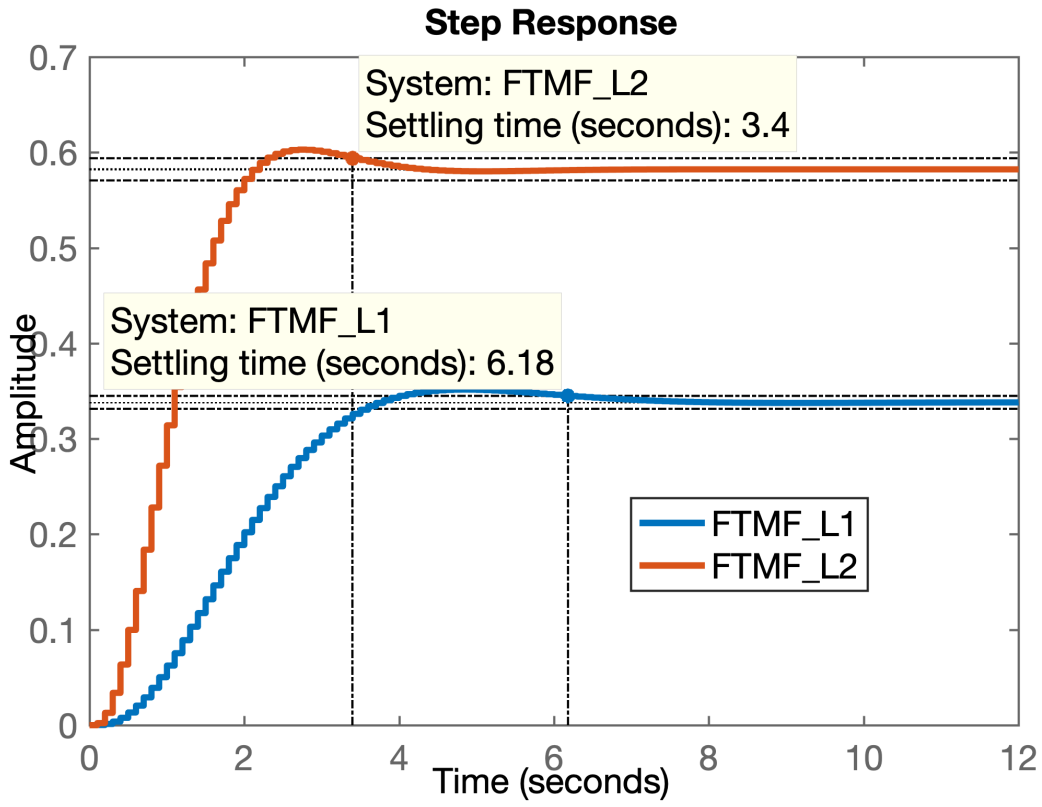





Continuando o projeto para o outro controlador:

```
>> figure(2); % desviando foco do MATLAB para RLocus do segundo controlador
>> zgrid(zeta,0);
>> % Sintonizando os controladores (obedecendo certo zeta e %OS)
>> figure(1); % voltando foco para RL do primeiro controlador
>> [K_Lag1, polos_MF_Lag1]=rlocfind(FTMA_L1);
Select a point in the graphics window
selected_point =
   0.9293 + 0.0675i
>> K_Lag1
K_Lag1 =
    1.6350
>> figure(2); % desviando foco para RL do segundo controlador
>> [K_Lag2, polos_MF_Lag2]=rlocfind(FTMA_L2)
Select a point in the graphics window
selected_point =
   0.8813 + 0.1101i
K_Lag2 =
   17.4228
polos_MF_Lag2 =
   0.9233 + 0.0000i
   0.8808 + 0.1102i
   0.8808 - 0.1102i
   0.3545 + 0.0000i
>> % fechando malhas para cada um dos controladores
>> FTMF_L1=feedback(K_Lag1*FTMA_L1,1);
>> FTMF_L2=feedback(K_Lag2*FTMA_L2, 1);
>> % verificando as respostas temporais obtidas para cada controlador, no mesmo gráfico
>> figure; step(K_degrau_L1*FTMF_L1, K_degrau_L2*FTMF_L2);
```

O comando anterior gera o seguinte gráfico:




Percebe-se que além dos diferentes tempos de assentamento, os valores dos erros em regime permanente são notadamente diferentes (além de elevados). O controlador da opção 1 além de mais lento é o que leva ao maior valor de erro em regime permanente. Já o controlador da opção 2 melhora tanto sua resposta temporal (mais rápido que o outro controlador) quanto o valor do erro em regime permanente.

## Determinando os valores dos erros em regime permanente:

Note que a função `dcgain` no MATLAB realiza o seguinte cálculo:

$$
\begin{aligned}
f(\infty) &= \lim_{k\to\infty} f(kT) \\
&= \lim_{z\to 1}(1-z^{-1})\cdot F(z) \quad \cdot \frac{z^{+1}}{z^{+1}} \\
&= \lim_{z\to 1}\frac{(z-1)}{z}\cdot F(z)
\end{aligned}
$$

← Lembrando teorema do valor final (Transformada Z)

No caso, `dcgain(f)` realiza mais que o cálculo acima. Esta função insere uma entrada degrau e depois aplica o teorema do valor final, resultando em:

$$
\begin{aligned}
y(\infty) &= \lim_{k\to\infty} \overbrace{u(kT)}^{\text{Degrau}} \cdot f(kT) \\
&= \lim_{z\to 1} \left[\frac{(z-1)}{z}\right] \cdot U(z) \cdot F(z) \\
&= \lim_{z\to 1} \cancel{\left[\frac{(z-1)}{z}\right]} \cdot \cancel{\left[\frac{z}{(z-1)}\right]} \cdot F(z) \\
&= \lim_{z\to 1} F(z)
\end{aligned}
$$

Então, uma vez entendido o que realiza dcgain, podemos facilmente determinar y(∞):

```
>> % verificando ganhos DC de cada FTMF(z)
>> dcgain(FTMF_L1)
ans =
    0.3382      % valor que pode ser comprovado no gráfico anterior
>> erro_L1 = (1 - dcgain(FTMF_L1))/1*100      % erro em percentual
erro_L1 =
   66.1838
>> dcgain(FTMF_L2)
ans =
    0.5823 % valor que pode ser comprovado no gráfico anterior
>> erro_L2 = (1 - dcgain(FTMF_L2))/1*100      % erro em percentual
erro_L2 =
   41.7741
>> % se percebe que os erros são elevados, sem a presença de um integrador.
>>
>> % se você abriu um arquivo usando 'diary', não esqueça de fechá-lo antes de sair do MATLAB:
>> diary off
>> save controladores_Lag % salva dados para eventual utilização futura.
```

# Conclusão:

Conforme os esboços realizados no início deste projeto, já era possível prever que a segunda versão do controlador de atraso de fase seria mais rápido que a primeira versão.

Apesar da 2a-versão alcançar resultados melhores que o controlador PI + Zero (um caso especial do controlador de atraso de fase), não foi possível respeitar o tempo de assentamento desejado (ts_d = 1,63 segundos).