

## PMR3409 CONTROLE II

### EXPERIÊNCIA 7:

#### IMPLEMENTAÇÃO DIGITAL DE CONTROLADORES TIPO PID

##### 1. Introdução

Nessa experiência será analisado algoritmos de conversão de controladores tipo PID do domínio do tempo contínuo para o domínio do tempo discreto.

Devem ser utilizados os parâmetros do controlador PI que você projetou na Experiência 4.

##### 2. Sistema de Controle Digital

A Figura 1(a) ilustra um sistema de controle digital. O sistema de controle é composto de um *Amostrador (Conversor Analógico-Digital - A/D)*, um *Controlador Digital* (no caso implementado em um computador) e um *Conversor Digital-Analógico (Reconstrutor de Ordem Zero – D/A)*. Observa-se que apesar do controlador estar implementado em um computador e trabalhar em tempo discreto, este sistema de controle produz um sinal de controle  $u(t)$  contínuo no tempo.

A Figura 1(b) ilustra o diagrama de blocos do sistema com as suas respectivas funções de transferência. Note que este sistema de controle possui simultaneamente sinais de tempo discreto e sinais de tempo contínuo. Os sinais  $E(z)$ ,  $H(z)$  e  $M(z)$  são sinais discretos no tempo. A parte de tempo discreto se refere ao algoritmo executado pelo computador e a parte de tempo contínuo se refere ao sistema físico correspondente a planta a ser controlada.

Note que existem variantes para este sistema. Por exemplo, a referência pode ser gerada pelo computador, assim, a saída da planta deve ser amostrada antes de ser comparada com o sinal de referência digital (tempo discreto), de forma a gerar um sinal de erro digital.

Como visto na Experiência 3 existem duas formas de projetar um controlador que será implementado através de um computador. A primeira consiste em projetar o controlador em tempo contínuo, depois discretizá-lo, usando algum método de integração numérica de equações diferenciais, para permitir a sua implementação em um computador. A segunda consiste em projetar o controlador diretamente em tempo discreto usando o modelo em tempo discreto do sistema a ser controlado.

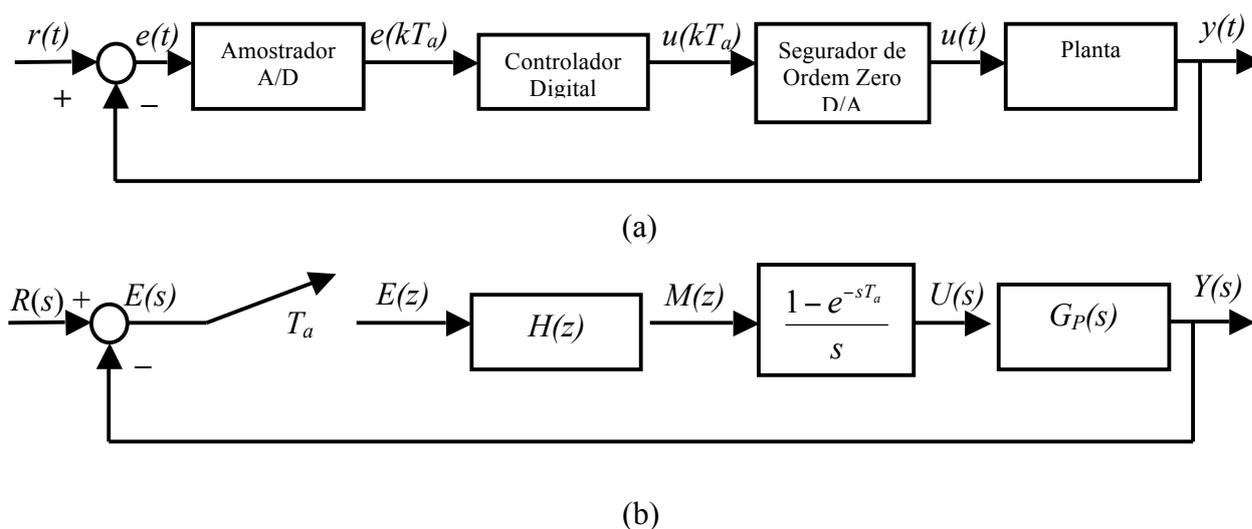


Figura 1: (a) Diagrama de blocos de um sistema de controle digital. (b) Diagrama de blocos equivalente evidenciando as funções de transferência.

Em qualquer uma das formas de abordagem, o projeto do controlador resulta em uma função de transferência, ou em  $s$  ou em  $z$ . Entretanto, a implementação digital de um controlador é realizada através de uma equação de diferenças, que é obtida pela função de transferência em  $z$ . Uma equação de diferenças tem em tempo discreto a mesma função de uma equação diferencial em tempo contínuo e ambas podem ser obtidas pela função de transferência correspondente em  $z$  ou em  $s$ . Ou seja, o equivalente de uma função de transferência em  $s$  é uma equação diferencial (domínio do tempo contínuo), e o equivalente de uma função de transferência em  $z$  é uma equação de diferenças.

## 2.1 Obtenção da equação de diferenças a partir da Transformada Z

Neste item é apresentado como obter a equação de diferenças correspondente à função de transferência do controlador em tempo discreto,  $H(z)$ , que será implementada em computador. Note que no caso de projeto do controlador ter sido realizado em tempo contínuo, antes de se obter a equação de diferenças, é necessário obter a função de transferência em tempo discreto equivalente à função de transferência em tempo contínuo. Os métodos para fazer esta aproximação (discretização) estão apresentados na Seção 3.

Para se obter a equação de diferenças equivalente a uma função de transferência em  $z$ , procede-se da seguinte forma. Seja a seguinte função de transferência genérica de ordem  $n$ :

$$\frac{Y(z)}{U(z)} = G(z) = \frac{b_m z^m + b_{m-1} z^{m-1} + \dots + b_1 z + b_0}{z^n + a_{n-1} z^{n-1} + \dots + a_1 z + a_0}, \quad (1)$$

onde  $m \leq n$ . O primeiro passo é colocar a função  $G(z)$  em função de potências negativas de  $z$ . Assim, dividindo-se em cima e em baixo por  $z^{-n}$ , tem-se:

$$\frac{Y(z)}{U(z)} = G(z) = \frac{b_m z^{m-n} + b_{m-1} z^{m-1-n} + \dots + b_1 z^{1-n} + b_0 z^{-n}}{1 + a_{n-1} z^{-1} + \dots + a_1 z^{-(n-1)} + a_0 z^{-n}}. \quad (2)$$

Multiplicando-se  $Y(z)$  pelo denominador de  $G(z)$  e  $U(z)$  pelo numerador de  $G(z)$ , obtém-se:

$$Y(z) + a_{n-1} z^{-1} Y(z) + \dots + a_1 z^{-(n-1)} Y(z) + a_0 z^{-n} Y(z) = b_m z^{-(n-m)} U(z) + b_{m-1} z^{-(n+1-m)} U(z) + \dots + b_1 z^{-(n-1)} U(z) + b_0 z^{-n} U(z). \quad (3)$$

Analogamente ao fato de que no plano  $s$  (domínio de tempo contínuo) a variável  $s^n$  multiplicando  $Y(s)$ , significa no tempo a derivada  $n$ -ésima de  $y(t)$ , tem-se que no plano  $z$  (domínio de tempo discreto) a variável  $z^n$  multiplicando  $Y(z)$ , significa no tempo a variável  $y(kT_a)$  atrasada de  $nT_a$  segundos, onde  $T_a$  é o tempo de amostragem. Assim, calculando-se a Transformada Z Inversa, obtém-se a seguinte equação de diferenças:

$$y(kT_a) = -a_{n-1} y(kT_a - T_a) - \dots - a_1 y(kT_a - (n-1)T_a) - a_0 y(kT_a - nT_a) + b_m u(kT_a - (n-m)T_a) + b_{m-1} u(kT_a - (n+1-m)T_a) + \dots + b_1 u(kT_a - (n-1)T_a) + b_0 u(kT_a - nT_a), \quad (4)$$

onde,  $kT_a$  representa o tempo absoluto. Esta equação representa uma fórmula de recorrência, onde conhecendo-se os valores passados de  $y(kT_a)$  e os valores da entrada  $u(kT_a)$  presente e passadas, pode-se avançar no tempo com um processo marchante. Note que são necessárias  $n$  condições iniciais para a variável  $y(kT_a)$ , ou seja,  $y(0)$ ,  $y(T_a)$ ,  $y(2T_a)$ , ...,  $y((n-1)T_a)$ , para se iniciar o processo de marcha no tempo. Observa-se ainda, que é com uma equação deste tipo que se implementa um controlador digital.

### 3. Implementação Digital de Controladores Analógicos

O projeto de um controlador em tempo contínuo resulta em uma função de transferência em  $s$ , digamos  $H(s)$ . Para implementar este controlador em um computador é necessário a sua discretização no tempo. Existem, também, algumas situações onde um controlador de tempo contínuo já está disponível e deseja-se converter este controlador para um controlador de tempo discreto.

#### 3.1. Aproximação digital de controladores analógicos

A discretização, ou aproximação digital de um controlador em tempo discreto pode ser realizada por diversos métodos de integração de equações diferenciais. O objetivo final da discretização de um controlador analógico é a obtenção da equação de diferenças que será implementada em um computador.

Uma forma de realizar esta discretização seria obter a equação diferencial correspondente à função de transferência do controlador  $H(s)$  utilizando algum método numérico de integração de equações diferenciais ordinárias. Contudo, na área de controle o processo preferido é obter a função de transferência em tempo discreto,  $H(z)$ , a partir de  $H(s)$  e depois obter a equação de diferenças. Assim, o problema passa a ser o seguinte: dado um

controlador em tempo contínuo descrito por sua função de transferência,  $H(s)$ , deseja-se encontrar uma função de transferência em  $z$ , que aproxime  $H(s)$  em tempo discreto.

Existem inúmeros métodos que podem ser utilizados para discretizar um controlador, contudo alguns são mais simples e mais usuais. A seguir são apresentados alguns destes métodos mais usuais.

### Método de diferenças para Frente (ou Método de Euler):

No método da diferença para frente, a primeira derivada no tempo de uma função é aproximada por uma diferença no tempo da seguinte forma:

$$\frac{dx(t)}{dt} \approx \frac{x(t + T_a) - x(t)}{T_a} \quad (5)$$

Calculando a Transformada de Laplace do lado esquerdo e a Transformada Z do lado direito da equação (5) tem-se:

$$sX(s) \approx \frac{X(z)z - X(z)}{T_a} = \frac{(z - 1)X(z)}{T_a}. \quad (6)$$

Para que os dois lados da equação (6) sejam equivalentes, ou pelo menos aproximadamente iguais, tem-se que:

$$s \approx \frac{(z - 1)}{T_a}. \quad (7)$$

Ou seja, para uma função  $H(s)$  a função em tempo discreto  $H(z)$  equivalente pode ser obtida pela simples substituição de  $s$  pela expressão de  $z$  dada pela equação (7), ou seja:

$$H(z) \approx H(s) \Big|_{s = \frac{(z-1)}{T_a}}. \quad (8)$$

### Método de Diferenças para Trás:

Utilizando agora uma *diferença para trás* para a aproximar a primeira derivada de uma função no tempo, tem-se:

$$\frac{dx(t)}{dt} \approx \frac{x(t) - x(t - T_a)}{T_a}. \quad (9)$$

Calculando a Transformada de Laplace do lado esquerdo e a Transformada Z do lado direito da equação (9) tem-se:

$$sX(s) \approx \frac{X(z) - z^{-1}X(z)}{T_a} = \frac{(1 - z^{-1})X(z)}{T_a} = \frac{(z - 1)X(z)}{zT_a}. \quad (10)$$

Desta forma, conclui-se que para este método tem-se a seguinte aproximação entre  $s$  e  $z$ :

$$s \approx \frac{(z - 1)}{zT_a}. \quad (11)$$

Ou seja, para uma função  $H(s)$  a função equivalente em tempo discreto  $H(z)$  pode ser obtida por:

$$H_D(z) = H(s) \Big|_{s = \frac{(z-1)}{zT_a}}. \quad (12)$$

### Transformação Bilinear, ou Método de Aproximação de Tustin:

Uma outra aproximação corresponde ao *método de integração trapezoidal* que é denominado Transformação Bilinear ou Método de Aproximação de Tustin. Neste caso a aproximação é feita utilizando a seguinte equivalência:

$$s = \frac{2}{T_a} \frac{(z - 1)}{(z + 1)}. \quad (13)$$

Ou seja, para uma função  $H(s)$  a função em tempo discreto  $H(z)$  equivalente pode ser obtida por:

$$H_D(z) = H(s) \Big|_{s = \frac{2(z-1)}{T_a(z+1)}}. \quad (14)$$

### Método do Casamento de Pólos e Zeros:

Uma maneira simples e eficiente para a obtenção do equivalente discreto é a utilização do mapeamento entre o plano  $s$  e o plano  $z$  através da Transformada  $Z$ . Se a transformada  $Z$  for aplicada a um sinal contínuo  $x(t)$ , então os pólos da transformação discreta  $X(z)$  estarão relacionados com os pólos de  $X(s)$  de acordo com a seguinte relação:

$$z = e^{sT_a}. \quad (15)$$

Para os zeros não existe uma relação tão direta, mas a idéia fundamental do casamento de pólos e zeros é que o mapeamento dado pela equação (5) também pode ser aplicado para os zeros. A técnica consiste em um conjunto de regras heurísticas para a localização de pólos e zeros e a imposição de um ganho equivalente entre  $H(s)$  e  $H(z)$  para baixas ou altas frequências dependendo do caso. As regras são as seguintes:

1. Todos os pólos de  $H(s)$  são mapeados de acordo com a equação (15);
2. Todos os zeros finitos são mapeados através da equação (15);
3. Os zeros de  $H(s)$  em  $s = \pm\infty$  podem ser mapeados em  $H(z)$  nos pontos  $z = -1$  ou  $z = 0$ . Assim,  $H(z)$  possui o mesmo número de zeros finitos do que pólos;
4. O ganho do filtro digital é selecionado para ser igual ao ganho de  $H(s)$  em um ponto crítico equivalente;

Na maioria das aplicações em controle, o ponto crítico equivale a  $s = 0$ , ou seja em regime permanente, desta forma o ganho pode selecionado de tal forma que:

$$H(s)|_{s=0} = H_D(z)|_{z=1}. \quad (16)$$

Para o caso de um controlador PI o ganho na frequência zero é infinito. Assim, neste caso, deve-se fazer com que a resposta a um impulso do controlador digital tenha o mesmo valor final da resposta do controlador analógico, ou seja:

$$T_a \lim_{s \rightarrow 0} sH(s) = \lim_{z \rightarrow 1} (z - 1)H(z). \quad (17)$$

Nota-se que neste caso o período de amostragem aparece multiplicando o lado esquerdo da equação. A razão disto é que um impulso em tempo discreto tem a duração de um período de amostragem e em tempo contínuo um impulso tem duração infinitesimal.

## Análise de Estabilidade

A Figura 2 ilustra como a região de estabilidade no plano  $s$  ( $Re(s) < 0$ ) é mapeada no plano  $z$  para os métodos de Diferenças para Frente, de Diferenças para Trás e da Transformação Bilinear.

Observa-se que com o Método de Diferenças para Frente, é possível que um sistema estável em  $s$  seja mapeado como um sistema instável em  $z$ .

Quando o Método de Diferenças para Trás é utilizado, para um sistema estável em  $s$  obtém-se sempre um sistema estável em  $z$ . Entretanto existem sistemas instáveis em  $s$  que são mapeados como sistemas estáveis em  $z$ . Note também que a região mapeada em  $z$  através deste método é bem menor que o círculo unitário que representa a estabilidade, o que indica uma distorção acentuada dos pólos quando mapeados em  $z$ .

O Método da Transformação Bilinear tem a vantagem de transformar o semi-plano esquerdo em  $s$ , no círculo unitário em  $z$ . Sistemas estáveis em  $s$  são transformados em sistemas estáveis em  $z$  e sistemas instáveis em  $s$  são transformados em sistemas instáveis em  $z$ .

O Método do Casamento de Pólos e Zeros como utiliza a própria definição de Transformada  $Z$  também mapeia o semi-plano esquerdo de  $s$  como um círculo unitário em  $z$ , preservando, desta forma, as condições de estabilidade.

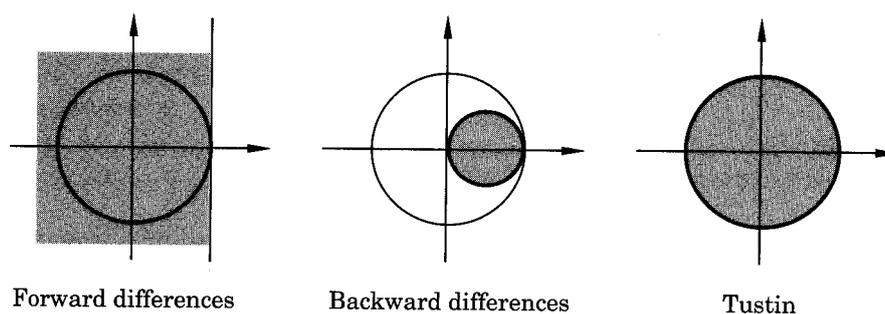


Figura 2: Mapeamento da região estável do s para o plano z de acordo com o método de diferenças a frente, diferenças a trás e o método de Tustin.

### 3.2. Seleção do período de amostragem

A escolha apropriada do período de amostragem  $T_a$  é uma decisão importante em um sistema de controle digital. Se o período de amostragem for muito longo será impossível reconstruir o sinal contínuo. Se o período de amostragem for muito pequeno o computador ficará sobrecarregado. A escolha do período de amostragem depende bastante do propósito do sistema.

Em princípio quanto menor for o período de amostragem melhor será a aproximação digital, certo? Não necessariamente! Este paradoxo pode ser verificado pelo seguinte: se o período de amostragem for muito pequeno, os pólos do controlador se aproximam da unidade, que é também o limite da estabilidade. De acordo com a equação de mapeamento entre s e z, repetida abaixo,

$$z = e^{sT_a},$$

se  $T_a$  for muito pequeno, tendendo a zero, os pólos do controlador em z tendem a 1, que são os pólos de um sistema marginalmente instável, tornando, assim, o sistema em malha fechada instável!!!! Contudo, não é necessário que  $T_a$  seja igual a zero para que problemas comecem a surgir, basta que  $T_a$  seja pequeno o suficiente para que os pólos em z não consigam ser distinguidos da unidade pelo computador. Lembre-se que o computador usa um número finito de *bits* para representar números e, assim, dependendo do número de *bits*, não consegue distinguir, por exemplo, 1 de 0,99999. Portanto, períodos de amostragem pequenos podem introduzir distorções significativas no comportamento dinâmico do sistema.

A escolha apropriada do período de amostragem  $T_a$  é uma decisão importante em um sistema de controle digital. Se o período de amostragem for muito longo será impossível reconstruir o sinal contínuo. Se o período de amostragem for muito pequeno o computador ficará sobrecarregado e componentes de alta frequência serão introduzidos na malha fechada pela conversão digital-analógica. A escolha do período de amostragem depende bastante do propósito do sistema.

A escolha do período de amostragem pode ser feita com base numa variável adimensional e que possui uma interpretação física. Para sistemas oscilatórios é natural a

normalização com base no período de oscilação, para sistemas não oscilatórios, o tempo de subida do sistema é o fator de normalização adequado.

Define-se o número de períodos de amostragem por tempo de subida,  $N_r$ :

$$N_r = \frac{t_r}{T_a}, \quad (72)$$

onde  $t_r$  é o tempo de subida. Para sistemas de 1ª ordem, o tempo de subida é igual a constante de tempo. **Portanto, é razoável escolher  $N_r$  por volta de 10 para o caso de projeto do controlador realizado em tempo discreto. Contudo, se o projeto do controlador foi realizado em tempo contínuo,  $N_r$  deve ficar por volta de 20.**

Para um sistema de segunda ordem com grau de amortecimento  $\xi$  e frequência natural  $\omega_n$ , o tempo de subida é dado por:

$$T_r = \omega_0^{-1} e^{\varphi / \tan \varphi}, \quad (73)$$

onde  $\xi = \tan \varphi$ . Para um grau de amortecimento em torno de  $\xi=0.7$ , isto resulta em:

$$\omega_0 T_a \approx 0,1 \text{ ou } 0,05, \quad (74)$$

onde  $\omega_0$  é dado em radianos por segundo.

Uma questão importante a ser considerada é que os pólos e zeros no domínio  $z$  dependem do período de amostragem. Como se sabe, os pólos do domínio  $s$  são mapeados no domínio  $z$  através da seguinte relação:

$$z = e^{sT_a}. \quad (75)$$

Uma outra forma de escolher o período de amostragem é utilizar a eq. (75) de modo que os pólos do sistema de malha aberta em tempo discreto fiquem entre 0,95 e 0,98. Assim, invertendo a eq. (75), para pólo de tempo em 0,95 tem-se:

$$T_a = \frac{\ln(0,95)}{p}, \quad (76)$$

onde  $p$  é o pólo mais rápido da malha aberta. Esse método garante que os pólos do sistema em tempo discreto tenham um número de dígitos significativos suficiente de forma que não sejam arredondados para a unidade devido a erros de quantização.

## 4 Roteiro experimental

Você deve utilizar aqui os parâmetros do controlador PI projetos na Experiência 4:

$$H(s) = K_p \left( 1 + \frac{1}{T_i s} \right). \quad (1)$$

Também será necessário a utilização dos parâmetros estimados da função de transferência de velocidade do motor CC:

$$G(s) = \frac{K_m}{T s + 1} \quad (2)$$

### 4.1 Controlador PI discreto - análise no domínio da frequência

O controlador PI contínuo deve ser inicialmente discretizado através dos métodos abaixo:

- Método de Euler de Diferenças para Frente,
- Método de Euler de Diferenças para Trás,
- Método da Transformação Bilinear,
- Método do Casamento de Pólos e Zeros.

Os quatro controladores PI discretos obtidos devem ser comparados no domínio da frequência  $z$ . A qualidade da aproximação pode ser analisada comparando-se os diagramas de bode dos controladores PI discretos com os diagramas de bode do controlador PI contínuo.

Um script MATLAB denominado `bodepi.m` calcula e plota os diagramas de Bode para todos os controladores PI. A listagem a seguir apresenta um trecho do código de `bodepi.m`.

```
% Controlador PI
% H(s) = Kp (1 + 1/Tis)
%
% Ta -> tempo de amostragem
% expfi -> potencia de 10 para o inicio da escala log ? i.e. grafico
%       comeca em 10?(expfi)
% expff -> idem para o final da escala log, i.e., 10^(expff)
%
Kp = 2.0
Ti = 0.2
Ta = 0.05
expfi = -1
expff = +2
%
clf % apaga a figura corrente
%
% definicao da faixa de frequencia
w = logspace(expfi,expff,100);
%
% Diagrama de Bode para o PI no dominio s
%
num=[Kp*Ti Kp];
den=[Ti 0];
[mag, phase]=bode(num, den, w);
mag=20*log10(mag);
% modulo
subplot(2,1,1)
semilogx(w, mag, 'k-');
```

```

xlabel('rad/s');
ylabel('Modulo_(Db)');
grid on
hold on
% fase
subplot(2,1,2)
semilogx(w,phase,'k-');
xlabel('rad/s');
ylabel('Fase');
grid on
hold on
%
% Controlador PI discretizado atraves da Transformacao Bilinear
%
num=[Kp*(Ta+2*Ti) Kp*(Ta-2*Ti)]
den=[2*Ti -2*Ti]
[mag,phase] = dbode(num,den,Ta,w);
mag=20*log10(mag);
% modulo
subplot(2,1,1)
semilogx(w,mag,'r-');
xlabel('rad/s');
ylabel('Modulo_(Db)');
grid on
hold on
% fase
subplot(2,1,2)
semilogx(w,phase,'r-');
xlabel('rad/s');
ylabel('Fase');
grid on
hold on
..... continua

```

Experimentos a serem realizados:

1. **Melhor aproximação:** Qual controlador PI discreto aproxima melhor o controlador PI contínuo no domínio da frequência ?

Utilize os seguintes parâmetros:

```

Ta = 0.05
expfi = -1
expff = +2

```

Compare os diagramas de Bode dos controladores PI discretos em relação aos diagramas de Bode do controlador PI contínuo.

2. **Sensibilidade à variação do intervalo de amostragem  $T_a$ :** Obtenha os diagramas de Bode dos controladores PI para diferentes valores de intervalos de amostragem  $T_a = 0.01, 0.05, 0.1$ . Analise os resultados obtidos.

3. **Instabilidade em altas frequências:** Utilize os seguintes parâmetros:

```

Ta = 0.05
expfi = -1
expff = +3

```

Analise os resultados obtidos.

## 4.2 Análise de margens de estabilidade - sensibilidade à variação do intervalo de amostragem $T_a$

Deseja-se analisar a variação das margens de ganho e de fase do sistema de controle utilizando um controlador PI discretizado através do método de transformação bilinear.

Para estimar as margens de ganho e de fase, no caso contínuo, devemos inicialmente plotar o diagrama de Bode de malha aberta ou seja:

$$L(s) = H(s)G(s), \quad (3)$$

onde:

$$H(s) = K_p \left( 1 + \frac{1}{s} \right). \quad (4)$$

e

$$G(s) = \frac{K_m}{Ts + 1}. \quad (5)$$

A malha aberta no caso discreto é representada por:

$$L(z) = H(z)G(z), \quad (6)$$

onde  $H(z)$  é o equivalente discreto do controlador PI utilizando transformação bilinear e  $G(z)$  é o equivalente discreto da planta acrescido de um segurador de ordem zero (ZOH).

Deve ser utilizado o script MATLAB denominado `openloopfreqdom.m` cuja listagem é mostrada a seguir. O script plota o diagrama de bode de malha aberta contínuo ( $L(s)$ ) e discreto ( $L(z)$ ).

```
% Intervalo de amostragem
Ta = 0.05
% Controlador PI Transformacao Bilinear
%
Kp = 2
Ti = 0.2
%
num=[Kp*(Ta+2*Ti) Kp*(Ta-2*Ti)]
den=[2*Ti -2*Ti]
control = tf(num,den,Ta)
%
% Planta G(s)= Km/(Ts+1)
%
Km = 1.0
T = 0.25
a=1/T
% zoh + planta
splanta = tf([Km*(1-exp(-a*Ta))], [1 -exp(-a*Ta)], Ta)
% malha aberta
openloop = control*splanta
clf
w = logspace(-1, 2, 100);
dbode(openloop, Ta, w)
grid
hold on
%
% Dominio do tempo contínuo
%
picont = tf([Kp*Ti Kp], [Ti 0])
splantacont = tf([Km], [T 1])
openloopcont = picont * splantacont
bode(openloopcont, w)
```

Após obtido o diagrama de bode o MATLAB estima as margens de ganho e de fase. Basta clicar com o botão direito do *mouse* e selecionar *Stability Margins* na opção *Characteristics*. Os pontos de *crossover* ficarão

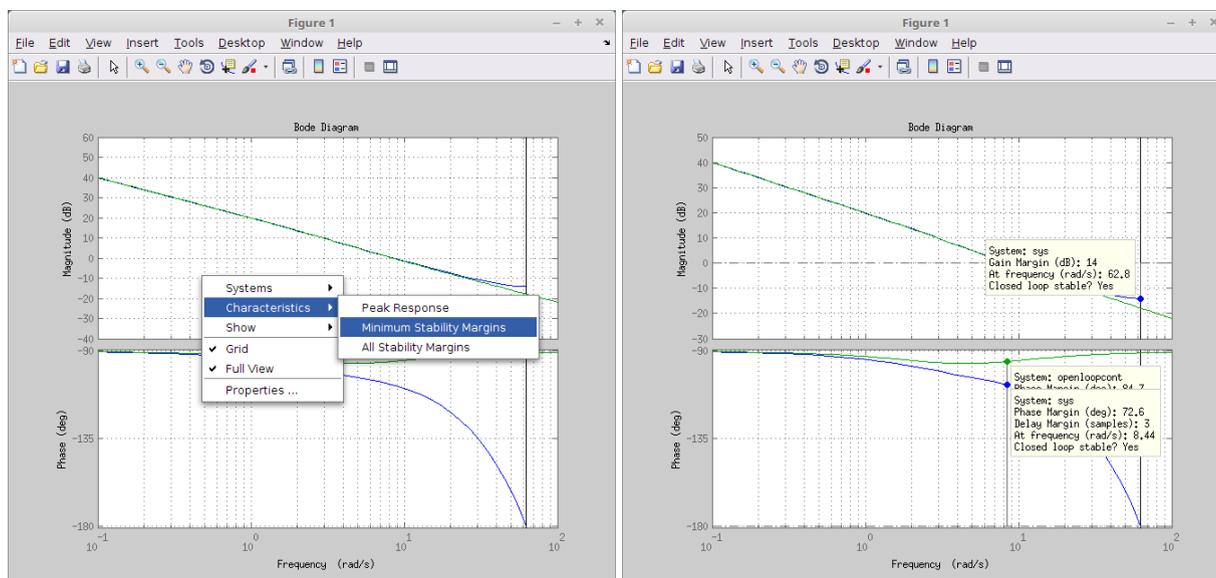


Figura 1: Estimativas de margem de ganho e de fase.

evidenciados. Ao clicar com o *mouse* nos pontos de *crossover* o sistema fornece a margem de ganho ou de fase associada a esse ponto. A Figura 1 ilustra esse processo.

Edite o script colocando os valores adequados que definem o seu controlador e a sua planta. Estime as margens de ganho e de fase para os seguintes valores de intervalos de amostragem  $T_a = 0.01, 0.05, 0.1$ seg. Analise os resultados obtidos.

### 4.3 Simulação do sistema de controle discreto - modelo Simulink

Nessa parte você necessitará de três arquivos:

- `ScriptDeInicializacaoPI_DigitalV2.m`: script que inicializa todas as variáveis necessárias para o modelo simulink.
- `ControladorPI_DigitalV2.mdl`: modelo simulink do sistema de controle digital com controlador PI discreto utilizando Transformação Bilinear. Veja Figura 2
- `plotgraficoPI_DigitalV2.m`: script que plota as variáveis salvas nas caixas laranja do modelo Simulink.

Inicialmente, modifique adequadamente no script `ScriptDeInicializacaoPI_DigitalV2.m` os valores das variáveis relativas ao controlador e à planta:  $K_p, T_i, K_m, T$ .

Os seguintes experimentos devem ser realizados:

1. Mantendo  $nbits = 12$  realizar simulações com os seguintes intervalos de amostragem  $T_a = 0.01, 0.05, 0.1$ seg. Analise os resultados obtidos.
2. Mantendo  $T_a = 0.05$ seg realizar simulações com os seguintes valores de número de bits de quantização  $nbits = 8, 10, 12$ . Analise os resultados obtidos.

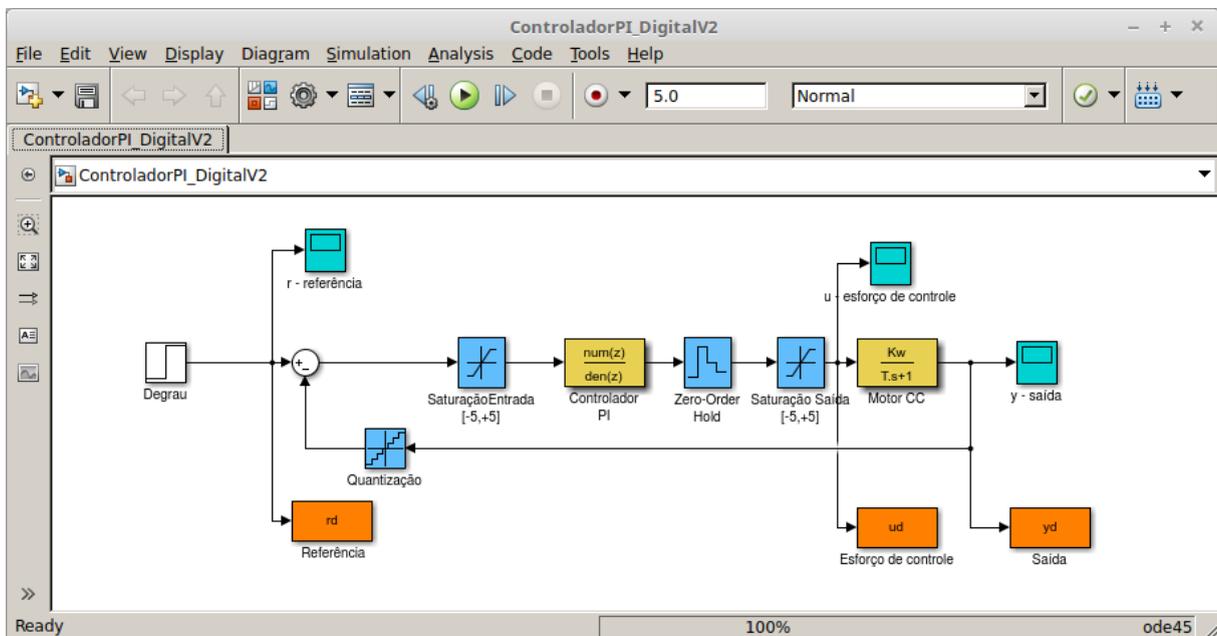


Figura 2: Modelo simulink do sistema de controle discreto.

#### 4.4 Testes de controle de velocidade no Módulo de Motor CC MS15

Nessa parte você deve utilizar o script `sistemadecontrolePI.m` que implementa um sistema de controle de velocidade para o Módulo de Motor CC MS15.

O controlador PI discreto é implementado através do método de Transformação Bilinear. O controlador é descrito através dos parâmetros da constante proporcional  $K_p$  e do tempo integral  $T_i$  (definidos no domínio  $s$ ).

Os seguintes experimentos devem ser realizados:

- Realizar experimentos com as seguinte frequências de amostragem:  $f_a = 10, 20, 100\text{Hz}$ . Analise os resultados obtidos.

```
function [vetorrk,vetoruk,vetoryk] = sistemadecontrolePI
Fa = 20;           % Sampling frequency
Ta = 1/Fa;        % Sampling time
Duration = 50;    % Duracao em segundos
NumberOfTasksToExecute = round(Duration/Ta)

% Create and configure timer object
tm = timer('ExecutionMode','fixedRate', ...           % Run continuously
           'Period',Ta, ...                          % Period = sampling time
           'TasksToExecute',NumberOfTasksToExecute, ... % Runs NumberOfTasksToExecute times
           'TimerFcn','@MyTimerFcn, ...              % Run MyTimerFcn at each timer event
           'StopFcn', @StopEverything);

% Setup da placa de aquisicao
ai=analoginput('nidaq','Dev1');
ao=analogoutput('nidaq','Dev1')
addchannel(ai,0:1);
addchannel(ao,0);
Set(ai,'ChannelSkewMode','Equisample')
ai.ChannelSkew

% inicializacao de variaveis do sistema de controle
rk = 0.0;
```

```

yk = 0.0;
uk = 0.0;
uk_1 = 0.0;
ek = 0.0;
ek_1 = 0.0;

vetorrk = zeros(NumberOfTasksToExecute+1,1);
vetorrk(1) = rk; % instante 0 -> k=1
vetoryk = zeros(NumberOfTasksToExecute+1,1);
vetoryk(1) = yk;
vetoruk = zeros(NumberOfTasksToExecute+1,1);
vetoruk(1) = uk;
k=2;

% Parametros do controlador PI
Kp = 1;
Ti = 0.2;

% Transformacao bilinear
a = Kp + 2*Kp*Ti/Ta;
b = Kp - 2*Kp*Ti/Ta;
c = 2*Ti/Ta;
a1 = a/c;
a2 = b/c;

% Start the timer
start(tm)
function MyTimerFcn(obj,event)

% Leitura da Porta A/D - e(k)
sample=getsample(ai);
rk = sample(1);
yk = sample(2);

% calculo do erro e(k)
ek = rk - yk;

% Calculo do controle
uk = u(k-1) + a1*e(k) + a2*e(k-1)

% Controle de Saturacao
if uk >= 5.0
    uk = 5.0;
else if uk <= -5.0
    uk = -5.0;
end
end

% Escrita na Porta D/A - u(k)
putsample(ao,uk);

% Salva os valores
vetorrk(k) = rk;
vetoryk(k) = yk;
vetoruk(k) = uk;

% Update de variaveis
k=k+1
uk_1 = uk;
ek_1 = ek;
end

function StopEverything(obj,event)
nsamples = length(vetorrk);
t=0.0:Ta:(nsamples-1)*Ta;
plot(t,vetorrk,'-',t,vetoruk,'-.',t,vetoryk,'--');
grid on
title('Referencia_r(k)_(-) / Esforco_de_controle_u(k)_(-) / Saida_da_planta_y(k)_(--)\')

```

```
    xlabel('tempo_(s)');  
    ylabel('tensao_(Volts)');  
    mensagem='acabou'  
end  
end % function controlador
```