Laboratório de Sistemas de Controle I

João Carlos Basilio
Lilian Kawakami Carvalho

Universidade Federal do Rio de Janeiro
Escola Politécnica – Depto. de Engenharia Elétrica
2012

Sumário

Capítulo 1

Introdução

Um projeto de um sistema de controle compreende, de uma maneira geral, as seguintes etapas [1, 2]:

1. Modelagem/identificação do sistema a ser controlado;

2. Projeto de um controlador que satisfaça as especificações de desempenho e estabilidade relativa exigidas;

3. Simulação utilizando computadores digitais;

4. Implementação do controlador no sistema real.

Note que as fases acima podem vir a constituir um ciclo com diversas iterações, uma vez que durante a modelagem do sistema a ser controlado (planta), várias hipóteses simplificadoras como, por exemplo, linearidade são feitas. Assim, ao se implementar um sistema de controle utilizando-se um controlador projetado a partir de um certo modelo matemático, o sistema real pode vir a ter um comportamento diferente daquele inicialmente previsto na simulação, isto é, as especificações de desempennho/estabilidade podem não ser satisfeitas quando da implementação do controlador no sistema real, embora tivessem sido plenamente verificadas nas simulações. Neste caso, torna-se necessário projetar um novo controlador ou até mesmo desenvolver um novo modelo matemático para a planta. Outro aspecto a ser observado é que, se durante a etapa de projeto/simulação as especificações de desempenho/estabilidade não forem atendidas para um determinado compensador com uma certa estrutura (controlador PI, por exemplo), uma nova estrutura para o controlador — ou até mesmo uma nova metodologia de projeto — deve ser adotada. Pode-se, até mesmo, em alguns casos, verificar se há alguma possibilidade de

se relaxar as especificações.

No Laboratório de Sistemas de Controle I, procuraremos abranger todas as fases do projeto de um sistema de controle por realimentação. O objetivo será controlar a velocidade angular de um motor de corrente contínua (referido no restante do texto como motor CC), cujo circuito equivalente está representado na figura 1.1, em que $v_a(t)$ representa a tensão nos terminais da armadura do motor e $v_t(t)$ a tensão nos terminais do tacômetro, que é proporcional à velocidade angular do gerador.

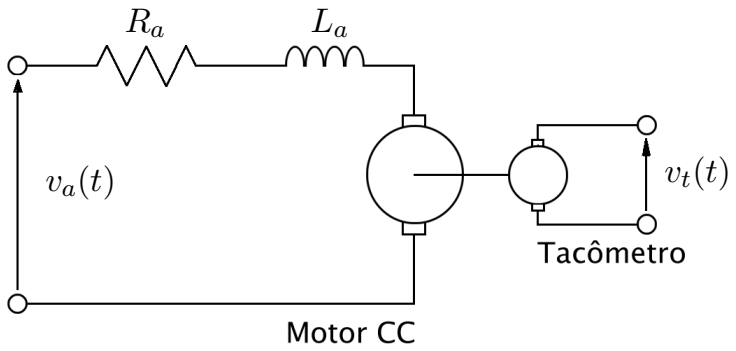


Figura 1.1: Circuito elétrico equivalente para o motor CC com sensor de velocidade (tacômetro)

Ao se formular um problema de controle, o primeiro passo é a definição da grandeza a ser controlada. Em nosso caso, a grandeza escolhida é a velocidade angular do motor. Assim, o problema de controle a ser resolvido neste laboratório pode ser enunciado da seguinte forma: projete um controlador de tal forma que o sistema realimentado tenha as seguintes características:

(i) seja estável;

(ii) tenha erro de estado permanente nulo para uma determinada velocidade de referência;

(iii) rejeite assintoticamente (quando o tempo tende para infinito) sinais externos de perturbação que, no sistema em estudo, são ocasionados pelo aumento do torque resistivo no eixo do motor.

No projeto de sistemas de controle, o engenheiro é geralmente assistido por programas computacionais tais como o MATLAB [3]. Em nosso projeto utilizaremos o MATLAB como ferramenta auxiliar de projeto e, em particular, uma de suas ferramentas, o SIMULINK, que será importante na simulação do sistema projetado, na

geração de sinais a serem aplicados no motor CC e na aquisição de sinais necessários provenientes do motor e do sistema de controle projetado. Uma breve introdução à programação em ambiente MATLAB será apresentada no capítulo 2. No capítulo 3 será desenvolvido um modelo matemático para a planta e em seguida os parâmetros da função de transferência do sistema serão identificados utilizando-se métodos de resposta ao degrau e de resposta em frequência. No capítulo 4 será feito o projeto de um sistema de controle de velocidade para o motor CC cujos parâmetros foram determinados de acordo com o capítulo anterior. Serão também realizadas simulações com o objetivo não só de ilustrar os benefícios da realimentação como também verificar o desempenho esperado do sistema com o controlador projetado. Finalmente, no capítulo 5 será feita a implementação do controlador de velocidade desenvolvido no capítulo 4.

É importante ressaltar que a abordagem aqui apresentada, embora aplicada a um sistema específico, pode ser utilizada no projeto de sistemas de controle realimentado para outros sistemas físicos. Assim sendo, caso o aluno venha a se deparar com um processo diferente daquele que estamos considerando neste laboratório, poderá seguir os mesmos passos que descritos neste laboratório para a solução do problema em questão.

Capítulo 2

Programando em MATLAB

Uma das principais razões do sucesso da programação em ambiente MATLAB nos meios científicos é certamente a possibilidade que o usuário tem de manipular matrizes, vetores, polinômios e diversas funções matemáticas de forma idêntica a que está acostumado a fazer no dia-a-dia. A ideia central dessa linguagem é transportar para um ambiente computacional os mesmos símbolos e convenções da matemática – daí o nome MATLAB. Para tanto, o MATLAB não só dispõe de funções pré-definidas como também permite que o usuário desenvolva suas próprias funções. Isto fez com que essa linguagem crescesse, uma vez que mais e mais funções foram desenvolvidas, aumentando cada vez mais o seu potencial e a sua aplicabilidade a diversos campos da ciência.

No projeto de sistemas de controle, o MATLAB é também uma poderosa ferramenta. Além de realizar cálculos que, em geral, são longos, tediosos e com grande probabilidade de ocorrência de erro quando realizados pelo ser humano, o MATLAB tem uma excelente interface gráfica que auxilia o engenheiro de controle a verificar se o sistema compensado com o controlador projetado satisfaz as especificações de projeto. Outro aliado importante do engenheiro de controle é o programa SIMULINK, que faz simulações de sistemas lineares e não-lineares, discretos e contínuos no tempo. Quando se dispõe de uma placa de conversão analógica/digital e digital/analógica, a ferramenta do Simulink denominada Real-Time Windows Target provê os meios para estabelecer uma comunicação entre os diagramas de blocos desenvolvidos no Simulink e a planta real, permitindo, assim, que sinais gerados no Simulink sejam aplicados à planta e também a aquisição de sinais correspondentes

às variáveis observadas (medidas) na planta.

Neste capítulo apresentaremos alguns tópicos sobre programação em ambiente MATLAB visando o seu uso no desenvolvimento de programas para identificação de sistemas e no projeto de sistemas de controle. Evidentemente há muito mais do que será apresentado aqui. Recomenda-se ao aluno mais interessado consultar os manuais do MATLAB e do SIMULINK [3].

2.1 Modos de programação

Ao iniciar o MATLAB, estando o usuário em ambiente WINDOWS, será aberta uma janela como aquela mostrada na figura 2.1. Na parte superior desta janela, à direita da opção "Current Folder" está indicado o diretório de trabalho, isto é, onde os arquivos Matlab desenvolvidos e as variáveis definidas pelo usuário serão armazenados. Caso o usuário deseje modificar o diretório de trabalho basta clicar no botão "..." e escolher o novo diretório. Os arquivos existentes no diretório de trabalho e os seus subdiretórios estão listados na subjanela da esquerda, que é denominada "Current Folder". Na subjanela central, denominada "Command Window",

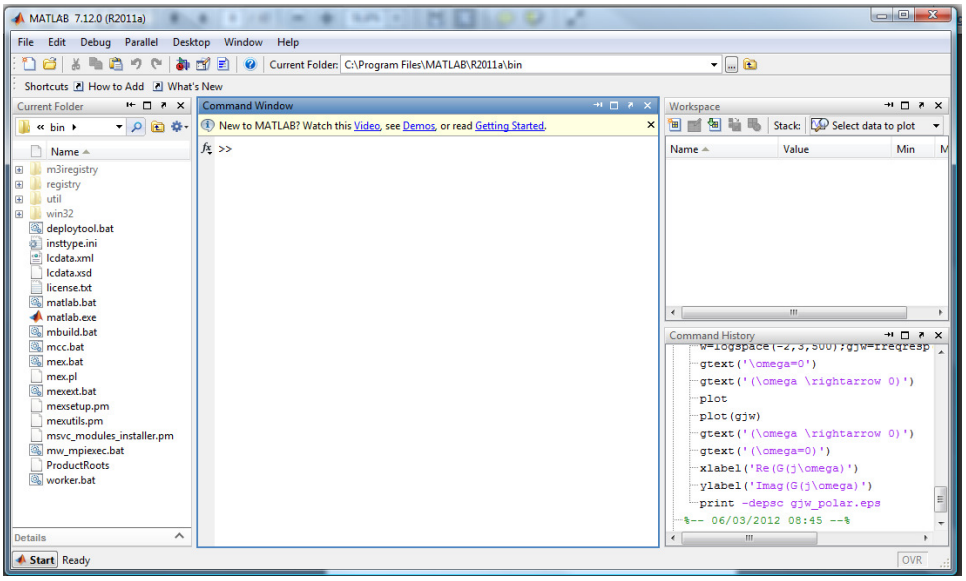


Figura 2.1: Janela inicial do Matlab no Windows

o usuário pode definir as variáveis e digitar os comandos referentes às operações matemáticas que devem ser realizadas com o auxílio do Matlab. As variáveis definidas pelo usuário na janela "Command Window" são listadas na subjanela superior direita denominada "Workspace". Finalmente, abaixo desta subjanela encontra-se a subjanela "Command History", onde são listados todas as linhas de comando já digitadas no "Workspace".

A programação em ambiente MATLAB pode ser feita nas seguintes formas:

1. Modo imediato;
2. Utilizando um arquivo de comandos;
3. Utilizando arquivos de funções.

No modo imediato, uma linha de comando é executada tão logo a tecla `<enter>` é pressionada. Com um arquivo de comandos pode-se executar uma série de instruções como se estivesse no modo imediato. A vantagem do arquivo de comandos sobre o modo imediato é que todos os comandos ficam guardados para uso futuro. A única desvantagem desse tipo de arquivo é que, rigorosamente falando, o usuário continua no modo imediato e, portanto, todas as variáveis que aparecerem em um arquivo de comando farão parte do espaço de trabalho do usuário. Isto aumenta a área de memória em uso pelo MATLAB e pode ter o inconveniente de alterar o valor de alguma variável caso um novo valor seja atribuído a ela quando da execução do arquivo de comando. Este problema não ocorre quando se utiliza um arquivo de funções. Neste caso o MATLAB define uma nova área de memória para armazenamento das variáveis a serem definidas no arquivo de funções, permitindo assim que variáveis de mesmo nome que aquelas em uso no espaço de trabalho sejam definidas em um arquivo de funções sem, contudo, alterar o valor daquelas definidas pelo usuário. Essa área de memória é apagada assim que a função que está sendo executada é finalizada.

2.1.1 Modo imediato

Uso de vírgula(,), ponto-e-vírgula (;) e reticências (...)

Comandos em uma mesma linha de comando podem ser separados por vírgula (,) ou ponto-e-vírgula (;) produzindo o mesmo efeito quanto à execução do comando, exceto que no primeiro caso o resultado da operação é mostrado na tela. Caso uma linha de comandos seja demasiadamente longa, ela pode ser continuada na linha

seguinte bastando para isso que a linha anterior termine com reticências seguida de `<enter>`. Neste caso o MATLAB somente irá executar a sequência de comandos quando a tecla `enter` for pressionada sem que tenha sido precedida de reticências.

Exemplo 2.1

» `a=2; b=5; <enter>`. *Neste caso, o MATLAB armazena 2 e 5 nas posições de memória* `a` *e* `b`*, respectivamente, porém não exibe os valores armazenados das variáveis.*

» `a=2, b=5 <enter>`. *Esta linha de comandos produz o mesmo efeito que a linha de comando anterior, exceto que a mensagem* `a=2` *e* `b=5` *é exibida na tela.*

» `c = a+b; <enter>`. *Aqui o MATLAB armazena o valor 7 na posição de memória* `c`*, mas não exibe na tela o seu conteúdo.*

Caso se deseje, posteriormente, saber o valor de uma determinada variável, basta digitar o nome dessa variável seguida de `<enter>`. *Por exemplo:*

» `c <enter>` *faz com que o MATLAB exiba na tela a mensagem* `c = 7`. □

Carregando e salvando arquivos de variáveis

O MATLAB armazena variáveis em um arquivo `.mat`. Para salvar as variáveis, basta selecionar na barra de ferramentas a opção `File`, escolher `Save Workspace As` na aba que se abrirá e definir o diretório e o nome do arquivo de acordo com as regras usuais do Windows Explorer para criação de arquivos. Caso o usuário não atribua um nome para o arquivo, o MATLAB salvará as variáveis do "Workspace" no arquivo `matlab.mat`. Deve ser ressaltado que caso já exista um arquivo como o mesmo nome que o escolhido pelo usuário, o arquivo já existente será apagado. O MATLAB, porém, pegunta se o usuário, de fato, deseja substituir o arquivo existente pelo atual.

Para carregar as variáveis armazenadas num arquivo `.mat` para o "Workspace", basta clicar no arquivo `.mat` desejado que se encontra listado na subjanela "Current Folder" e arrastá-lo para a janela "Command Window". Quando o arquivo desejado não se encontrar no diretório de trabalho, deve-se utilizar a opção `File` da barra de ferramentas e escolher `Import Data` na aba que será aberta. A partir daí basta seguir a sequência usual do Windows Explorer para a abertura de um arquivo. Deve-se tomar cuidado ao se realizar a operação "Import Data" pois, uma vez que variáveis carregadas de mesmo nome que variáveis já existentes no "Workspace"

substituem as já existentes. O usuário, contudo, tem a opção de alterar o nome das variáveis que estão sendo carregadas durante a operação "Import Data".

Exercício 2.1

Realize as operações a seguir no MATLAB, observando ao final de cada uma delas os valores das variáveis no "Workspace".

1. *Defina três variáveis* `a`, `b` *e* `c` *da seguinte forma:* `a=[1 2 3]`, `b=[3 5 7]'` *e* `c=a*b`
2. *Salve as variáveis no arquivo* `aula1v.mat`
3. *Salve somente as variáveis* `a`, `b` *no arquivo* `aula1ab.mat`
4. *Apague a variável* `c` *do seu "Workspace"*
5. *Apague, agora, as variáveis restantes do seu "Workspace" com um único comando*
6. *Carregue o arquivo* `aula1ab.mat`
7. *Mude a variável* `b` *para* `b=2*b`
8. *Carregue somente a variável* `b` *do arquivo* `aula1v.mat`
9. *Repita as operações 5, 6 e 7*
10. *Carregue somente a variável* `b` *do arquivo* `aula1v.mat`*, renomeando-a para* `d`

2.1.2 Arquivo de comandos

Os comandos editados em um arquivo de comandos podem ser executados diversas vezes. Para isso, basta digitar o nome do arquivo, sem a terminação `.m`, que o MATLAB executará todos as linhas de comandos do arquivo como se estivesse no modo imediato.

Para a criação de um arquivo de comandos, o primeiro passo é criar um novo arquivo utilizando o ícone "New File". Em seguida, edite o arquivo `<arquivo>.m` com comandos a serem executados como se estivesse no modo imediato. Após alguma modificação no arquivo `<arquivo>.m`, deve-se salvar novamente esse arquivo,

para que a alteração possa ser considerada pelo MATLAB na próxima vez que esse arquivo for executado. Para executar os comandos do arquivo `<arquivo>.m` basta digitar

```
» <arquivo> <enter>
```

ou clicar no ícone "Save and Run (F5)" .

Exemplo 2.2

Suponha que no arquivo `seq1.m`*, sejam editados os seguintes comandos:*

```
c=[1:2:10];d=[2:2:10];
e=c';f=c'*d;g=c*d';
h=c.*d;l=c./d;
```

Ao se digitar, na janela de comandos do MATLAB,

```
» seq1 <enter>
```

o programa MATLAB, inicialmente, verifica se existe no espaço de trabalho alguma variável denominada `seq1`*. Caso não encontre, MATLAB então procura no diretório de trabalho por um arquivo de comandos de nome* `seq1.m`*. Encontrando o arquivo, executa os comandos editados nele. Caso contrário, procura por um arquivo de comandos* `seq1.m` *em todos os caminhos de busca definidos utilizando-se a opção "Set Path" do menu do "File" da barra de ferramentas. Caso o MATLAB não encontre um arquivo* `seq1.m`*, exibirá na tela a mensagem*

```
??? Undefined function or variable seq1.
```

É importante ressaltar que se o arquivo `seq1.m` *for gravado em dois diretórios, o MATLAB executará aquele que estiver armazenado no diretório pertencente ao caminho de busca que aparece em primeiro lugar no "Set Path".* □

2.1.3 Arquivos de funções

Quando um arquivo de comandos é executado, todas as variáveis nele definidas passam a fazer parte do espaço de trabalho. Além de aumentar a área de memória utilizada pelo MATLAB (o que pode muitas vezes torna lenta a execução de comandos, visto que o MATLAB pode vir a ter que armazenar variáveis no disco rígido), isto faz com que o usuário tenha no seu espaço de trabalho um número, às vezes, excessivamente grande de variáveis. Outro aspecto a ser ressaltado é que, ao se usar arquivos de comandos, deve-se ter o cuidado de usar nomes de variáveis diferentes daquelas do espaço de trabalho, no caso de não se querer modificar o conteúdo delas.

Esses problemas são, em parte, solucionados com o emprego de arquivos de funções. Ao se executar uma função, o MATLAB define um novo espaço de trabalho diferente daquele definido inicialmente pelo usuário. Isto permite que variáveis de mesmo nome sejam definidas simultaneamente no espaço de trabalho e num arquivo de funções. O espaço de trabalho definido ao se executar uma função irá, como no caso da execução de um arquivo de comandos, aumentar a área de memória utilizada pelo MATLAB. Contudo, diferentemente do que ocorre quando se usa um arquivo de comandos, essa área será liberada após a execução da função.

Para se criar uma função, procede-se da seguinte forma:

1. Cria-se um arquivo `<função>.m`;

2. A primeira linha de execução desse arquivo deve ser:

```
function [y1,y2,...,yn] = <função>(x1,x2,...,xm);
```

onde `x1,x2,...,xm` são as variáveis de entrada e `y1,y2,...,yn` são as variáveis de saída. Note que o nome da função, definido em `<função>`, deve ser o mesmo do arquivo onde ela foi editada.

3. Antes do primeiro comando a ser executado, pode-se escrever alguns comentários para orientação dos possíveis usuários daquela função. Toda linha de comentário se inicia com o sinal `%`.

4. Ao se digitar

```
» help <função> <enter>
```

são exibidos, na janela aberta para o MATLAB, os comentários iniciais do arquivo `<função>.m`, isto é, aquelas iniciadas com `%` que precedem uma linha de comandos ou uma linha totalmente em branco.

Exemplo 2.3 *Crie um arquivo de funções para calcular a norma euclideana (norma-2) de um vetor* $\underline{x} = [\ x_1\ \ x_2\ \ \ldots\ \ x_n\ ]^t$.

Inicialmente, lembre-se de que a norma euclideana de um vetor é definida como:

$$\| \underline{x} \|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = (\underline{x}^t \underline{x})^{1/2} \tag{2.1}$$

A função que calcula $\| \underline{x} \|_2$ *deve, portanto, executar a equação (2.1), o que requer que a variável de entrada seja um vetor (*`x`*) de dimensão n e a saída um escalar (*`y`*) representativo da norma-2 de* $\underline{x}$. *Se* `norma2.m` *for o nome do arquivo onde esta função será editada, então esse arquivo poderia ter o conteúdo mostrado na figura 2.2.*

```
% y=norma2(x) :  calcula a norma euclideana de um vetor
%  sendo:  x  :  vetor de entrada (vetor coluna)
%          y  :  norma euclideana de x
%
function y=norma2(x);
y=sqrt(x'*x);
```

Figura 2.2: Listagem da função `norma2.m`.

Observe que digitando-se

» `help norma2 <enter>`

as 4 primeiras linhas do arquivo serão exibidas na janela do MATLAB, porém o símbolo `%` não será exibido. Por outro lado, o comando

» `x=[1 2 2]';y=norma2(x);<enter>`

faz com que MATLAB execute a função `norma2` para o vetor `x`, armazenando na variável `y` o valor 3. Verifique!

Para que o resultado da função acima seja realmente a norma euclideana do vetor $\underline{x}$, `x` deve ser um vetor coluna. Caso contrário `x'*x` será uma matriz e a saída `y` será uma matriz $m \times m$ cujos elementos são as raízes quadradas de $x_i x_j$, onde x_i e x_j denotam as componentes i e j do vetor $\underline{x}$. Uma outra particularidade da função acima é o uso do comando `sqrt`, que é uma função pré-definida pelo MATLAB. Em um arquivo de função podem ser usadas outras funções quer sejam definadas anteriormente pelo usuário ou pré-definidas pelo MATLAB. Essas últimas permitem a manipulação de matrizes, vetores, polinômios e outras funções matemáticas de uma forma robusta, isto é por algoritmos que não são susceptíveis a problemas de mau condicionamentos.

Na seção seguinte serão apresentadas algumas funções pré-definidas pelo MATLAB. Essas funções foram escolhidas tendo em vista sua importância no desenvolvimento de programas para o projeto de sistemas de controle.

2.2 Funções pré-definidas pelo MATLAB

A característica principal do MATLAB é tratar indistintamente escalares, vetores e matrizes como sendo matrizes. Por exemplo, a variável `a=3` é para o MATLAB

uma matriz de dimensão 1×1; um vetor de dimensão n é, na verdade, uma matriz de dimensões $1 \times n$ ou $n \times 1$, dependendo se o vetor considerado é linha ou coluna. Assim sendo, tanto o argumento (entrada) de uma função como o valor que ela retorna (saída) são matrizes, embora para o usuário tratam-se simplesmente de vetores ou até mesmo escalares.

Para definir uma matriz, deve-se fornecer cada um dos seus elementos. Por exemplo, a matriz

$$A = \begin{bmatrix} 1 & 2,2 & 3 \\ 4,6 & 5 & 6 \end{bmatrix}$$

pode ser armazenada na variável `a` das seguintes formas:

(i) Fornecendo-se cada um dos elementos da matriz:

```
» a(1,1)=1;a(1,2)=2.2;a(1,3)=3; <enter>
» a(2,1)=4.6;a(2,2)=5;a(2,3)=6; <enter>
```

(ii) Entrando com os elementos da matriz entre colchetes (`[ ]`), sendo os elementos de uma mesmas linha separados por vírgula (`,`) ou espaços em branco e as linhas da matriz por ponto-e-vírgula (`;`):

```
» a=[1,2.2,3;4.6,5,6] <enter>
» a=[1 2.2 3;4.6 5 6] <enter>
```

Deve-se tomar o cuidado para, no caso de números com parte decimal, separar a parte inteira da parte decimal usando ponto (`.`);

O acesso a qualquer elemento de uma matriz que tenha sido anteriormente definida, e que ainda esteja armazenada no espaço de trabalho, é feito de forma semelhante, isto é, fornecendo-se a posição do elemento desejado. Suponha, então, que os elementos de uma matriz $M : p \times q$ tenham sido armazenados na variável `m`.

(i) Para se ter acesso ao elemento (i, j) de `m` faz-se `m(i,j)`. Por exemplo, para modificar o elemtno $(1, 3)$ da variável `a`, acima, fazendo-o igual a 7, basta digitar:

```
» a(1,3)=7 <enter>
```

Para se armazenar o elemento $(1, 2)$ na variável `x`, faz-se:

```
» x = a(1,2) <enter>
```

(ii) Fazendo-se `m(vetorlinha,vetorcoluna)` forma-se uma submatriz de uma dada matriz `m`, formada pelos elementos comuns às linhas cujos índices são dados no vetor `vetorlinha` e às colunas indicadas pelas componentes do vetor `vetorcoluna`. Por exemplo, para se tomar os elementos da interseção das linhas 1 e 2 e das colunas 2 e 3 de `a` e armazená-las em `x`, deve-se fazer:

`» x = a([1 2],[2 3]) <enter>`

(iii) Para ter acesso à coluna j da matriz `m` faz-se `m(:,j)`. Por exemplo, para atribuir a terceira coluna da matriz `a`, acima, à variável `x`, faz-se:

`» x = a(:,j) <enter>`

(iv) Analogamente, a coluna i de uma matriz `m` pode ser acessada fazendo-se `m(i,:)`. Por exemplo, suponha que se queira armazenar a linha 2 da matriz `a`, acima, em `x`. Para tanto, deve-se fazer

`» x = a(i,:)  <enter>`

Algumas matrizes não requerem que o usuário entre com todos os seus elementos. São elas: as matrizes identidade e diagonal, a matriz cujos elementos são todos iguais a 1 ou zero. A atribuição dessas matrizes a variáveis é feita da seguinte forma:

(i) `id = eye(m)` : atribui-se à variável `i` uma matriz identidade de ordem m;

(ii) `d = diag([d1 d2 ...  dm])` : forma uma matriz diagonal `d`, $m \times m$, cujos elementos da diagonal principal são `d1,d2,...,dm`;

(iii) `um = ones(m,n)` : cria uma variável `um`, $m \times n$, em que todos os elementos são iguais à unidade;

(iv) `z = zeros(m,n)` : forma uma matriz `z`, $m \times n$, cujos elementos são todos iguais a zero.

O MATLAB possui um conjunto de funções pré-definidas que permitem manipular matrizes e vetores de semelhante a que o engenheiro está acostumado a fazer no seu dia-a-dia. Estas funções são apresentadas a seguir.

2.2.1 Funções para manipulação de matrizes

Sejam

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \text{ e } B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1q} \\ b_{21} & b_{22} & \dots & b_{2q} \\ \vdots & \vdots & & \vdots \\ b_{p1} & b_{p2} & \dots & b_{pq} \end{bmatrix}$$

matrizes de dimensões $m \times n$ e $p \times q$, respectivamente, com elementos pertencentes ao conjunto dos números complexos. Suponha que as matrizes A e B tenham sido armazenadas nas variáveis `a` e `b`, respectivamente. Então:

(i) `[m,n]=size(a)` : armazena, respectivamente, nas variáveis `m` e `n`, o número de linhas e de colunas da matriz `a` ;

`m=size(a,1)` : armazena na variável `m` o número de linhas de `a`;

`n=size(a,2)` : armazena em `n` o número de colunas de `a`.

(ii) `c=a*b` : calcula o produto das matrizes `a` e `b` (se $n = p$) e armazena na variável `c`. Quando $n \neq p$, MATLAB retorna uma mensagem de erro.

(iii) `c=a.*b` e `c=a./b` : forma uma matriz C tal que $c_{ij} = a_{ij}b_{ij}$ e $c_{ij} = a_{ij}/b_{ij}$, respectivamente, e armazena o resultado na variável `c`. Este comando requer $m = p$ e $n = q$.

(iv) `c=inv(a)` : calcula a inversa da matriz armazenada na variável `a` (se $m = n$) e armazena na variável `c`.

(v) `c=det(a)` : calcula o determinante da matriz armazenada em `a`, armazenando o resultado na variável `c`.

(vi) `c=a+b` e `c=a-b` : computa, respectivamente, a soma e a subtração das matrizes `a` e `b` e as armazena em `c`.

(vii) `[w,l]=eig(a)` : retorna a matriz `w`, cuja i-ésima coluna é formada pelo o i-ésimo autovetor de `a` associado ao i-ésimo autovalor e uma matriz diagonal `l`, cujo elemento (i, i) corresponde ao i-ésimo autovalor de `a`.

`l=eig(a)` : retorna um vetor de dimensão n cujas componentes são os autovalores de `a`.

2.2.2 Funções matemáticas elementares

Seja $Z : m \times n$, uma matriz cujo elemento i, j é $z_{ij} =\mid z_{ij} \mid e^{j\theta_{ij}} = x_{ij} + jy_{ij}$ e suponha que a matriz Z tenha sido armazenada na variável `z`. Tem-se que:

(i) `w = abs(z)` retorna uma matrix `w` cujo elemento (i, j) é $w_{ij} =\mid z_{ij} \mid$;

(ii) `w = angle(z)` retorna uma matriz `w` cujo elemento (i, j) é $w_{ij} = \theta_{ij}$, em radianos;

(iii) `w = real(z)` é uma matriz onde $w_{ij} = x_{ij}$;

(iv) `w = imag(z)` é tal que $w_{ij} = y_{ij}$;

(v) `w = conj(z)` retorna uma matriz `w` tal que $w_{ij} = x_{ij} - jy_{ij}$;

(vi) `w = z'` calcula uma matriz $W = Z^* = \bar{Z}^t$ (associada da matriz Z), isto é $w_{ij} = x_{ji} - jy_{ji}$ e armazena na variável `w`;

(vii) `w = exp(z)` retorna uma matriz cujo elemento (i, j) é $w_{ij} = \exp(z_{ij})$;

(viii) `w = log(z)` retorna uma matriz tal que $w_{ij} = \ln(x_{ij} + jy_{ij})$;

(ix) `w = log10(z)` produz uma matriz em que $w_{ij} = \log(x_{ij} + jy_{ij})$;

(x) `w=sin(z)`, `w=cos(z)`, `w=tan(z)`, `w=acos(z)`, `w=asin(z)` e `w=atan(z)` calculam matrizes W tais que $w_{ij} = \text{Sen}\ (z_{ij})$, $w_{ij} = \text{Cos}\ (z_{ij})$, $w_{ij} = \text{tg}\ (z_{ij})$, $w_{ij} = \text{arc Cos}\ (z_{ij})$, $w_{ij} = \text{arc Sen}\ (z_{ij})$ e $w_{ij} = \text{arc tg}\ (z_{ij})$, respectivamente, armazenando o resultado na variável `w`. Note que: *(a)* as funções `sin`, `cos` e `tan` requerem que os argumentos sejam expressos em radianos; *(b)* os ângulos retornados pelas funções `acos`, `asin` e `atan` são expressos em radianos.

É importante salientar os seguintes pontos relativos ao uso dos comandos acima:

(1) quando se desejar obter a transposta de uma matriz e não o conjugado complexo da sua transposta (associada), deve-se proceder de uma das seguintes formas: `w = z.'` ou então `w = conj(z')`;

(2) deve-se tomar cuidado ao usar os comandos `log` e `log10` — o primeiro calcula o logaritmo neperiano, enquanto o último retorna o logaritmo decimal. Os exercícios propostos 2.1 e 2.2, ao final deste capítulo, servirão para ilustrar o uso dos comandos apresentados acima.

2.2.3 Manipulação de vetores

Para o MATLAB, vetores são, na verdade, matrizes nas quais uma das dimensões (número de linhas ou colunas) é igual a 1. Portanto, podem ser manipulados da mesma forma que matrizes. Por exemplo, o vetor $\underline{v} = \left[\begin{array}{ccc} 1 & 2,5 & 7 \end{array}\right]$ pode ser definido das seguintes formas:

(1.i) `» v(1,1)=1;v(1,2)=2.5;v(1,3)=7; <enter>`

(1.ii) `» v=[1 2.5 7]; <enter>`

(1.iii) `» v=[1,2.5,7]; <enter>`

Caso se tratasse de um vetor coluna, ($\underline{w} = \underline{v}^t$), poder-se-ia definir uma variável `w` como se segue:

(2.i) `» w(1,1)=1;w(2,1)=2.5;w(3,1)=7; <enter>`

(2.ii) `» w=[1 2.5 7]'; <enter>`

(2.iii) `» w=[1;2.5;7]; <enter>`

Apesar de considerar todas as variáveis como sendo matrizes, o MATLAB é capaz de reconhecer quando se trata de um vetor. Para isso, basta que uma das dimensões seja igual a 1 ou que, ao se definir o vetor, apenas uma das dimensões seja dada. Neste último caso, o MATLAB armazenará o vetor como sendo um vetor linha. Desta forma, para se definir os vetores `v` e `w`, acima, poder-se-ia substituir as intruções *(1.i)* e *(2.i)* por:

» `v(1)=1;v(2)=2.5;v(3)=7;w=v'; <enter>`.

Outro aspecto importante na definição de um vetor é quando as componentes formam uma sequência. Por exemplo, para armazenar a sequência

$$f_n = \left[\begin{array}{cccc} a_1 & a_1 + r & a_1 + 2r & \ldots \quad a_n \end{array} \right],$$

em que $a_n = a_1 + nr$, na variável `fn` faz-se

» `fn=[a1:r:an] <enter>`;

Deve-se salientar que, caso a_n não fosse um elemento da sequência, o último elemento do vetor `sf` seria o maior valor tal que $a_1 + kr < a_n$, para um dado k. É importante lembrar que pode também ser feito o armazenamento de uma sequência decrescente. Para tanto, a_1 deve ser maior que a_n e r deve ser negativo.

MATLAB também possui funções especiais para a manipulação de vetores. Sejam

$$\begin{aligned} \underline{x} &= \left[\begin{array}{cccc} x_1 & x_2 & \ldots & x_n \end{array} \right] \\ \underline{y} &= \left[\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_m \end{array} \right] \end{aligned}$$

vetores de dimensões n e m, respectivamente, e suponha que $\underline{x}$ e $\underline{y}$ estejam armazenados nas variáveis `x` e `y`. Os seguintes comandos são especialmente usados na manipulação de vetores:

(i) `length(x)` e `length(y)` retornam as dimensões dos vetores `x` e `y` (n e m, respectivamente);

(ii) `norm(x,p)` retorna a norma-p do vetor $\underline{x}$ ($\|\underline{x}\|_p$). Deve ser ressaltado que a função `norm` permite que o vetor, cuja norma-p se deseja calcular, seja linha ou coluna. Lembre-se de que, a norma-p de um vetor $\underline{x}$ é definida como

$$\|\underline{x}\|_p = (\sum_{i=1}^{n} \mid x_i \mid^p)^{1/p}$$

e, portanto:

$$\begin{aligned} \|\underline{x}\|_1 &= \sum_{i=1}^{n} \mid x_i \mid \\ \|\underline{x}\|_2 &= (\sum_{i=1}^{n} \mid x_i \mid^2)^{1/2} \end{aligned}$$

(iii) `[mx,i]=max(x)`. Se `x` é um vetor cujas componentes são números reais, então `mx` é o maior valor de `x` e `i` corresponde à posição onde o máximo ocorre. Caso `x` tenha duas componentes onde os valores máximos ocorrem, `i` corresponderá à componente onde o valor máximo ocorre primeiro. Quando `x` é um vetor complexo, `mx` correponderá à componente de maior valor absoluto de `x` e `i` correponderá ao índice dessa componente.

(iv) `hflip=fliplr(x)` e `vflip=flipud(y)` retornam os seguintes vetores:

$$\underline{h}_{\text{flip}} = \begin{bmatrix} x_n & x_{n-1} & \dots & x_2 & x_1 \end{bmatrix}$$

$$\underline{v}_{\text{flip}} = \begin{bmatrix} y_m \\ y_{m-1} \\ \vdots \\ y_2 \\ y_1 \end{bmatrix}$$

Observação 2.1 *Às vezes é necessário acessar a última componente de um vetor* `x`. *Isso pode ser feito de duas formas: a primeira, que utiliza a função length, isto é,* `x(length(x))`, *e a segunda que utiliza a palavra reservada* `end`, *isto é,* `x(end)`. *Por exemplo, suponha que tenha sido definido o vetor* `x=[1 3 5 9 -2]`. *O comando*

```
» y=[3 x(end)] <enter>
```

faz com que o MATLAB crie o vetor $\underline{y} = \begin{bmatrix} 3 & -2 \end{bmatrix}$. *Por outro lado, o comando*

```
» z=[3 x(2:end)] <enter>
```

cria o vetor $\underline{y} = \begin{bmatrix} 3 & 3 & 5 & 9 & -2 \end{bmatrix}$ □

Os exemplos a seguir ilustram algumas das funções utilizadas para a manipulação de vetores definidas acima.

Exemplo 2.4 *Escreva um arquivo do tipo funções para calcular a derivada de um polinômio de qualquer grau.*

Seja

$$p(x) = p_0x^n + p_1x^{n-1} + p_2x^{n-2} + \ldots + p_{n-1}x + p_n$$

um polinômio de grau n. Sabe-se que a derivada de $p(x)$ *é dada por:*

$$\frac{d}{dx} = np_0x^{n-1} + (n-1)p_1x^{n-2} + \ldots + 2p_{n-2}x + p_{n-1}$$

sendo, portanto, um polinômio de grau $n-1$*. A função* `pderiv`*, listada na figura 2.3, permite calcular a derivada de* $p(x)$ *em relação a* x*:*

```
% PDERIV : calcula a derivada de um polinomio p(x)
% plinha = pderiv(p)
%
% sendo:  p = [po p1 p2 ...  pn] os coeficientes do polinomio
%
function plinha = pderiv(p);
n=length(p)-1;plinha=[n:-1:1].*p(1:n);
```

Figura 2.3: Função `pderiv` que calcula a derivada de um polinômio de grau $n > 1$

Verifique, agora, se o programa está correto, calculando a derivada em relação a x *dos polinômios:* $p(x) = 5x^3 + 2x^2 - x + 1$ *e* $q(x) = 5$*.*

Exemplo 2.5 *Escreva uma função MATLAB para calcular o instante de pico, o percentual de ultrapassagem, o valor máximo e o valor de estado permanente da resposta ao degrau do sistema cuja função de transferência é*

$$G(s) = \frac{0,6618}{s^2 + 1,0069s + 0,6618}.$$

Considere que o intervalo de simulação seja de 0 *a* $20s$*.*

A função `indices` *listada na figura 2.4 permite encontrar os índices de desempenho desejados.*

Para executar a função desenvolvida no exemplo 2.5, deve-se fornecer os vetores `t` e `y`. Supondo que os instantes de simulação estejam espaçados de $0.1s$, tem-se que `t` será definido da forma seguinte:

```
» t=[0:0.1:20]; <enter>
```

O vetor `y` é obtido usando-se a função `step` que pertence à caixa de ferramentas de controle (*control toolbox*). A função `step` calcula a resposta ao degrau de um sistema, sendo conhecidos os polinômios do numerador e do denominador da função de transferência do sistema. O terceiro argumento da função `step` é o vetor contendo os instantes de simulação. As linhas de comando seguintes permitem obter `y` e também o tempo de pico (`tp`), o percentual de ultrapassagem (`po`), o valor máximo de `y` (`ymax`) e o valor de estado permanente de $y(t)$ (`yinf`):

```
% INDICES : calcula indices de desempenho de um sistema
% [tp,po,ymax,yss] = indices(t,y)
%
%  sendo  y     :  vetor contendo as repostas correspondentes aos
%                  instantes de simulacao armazenados no vetor t
%         t     :  vetor contendo os instantes de simulacao
%         tp    :  tempo de pico
%         po    :  percentual de ultapassagem
%         ymax  :  valor maximo da resposta
%         yinf  :  valor de estado permanente da resposta
%
function [tp,po,ymax,yinf] = indices(t,y);
% calculando o ponto de maximo e o instante onde maximo ocorre:
[ymax,i]=max(y);tp=t(i);
% calculando o valor de estado permanente:
yinf=y(length(y));
% calculando o percentual de ultrapassagem:
po=(ymax-yinf)/yinf*100;
```

Figura 2.4: Listagem da função `indices`.

```
» n=0.6618;d=[1 1.0069 0.6618];y=step(n,d,t); <enter>
» [tp,po,ymax,yinf]=indices(t,y); <enter>
```

É importante observar que quando o intervalo de simulação não for suficientemente grande, o valor de estado permanente `yinf` poderá não ser o correto, isto é, igual ao valor de $\lim_{t\to\infty} y(t)$. Portanto, o uso da função `indices` somente faz sentido quando se dispõe da representação gráfica de $t \times y$ ou então verificando-se o vetor `y` está, de fato, convergindo para o valor final esperado.

2.2.4 Funções polinomiais

Sejam $p(x) = p_0x^n + p_1x^{n-1} + \ldots + p_{n-1}x + p_n$ e $q(x) = q_0x^m + q_1x^{m-1} + \ldots + q_{m-1}x + q_m$ polinômios de graus n e m $(n \geq m)$, respectivamente. Para trabalhar com polinômios no MATLAB, devemos definir vetores cujas componentes são os coeficientes desses polinômios. Assim, para introduzir os polinômios $p(x)$

e $q(x)$ no MATLAB, devemos definir os vetores $\underline{p} = \left[\begin{array}{cccc} p_0 & p_1 & \ldots & p_{n-1} & p_n \end{array}\right]$ e $\underline{q} = \left[\begin{array}{cccc} q_0 & q_1 & \ldots & q_{n-1} & q_n \end{array}\right]$.

O MATLAB permite que polinômios sejam multiplicados ou divididos, que suas raízes sejam calculadas, ou o oposto, que se obtenha os coeficientes de um polinômio a partir de suas raízes, ou ainda, calcular o valor de $p(x)$ para diversos valores de x, conforme será descrito a seguir:

(i) Se desejarmos obter as raízes de $p(x)$ e armazená-las no vetor $\underline{y}$ devemos executar o seguinte comando: `y=roots(p)`.

(ii) O produto $r(x) = p(x)q(x)$ é efetuado pelo comando `r=conv(p,q)`.

(iii) Para dividir $p(x)$ por $q(x)$ e obter como quociente $a(x)$ e resto $r(x)$, devemos fazer: `[a,r]=deconv(p,q)`. Se não desejarmos obter o resto, basta fazer `a=deconv(p,q)`.

(iv) Suponha que desejamos saber o valor de $p(x)$ nos pontos x_1, x_2, ..., x_q. Para isso devemos, inicialmente, definir um vetor `x = [x1 x2 ... xq]` e, em seguida, fazer `y=polyval(p,x)`, obtendo `y = [p(x1) p(x2) ... p(xq)]`.

(v) Finalmente, para calcularmos as raízes do polinômio $p(x)$, utilizamos o comando `p=poly(raizes)`, que fornece um vetor `p` formado pelos coeficientes do polinômio cujas raízes são as componentes do vetor `raizes`. Por exemplo, `p=poly([-1 -2])` retorna o vetor `p=[1 3 2]`, formado pelos coeficientes do polinômio $p(x) = (x+1)(x+2) = x^2 + 3x + 2$.

Exemplo 2.6 *A função MATLAB cujo código está listado na figura 2.5 pode ser usada para calcular a resposta em frequência de uma função de transferência* $G(s) = n(s)/d(s)$ *para um dado um vetor de frequências* $\underline{\omega} = \left[\begin{array}{cccc} \omega_0 & \omega_1 & \ldots & \omega_q \end{array}\right]$, *no qual* ω_i, $i = 0, 1, \ldots, q$ *são as frequências para as quais desejamos calcular* $G(jw)$. *Os parâmetros de entrada são os vetores* `n` *e* `d` *formado pelos coeficientes dos polinômios do numerador e denominador de* $G(s)$, *respectivamente, e o vetor* `w`, *cujas componentes são as frequências* ω_i, $i = 0, 1, \ldots, q$.

Suponha, agora, que desejamos calcular a resposta em frequência do sistema cuja função de transferência seja

$$T(s) = \frac{20}{s^2 + 3s + 5}.$$

```
% RESPFREQ : calcula a resposta em frequencia de G(s)=n(s)/d(s)
% gjw = respfreq(n,d,w)
% sendo  :  n,d  :  vetores cujas componentes sao os coeficientes
%                   de n(s) e d(s)
%             w  :  vetor cujas componentes sao as frequencias para
%                   as quais se deseja calcular g(jw)=n(jw)/d(jw)
%
function gjw = respfreq(n,d,w);
jw = j*w;gjw=polyval(n,jw)./polyval(d,jw);
```

Figura 2.5: Função `respfreq` para o cálculo da resposta em frequência de uma função de transferência de $G(s)$.

Para tanto, o primeiro passo é definir um vetor `w` contendo as frequências nas quais se deseja calcular $T(j\omega)$. Supondo que a menor frequência seja $\omega_0 = 0$ e a maior seja igual a $\omega_f = 100$. Utilizando as seguintes linhas de comando:

```
» n=20;d=[1 3 5];w=[0:1:100];
» tjw=respfreq(n,d,w);plot(tjw) <enter>
```

obtém-se a resposta em frequência de $T(s)$ para os pontos considerados, cuja representação gráfica é dada na figura 2.6(a). Note que nas baixas frequências a representação gráfica é grosseira. Isto se deve ao fato de que, como $T(j\omega)$ varia muito rapidamente nessas frequências, os pontos imagens estarão bastante separados uns dos outros, fazendo com que as retas que os ligam sejam mais ressaltadas. Este problema não ocorrerá se o vetor `w` for definido usando o comando `logspace`, cuja forma geral é `logspace(exp1,exp2,n)`. Este comando, cria um vetor de `n` elementos espaçados logaritmicamente, cujo primeiro elemento é $10^{\exp 1}$ e o último $10^{\exp 2}$. Quando `n` for omitido, o comando acima criará um vetor composto de 50 elementos. Assim sendo, se substituirmos a linha de comandos acima por

```
» n=20;d=[1 3 5];w=logspace(-2,2,100);
» tjw=respfreq(n,d,w);plot(tjw) <enter>
```

teremos uma representação gráfica da resposta em frequência de $T(s)$ mais uniforme, conforme mostrado na figura 2.6(b).

Nas linhas de comando utilizadas para ilustrar o cálculo da resposta em frequência de $T(s)$ foi introduzido o comando `plot`. Vamos deixar o detalhamento desse

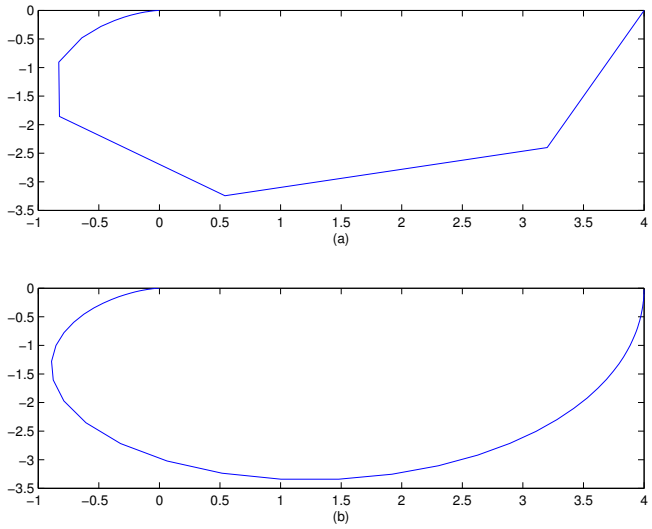


Figura 2.6: Representação gráfica de resposta em frequência de $T(s) = 20/(s^2 + 3s + 5)$ para `w=[0:1:100]` *(a)* e `w=logspace(-2,2,100)` *(b)*.

comando para a seção seguinte, onde iremos apresentar a sintaxe do comando `plot` na sua forma mais simples e diversas outras maneiras de utilizarmos esse comando.

2.3 Representação gráfica

A representação gráfica é uma importante característica do MATLAB. Utilizando o MATLAB, podemos representar gráficos na forma de curvas, diagramas de barras, histogramas, gráficos tridimensionais etc. Nesse laboratório iremos nos ater à representação gráfica que será mais utilizada em nosso curso. Outras formas de representação gráfica utilizando o MATLAB podem ser estudadas em [3].

2.3.1 O comando plot

Suponha que se deseje representar graficamente um ponto de coordenadas (x, y) no plano cartesiano. Isto pode ser feito em MATLAB utilizando-se o comando `plot`, cuja forma geral é a seguinte:

```
plot(x,y,'<cor><forma>')
```

em que `<cor>` define a cor usada na representação gráfica (conforme tabela 2.1(a)) e `<forma>` define como o ponto será representado, isto é se será usado um ponto, círculo, um 'x', uma cruz ou um asteriscos, de acordo com as primeiras cinco linhas da tabela 2.1(b).

Tabela 2.1: Cores usadas para representação gráfica em MATLAB (a); formas de se representar pontos e curvas em MATLAB (b)

(a)

Cor	Símbolo
Amarelo	`y`
Magenta	`m`
Azul claro	`c`
Vermelho	`r`
Verde	`g`
Azul escuro	`b`
Branco	`w`
Preto	`k`

(b)

Forma	Símbolo
Ponto	`.`
Círculo	`o`
x	`x`
Cruz	`+`
Asteriscos	`*`
Linha contínua	`-`
Linha pontilhada	`:`
Linha tracejada	`--`
Linha traço-pontilhada	`-.`

Por exemplo, caso se deseje representar graficamente o ponto de coordenadas $(-2, 1)$, na cor azul claro e com uma cruz, faz-se

```
» plot(-2,1,'c+') <enter>
```

É importante ressaltar que, caso a forma de representação não esteja presente no comando `plot`, o MATLAB assumirá, como padrão, o ponto (`.`). Por exemplo, caso a linha de comando acima seja substituída por

```
» plot(-2,1,'c') <enter>
```

então, o MATLAB utilizará o ponto (`.`) em vez de `+` para representar o ponto $(-2, 1)$ no plano cartesiano.

Suponha, agora, que se deseje representar graficamente os pontos de coordenadas (x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n) num mesmo gráfico. Para tanto, o primeiro passo é definir dois vetores, $\underline{x}$ e $\underline{y}$ por exemplo, cujas componentes são, respectivamente, as abscissas e as ordenadas dos pontos que se deseja representar. Em seguida, usar o comando `plot`, de acordo com a forma geral dada acima, onde `x` e `y` são agora vetores e não as coordenadas de um ponto, conforme consta da forma geral

do comando `plot`. Por exemplo, suponha que se queira representar graficamente os pontos $(-1, 1)$, $(0, 2)$, $(1, -1)$ e $(2, 2)$. A linha de comando seguinte permite representá-los na cor azul, unindo-os com retas:

```
» x=[-1,0,1,2];y=[1 2 -1 2];plot(x,y) <enter>
```

Note que, no comando `plot` acima, não foram definidas a `<cor>` e a `<forma>` dos pontos. Quando isto ocorre, o MATLAB utiliza a cor amarela (definida como padrão) e liga os pontos por meio de retas. Para se representar os mesmos pontos acima sem uni-los, isto é, isoladamente, com asteriscos e na cor azul escuro, substitui-se a linha de comando acima por:

```
» x=[-1,0,1,2];y=[1 2 -1 2];plot(x,y,'b*') <enter>
```

Finalmente, para representar aqueles pontos, unindo-os com linhas verdes tracejadas, deve-se digitar a seguinte linha de comando:

```
» x=[-1,0,1,2];y=[1 2 -1 2];plot(x,y,'g- -') <enter>
```

2.3.2 Representação de diversas curvas num mesmo gráfico

Você deve ter observado que, toda vez que foi digitado um novo comando `plot`, o gráfico anterior foi apagado, sendo substituído pelo gráfico cujas coordenadas dos pontos foram fornecidas no comando `plot` mais recente. A representação gráfica de diversas curvas em um mesmo gráfico se faz necessária em muitos problemas, principalmente quando se deseja analisar simultaneamente o comportamento de diversas grandezas ou, ainda, quando se deseja analisar a resposta de um sistema a diferentes entradas. Suponha, portanto, que se deseje utilizar um único gráfico para representar as curvas C_1, C_2, ..., C_n, cujas abscissas e ordenadas estão armazenadas nos vetores $\underline{x}_1, \underline{y}_1, \underline{x}_2, \underline{y}_2, \ldots$ e $\underline{x}_n, \underline{y}_n$. Isto pode ser feito de duas formas: *(i)* através de um comando `plot` com vários pares de vetores, isto é

```
» plot(x1,y1,'<cor1><forma1>',...,xn,yn,'<corn><forman>')
```

ou *(ii)* utilizando-se o comando `hold`. A forma geral do comando `hold` é a seguinte:

```
hold on
```

para fazer com que todos os comandos `plots` após o uso do comando `hold on` representem graficamente os pontos sobre o último gráfico criado antes do comando `hold on`. Para se desativar o comando `hold` basta digitar

```
» hold off <enter>
```

Por exemplo, para se representar as funções $y(t) = \text{Sen}\,(t)$ e $z(t) = \text{Cos}\,(t)$, $t \in [0, 2\pi]$ em um mesmo gráfico deve-se, inicialmente, definir um vetor `t` cujas

componentes são os instantes em que devem ser calculados $y(t)$ e $z(t)$ e, em seguida, calcular $y(t) = \text{Sen}\ (t)$ e $z(t) = \text{Cos}\ (t)$, conforme as linhas de comando a seguir:

```
» t=[0:2*pi/100:2*pi]; <enter>
» y=sin(t);z=cos(t); <enter>
```

Caso queira utilizar o comando `plot` com múltiplas entradas para representar $y(t)$ e $z(t)$ nos pontos definidos nas linhas de comando acima, o usuário deve digitar:

```
» plot(t,y,t,z) <enter>
```

A partir do gráfico obtido com o comando acima, vê-se que, embora o usuário não tenha definido nenhuma cor para as abscissas `y` e `z`, o MATLAB utilizou o azul para representar $y(t)$ e a magenta para representar $z(t)$. De fato, quando do uso do comando `plot` com múltiplas entradas, caso não sejam definidos `<cor>` e `<forma>` para cada curva, o MATLAB atribuirá cores diferentes a cada uma das curvas (na sequência das linhas da tabela 2.1(a)) e utilizará linhas contínuas (isto é, ligará os pontos por retas).

Uma maneira alternativa de representar as funções $y(t)$ e $z(t)$ num mesmo gráfico é usando o comando `hold`, conforme a linha de comando a seguir:

```
» plot(t,y);hold on;plot(t,z);hold off
```

A diferença principal entre as duas formas de se representar diversas curvas em um mesmo gráfico está no fato de que, ao se usar o comando `hold on`, e não havendo definição de `<cor>` e `<forma>`, então o MATLAB usará a cor azul para representar a curva e utilizará linha contínuas. Isto faz com que visualização de curvas representadas em um mesmo gráfico com o auxílio do comando `hold` não é tão boa quanto aquela fornecida pelo comando `plot` com múltiplas entradas, a menos que o usuário forneça detalhes referentes à `<cor>` e `<forma>` cada vez que usar o comando `plot`. □

2.3.3 Representação de diversos gráficos numa mesma figura

Vários gráficos podem ser representados numa mesma figura com a ajuda do comando `subplot`. Este comando tem a forma geral

```
subplot(m,n,k)
```

e gera uma matriz de gráficos de dimensão $m \times n$, e faz com que todos os comandos associados à representação gráfica tenham efeito somente sobre o gráfico da posição k. Esta posição é definida como 1 para o gráfico correspondente à posição $(1, 1)$ da matriz, 2 para o gráfico da posição $(1, 2)$, n para o gráfico da posição $(1, n)$,

$n + 1$ para o gráfico da posição $(2, 1)$ da matriz e assim por diante. Por exemplo, para se representar as duas função do exemplo 2.10 em dois gráficos diferentes, um situado no topo da figura ($y(t) = \text{Sen}\ (t)$) e o outro na parte inferior da figura ($z(t) = \text{Cos}\ (t)$) procede-se da seguinte forma:

```
» subplot(2,1,1);plot(t,y) <enter>
» subplot(2,1,2);plot(t,z) <enter>
```

2.3.4 Outros comandos para representação gráfica

Muitas vezes, a representação gráfica, por si só, não é auto-explicativa. Ela requer legendas para que se visualize, com maior clareza, o que significa cada gráfico (no caso de múltiplos gráficos) ou cada linha do gráfico (quando se trara de um gráfico com diversas curvas). Para tanto, utiliza-se os seguintes comandos:

(i) `title('<texto>')` : coloca um título (`<texto>`) na parte superior do gráfico.

(ii) `xlabel('<texto>')` : escreve o `<texto>` imediatamente abaixo do eixo das abscissas.

(iii) `ylabel('<texto>')` : escreve o `<texto>` à esquerda do do eixo das ordenadas.

(iv) `gtext('<texto>')` : escreve na tela o `<texto>` em posição a ser definida pelo usuário. Após teclar `<enter>`, o MATLAB apresenta a figura e um sinal `+`, para ser posicionado, com o auxílio do *mouse*, sobre o ponto do gráfico onde se deseja escrever o `<texto>`. Deve-se tomar o cuidado de maximizar a janela que contém a figura para que o ponto escolhido para se colocar o `<texto>` corresponda àquele em que ele, de fato, será posicionado.

(v) `grid` : quadricula a tela com linhas tracejadas.

Finalmente, deve ser ressaltado que o MATLAB permite representar gráficos em escala logarítmica (muito importante na análise da resposta em frequência de sistemas lineares estáveis). Isto é feito utilizando-se os seguintes comandos:

(vi) `semilogx(x,y,'<cor><forma>')`, `semilogy(x,y,'<cor><forma>')` e, ainda, `loglog(x,y,'<cor><forma>')` : permitem representar gráficos nos quais as abscissas, ordenadas e ambas, respectivamente, estão em escalas logarítmicas.

Exemplo 2.7 *Escreva um arquivo de funções para calcular e traçar os diagramas de módulo e fase de Bode de uma dada função de transferência.*

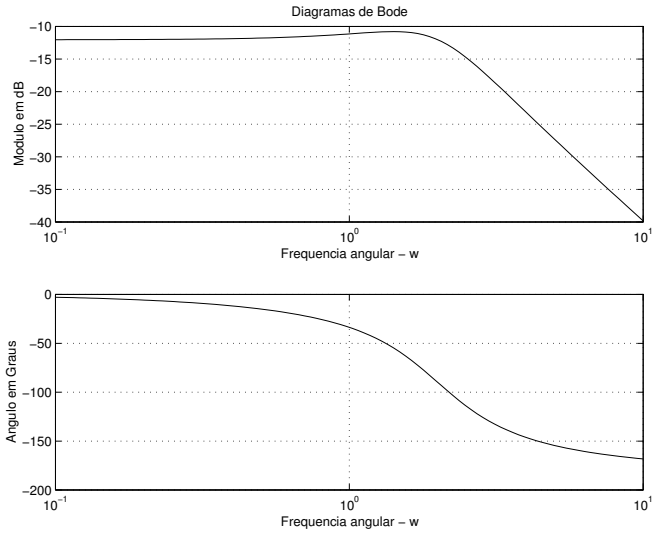


Figura 2.7: Diagramas de Bode para a função de transferência $G(s) = 1/(s^2+2s+4)$

Solução: *Denominemos* `diagbode` *a função que calcula os diagramas de Bode de módulo e de fase. Como entradas da função, temos os vetores formados pelos coeficientes dos polinômios do numerador e denominador da função de transferência,* `n` *e* `d`*, respectivamente, e o vetor* `w` *contendo as frequências onde se deseja calcular os diagramas de Bode. As saídas serão* `gjwdb` *e* `fasegjw`*, que são vetores cujas componentes são, respectivamente, o módulo, em decibéis, e a fase, em graus, da função de transferência nas frequências dadas em* `w`*. A figura 2.8 apresenta a listagem da função* `diagbode` *que traça os diagramas de módulo e fase de Bode de uma dada função de transferência. Note que os parâmetros de saída da função são os vetores cujos elementos são o módulo em dB e a fase em graus de* $G(j\omega)$ *para cada valor de frequência do vetor de entrada* `w`*.*

Para se obter os diagramas de Bode da função de transferência $G(s) = 1/(s^2 + 2s + 4)$*, procede-se da seguinte forma:*

```
» n=1;d=[1 2 4];w=logspace(-1,1,100);
» [gjwdb,fasegjw]=diagbode(n,d,w);
```

A função `diagbode` *retornará o módulo de* $G(j\omega)$ *em decibéis e a fase de* $G(j\omega)$*, em graus, nas frequências definidas no vetor* `w`*, armazenando-os, respectivamente, nos vetores* `gjwdb` *e* `fasegjw` *e, também, produzirá os diagramas de módulo e fase,*

```
% DIAGBODE  :  Calcula e traça os diagramas de Bode
%              de uma funcao de transferencia G(s)=b(s)/a(s)
% sendo  b(s) = b0.s^m + b1.s^(m-1) + ...  + bm
%        a(s) = a0.s^n + a1.s^(n-1) + ...  + an
% [gjwdb,fasegjw] = diagbode(b,a,w);
% sendo  b = [b0 b1 ...  bm]
%        a = [a0 a1 ...  an]
%        w = [w1 w2 ...  wp]
%
function [gjwdb,fasegjw] = diagbode(b,a,w);
jw=j*w;gjw=polyval(b,jw)./polyval(a,jw);
gjwdb=20*log10(abs(gjw));fasegjw=angle(gjw)*180/pi;
subplot(2,1,1);semilogx(w,gjwdb);grid;
title('Diagramas de Bode');xlabel('Frequencia angular - w');
ylabel('Modulo em dB');
subplot(2,1,2);semilogx(w,fasegjw);grid
xlabel('Frequencia angular - w');ylabel('Angulo em Graus');
```

Figura 2.8: Listagem da função `diagbode` para o cálculo dos diagramas de Bode de uma função de transferência.

representados na figura 2.7.

2.4 Controle de fluxo

Às vezes, torna-se necessário mudar a ordem de execução dos comandos em uma determinada função, ou ainda repetir uma sequência de comandos até que uma determinada condição seja verificada. Isto é feito, em MATLAB, usando-se as declarações `if` e os laços `for...end` e `while...end`, conforme será visto a seguir.

2.4.1 Declaração if

A forma geral para o uso da declaração `if` é a seguinte:

```
if <expressão booleana>
```

&	V	F
V	V	F
F	F	F

(a)

|	V	F
V	V	V
F	V	F

(b)

~	V
V	F
F	V

(b)

Tabela 2.2: Tabelas verdades para os operadores lógicos `&`(e), `|`(ou) e `~`(não)

```
    <sequência de comandos 1>
else
    <sequência de comandos 2>
end
```

A declaração `if` funciona da seguinte forma: quando a `<expressão booleana>` for verdadeira, as linhas de comandos definidas em `<sequência de comandos 1>` serão executadas e, caso contrário, serão executadas os comandos definidos na `<sequência de comandos 2>`. Uma forma mais simples da declaração `if` é:

```
if <expressão booleana>
    <sequência de comandos>
end
```

Neste caso, a `<sequência de comandos>` será executada somente quando a `<expressão booleana>` for verdadeira.

O resultado da `<expressão booleana>` (verdadeiro ou falso) é definido a partir de operadores lógicos `&`(e), `|`(ou) e `~`(não), cujas tabelas verdades são mostradas na tabela 2.2 e dos operadores relacionais: `<` (menor que), `>` (maior que), `=<` (menor ou igual), `>=` (maior ou igual), `==` (igual) e `~=` (diferente).

Usando a declaração `if then else`, estamos em condições melhorar a entrada de dados para o usuário na função `norma2` e também corrigir o erro da função `pderiv`, que não calcula corretamente a derivada de polinômios de grau zero.

Exemplo 2.8 *O resultado encontrado pela função* `norma2` *do exemplo 2.3 estará correto quando a entrada for um vetor coluna. Para que o usuário possa utilizá-lo também para calcular a norma euclideana de um vetor linha, o arquivo deve ter linhas que verifiquem se o vetor é do tipo linha ou coluna. Uma outra deficiência da função* `norma2` *é que ela é incapaz de distinguir se o usuário forneceu um vetor*

ou uma matriz. Utilizando-se declarações `if`*, todos esses problemas são sanados, conforme mostrado na listagem da figura 2.9*

```
% y=norma2(x) :  calcula a norma euclideana de um vetor
% sendo  x :  vetor de entrada
%        y :  norma euclideana de x
%
function y=norma2(x);
[m,n]=size(x);
if m>1 & n>1;
   disp(' ');disp('???? ERRO: a entrada deve ser um vetor');
   disp(' ');
   y=[];
else
   if m==1;y=sqrt(x*x');else;y=sqrt(x'*x);end
end
```

Figura 2.9: Listagem da função `norma2` modificada com a introdução de declarações `if`.

Agora verifique o programa para os vetores

$$\underline{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad e \quad \underline{w} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

e para a matriz

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}.$$

Note que, ao se fazer

```
» a=[1 2;3 4];na=norma2(a);<enter>
```

o MATLAB retornou a mensagem

```
???  ERRO: a entrada deve ser um vetor.
```

Isto se deve ao comando `disp`*, cuja forma geral é* `disp('<texto>')`*. Este comando faz com que a mensagem contida em* `<texto>` *seja impressa na janela do MATLAB. No arquivo acima usou-se, além do comando* `disp` *que contém a mensagem de erro a ser impressa na tela, outros dois contendo espaços em branco como texto. Eles*

foram usados para que o MATLAB deixe uma linha em branco entre a última linha de comando digitada pelo usuário e a mensagem impressa na tela e uma outra linha em branco entre essa mensagem e o próximo sinal de solicitação `»`.

Exemplo 2.9 *A correção da função* `pderiv` *é feita introduzindo-se uma declaração* `if` *conforme mostrado na figura 2.10.*

```
% PDERIV : calcula a derivada de um polinomio p(x)
% plinha = derivada(p)
%
% sendo:  p = [po p1 p2 ...  pn] os coeficientes do polinomio
%
function plinha = pderiv(p);
n=length(p)-1;
if n==0
   plinha=0;
else
   plinha=[n:-1:1].*p(1:n);
end
```

Figura 2.10: Função `pderiv` que calcula a derivada de um polinômio de grau n qualquer

2.4.2 Laço while

A estrutura de um laço `while` é a seguinte:

```
while <expressão booleana>
   <sequência de comandos>
end
```

Na estrutura acima, a `<sequência de comandos>` será executada enquanto a `<expressão booleana>` retornar um valor verdadeiro. Por exemplo, considere um arquivo de comandos (`comandos.m`), cujo conteúdo é o seguinte:

```
resp='sim';i=0;
while resp=='sim'
   i=i+1
```

```
    resp=input('Deseja continuar (sim/nao)?  ','s');
end
```

Crie um arquivo denominado `comandos.m` e digite os comandos acima. Em seguida execute o arquivo digitando

`» comandos <enter>`.

Observe que foram impressas na tela as seguintes mensagens:

```
i = 1
Deseja continuar (sim/nao)?
```

e o programa permanecerá parado até que o usuário digite `sim`, para continuar e `nao`, para parar definitivamente a execução do programa. Isto se deve ao comando `input`, que permite que o usuário atribua um valor a uma variável. A forma geral do comando input é

`<variável> = input('<texto>');`

quando a `<variável>` é numérica ou

`<variável> = input('<texto>','s');`

quando a `<variável>` é do tipo cadeia de caracteres. Observe, no exemplo acima que, como a expressão booleana é `resp=='sim'`, o programa, na verdade, interromperá sua execução sempre o usuário digitar qualquer expressão que não seja `sim`.

2.4.3 Laço for

A forma geral do laço `for` é a seguinte:

```
for <contador>=<valor inicial>:<incremento>:<valor final>
    <sequência de comandos>
end
```

onde

`<contador>` : variável de controle;

`<valor inicial>` : valor inicial da variável;

`<incremento>` : incremento dado à variável `<contador>`, podendo ser positivo ou negativo; quando o incremento for igual a 0, o laço `for` não será executado;

`<valor final>` : maior ou menor valor que a variável `<contador>` pode assumir.

O funcionamento do laço `for` pode ser explicado com a ajuda das estruturas `if` e `while`, conforme mostrado abaixo:

```
<contador> = <valor inicial>;
```

```
if <incremento> > 0
   while <contador> <= <valor final>
     <sequência de comandos>
     <contador> = <contador> + <incremento>
   end
else
   while <contador> >= <valor final>
     <sequência de comandos>
     <contador> = <contador> + <incremento>
   end
end
```

Exemplo 2.10 *Uma das formas de se representar matematicamente um sistema físico é através das chamadas equações de estados. Nesta representação, um sistema de ordem n será epresentado por um sistema de n equações diferenciais de primeira ordem, dado por:*

$$\begin{aligned} \dot{\underline{x}}(t) &= A\underline{x}(t) + \underline{b}u(t) \\ y(t) &= \underline{c}\,\underline{x}(t) + du(t) \end{aligned}$$

onde $\underline{x}(t) \in \mathbb{R}^{n\times 1}$ *é o vetor de estados,* $u(t) \in \mathbb{R}$ *e* $y(t) \in \mathbb{R}$ *denotam, respectivamente a entrada e a saída do sistema,* $A \in \mathbb{R}^{n\times n}$, $\underline{b} \in \mathbb{R}^{n\times 1}$ *e* $\underline{c} \in \mathbb{R}^{1\times n}$. *As equações de estados permitem, por exemplo, verificar se o sistema é controlável, isto é, se existe uma entrada* $u(t)$ *capaz de levar o sistema a qualquer estado* $\underline{x}_f$. *Para tanto é necessário verificar se a matriz de controlabilidade*

$$\mathcal{C} = \left[\begin{array}{cccc} \underline{b} & A\underline{b} & A^2\underline{b} & \ldots & A^{n-1}\underline{b} \end{array} \right]$$

tem posto igual a n. O arquivo de comandos (`controlabilidade.m` *cujo código está listado na figure 2.11 pode ser utilizado para verificar se o sistema é controlável ou não.*

Agora verifique se os seguintes sistemas são controláveis:

(a)

$$\dot{\underline{x}}(t) = \left[\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right] \underline{x}(t) + \left[\begin{array}{c} 1 \\ 0 \end{array} \right] u(t)$$

```
% CONTROLABILIDADE : verifica se um SLIT é controlável ou não
% sendo  :  a  :  n X n (matriz de transição de estados) e
           b  :  n X 1 (matriz de entrada)
[ma,na]=size(a);[mb,nb]=size(b);
% verificando se as dimensoes das matrizes
% a e b sao compativeis
if ma==na & mb==na
   n=ma;ci=b;c=ci;
% construindo a matriz controlabilidade
   for i=1:n-1
     ci=a*ci;c=[c ci];
   end
% verificando se o sistema é controlavel
   disp(' ');
   if rank(c)==n
     disp('O sistema e controlavel');disp(' ');
   else
     disp('O sistema nao e controlavel');disp(' ');
   end
else
   disp(' ');
   disp('As dimensoes das matrizes nao sao compativeis');
   disp(' ');
end
```

Figura 2.11: Código do arquivo de comandos `controlabilidade.m` para verificar a controlabilidade de um sistema.

(b)

$$\dot{\underline{x}}(t) = \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 2 \\ 0 \end{bmatrix} u(t)$$

2.4.4 A função find

Embora não tenha o poder de alterar a sequência de execução de um arquivo, a função `find`, pré-definida pelo MATLAB, tem como argumento uma expressão boolena. Sua forma geral é a seguinte:

`<variável> = find(<expressao booleana contendo um vetor>) .`

A funcão `find` retorna todos os índices das componentes do vetor para os quais a expressão booleana é verdadeira. Por exemplo, suponha que os vetores `t` e `y` contêm os instantes de simulação e os correspondentes valores da saída de um sistema de segunda ordem sub-amortecido para uma entrada igual ao degrau unitário. A função `find` pode ser usada para determinar os instantes de tempo imediatamente anterior e posterior àquele em que a saída atinge y_{ss} pela primeira vez e, consequentemente permite que se determine o tempo de subida da resposta, conforme mostra a seguinte sequência de comandos:

```
yss=y(length(y));ind=find(y>yss);
% instante de tempo imediatamente posterior a tr
trp=t(ind(1));yrp=y(ind(1));
% instante de tempo imediatamente anterior ou igual a ts
tra=t(ind(1)-1);yra=y(ind(1)-1);
% calculo do tempo de subida (por interpolação)
tr=tra+(trp-tra)*(yss-yra)/(yrp-yra)
```

Use os vetores `y` e `t` do exemplo 2.6 para verificar se os comandos acima estão, de fato, levando a uma correta determinação do tempo de subida da resposta.

2.5 Simulink

Para simular sistemas de controle, normalmente utiliza-se o Simulink, que é uma ferramenta do Matlab. Para abrir o Simulink no MATLAB, bastar clicar no icone da barra de ferramentas. Feito isso, é aberta a janela da biblioteca do Simulink representada na figura 2.12.

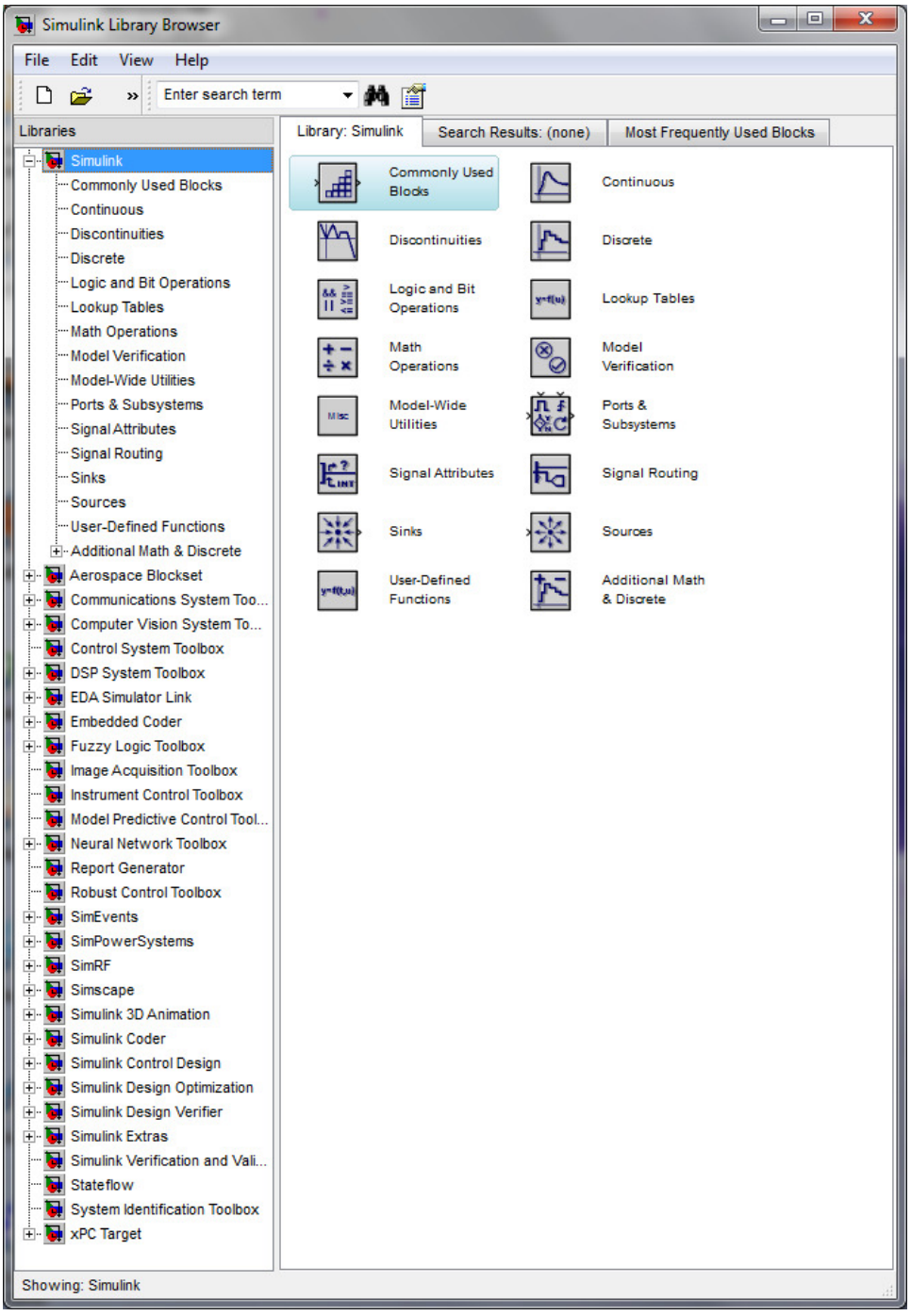


Figura 2.12: Biblioteca do Simulink

Para aprendermos a criar um arquivo para simulação no Simulink vamos utilizar o exemplo a seguir.

Exemplo 2.11 *Considere o esquema representado na figura 2.13 e seus parâmetros são:* $K_p = 0.6$, $T_i = 20$, $\tau = 0.1$ *e* $K = 1.36$. *Suponha que desejamos obter a resposta a um degrau unitário aplicado em* $t = 1$.

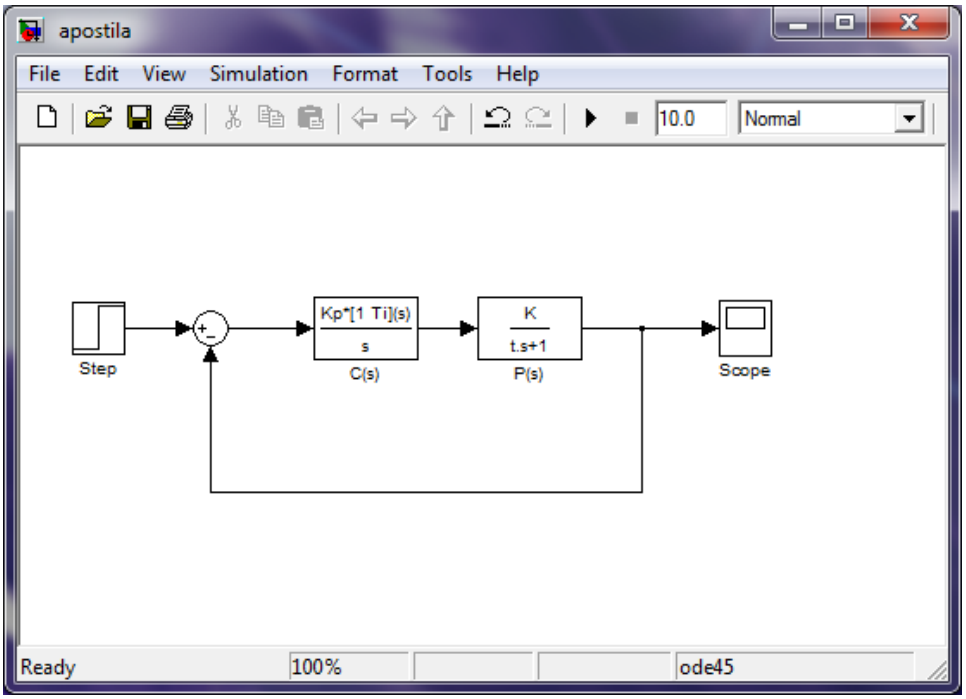
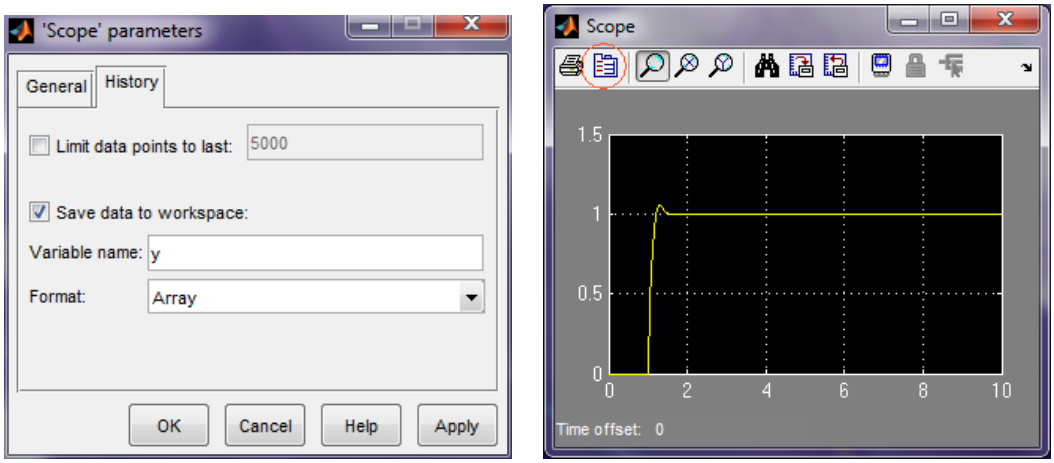


Figura 2.13: Esquema do diagrama em bloco do exemplo 2.11

Para criar um arquivo para simulação no Simulink, o primeiro passo é criar um arquivo utilizando-se opção `File -> New`. *Uma vez criado o arquivo, o próximo passo é adicionar os seguintes blocos:* `Step`, `Transfer function`, `Sum` *e* `Scope`. *Esses blocos são arrastados das bibliotecas* `Sources`, `Continuous`, `Math operations` *e* `Sinks`, *respectivamente. A configuração dos três primeiros blocos é feita de forma natural, seguindo a estrutura até aqui adotada pelo Matlab. Para realizar a configuração do bloco* `Scope` *deve-se dar um duplo click no bloco e, em seguida, selecionar o botão* `formato do arquivo` *(circundado em vermelho) como mostrado na figura 2.14(b). O próximo passo é selecionar a aba* `History` *e realizar a seguinte sequência do operações:* (i) *desmarque a opção* `Limit data points to`

`last`; (ii) *habilite a opção* `Save data to workspace`; (iii) *escolha o nome da variável e o format* `Array`. *Procedendo dessa forma, o resultado da simulação será salvo em uma matriz* $n \times 2$, `y=[y1 y2]`, *em que* n *é o número de pontos da simulação,* `y1=t` *(vetor com os instantes de tempo da simulação) e* `y2` *é a resposta do sistema.*



(a) (b)

Figura 2.14: Configuração do bloco scope (a) e resultado da simulação (b)

[htb]

Os parâmetros de simulação são configurados na janela mostrada na figura 2.15, que é aberta ao selecionar a opção `Simulation -> Configuration Parameters`. *Nessa opção, deve-se escolher como* `Solve options fixed-step`, `Solver: ode 4 (Runge-Kutta)` *e, na opção* `Fixed-step size`, *declarar a variável* `h` *e atribuir* 10^{-3} *a* `h` *no workspace. Finalmente, na opção* `Tasking mode`, *escolher* `SingleTasking`. *Para realizar a simulação, basta clicar no botão* `Play` *da janela que contém o diagrama de blocos para simulação. O resultado da simulação com os parâmetros acima está mostrado na figura 2.14(a)*

2.6 Comentários finais

Conforme foi mencionado na introdução deste capítulo, o MATLAB possui muito mais recursos do que foi apresentado aqui. O objetivo principal desse capítulo foi introduzir os comandos básicos para que o usuário se sinta confortável ao utilizar

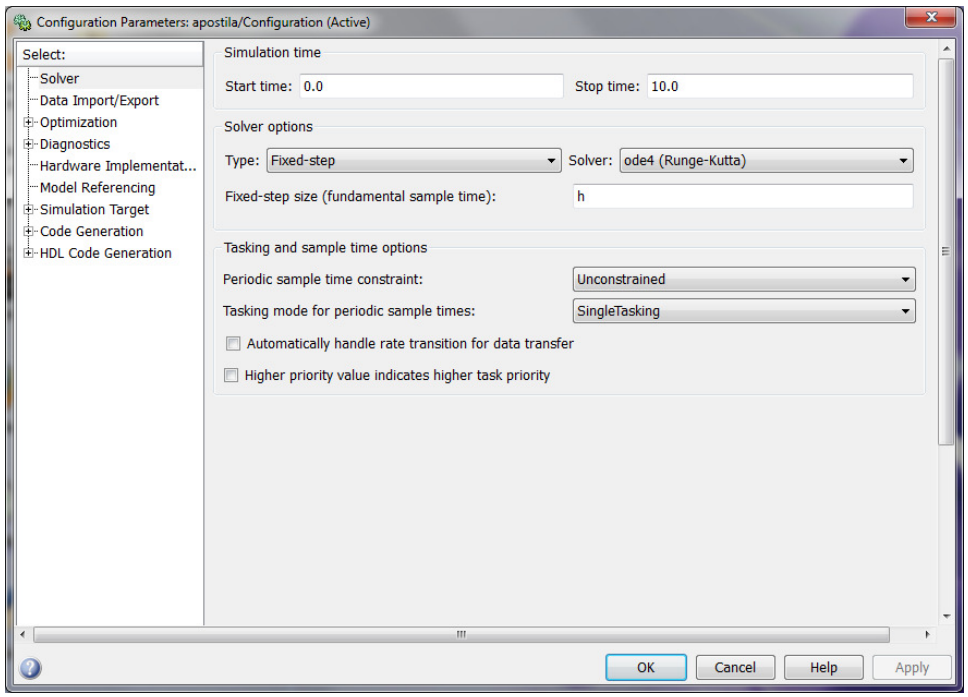


Figura 2.15: Configuração dos parâmetros de simulação

o MATLAB e tenha a ferramenta mínima necessária para que possa vir a usar com sucesso linguagem. Para se ter acesso a comandos/recursos mais poderos, recomenda-se a consulta aos manuais do MATLAB e do SIMULINK.

Exercícios propostos

1. Entre com a matriz

$$Z = \begin{bmatrix} 1-j & 1+2j & 2-j \\ 2 & 1+2j & 3j \end{bmatrix}$$

   e execute os comandos *(i)* a *(vi)* da seção 2.2.2.

2. Para o vetor

$$\underline{x} = \begin{bmatrix} 1 & 10 & 7 & 0.1 \end{bmatrix}$$

execute os comandos *(vii)* a *(ix)* da seção 2.2.2.

3. Faça um programa MATLAB (arquivo de função) para traçar o diagrama de Nyquist de uma determinada função de transferência

$$G(s) = \frac{n(s)}{d(s)}$$

sendo dados os coeficientes de $n(s)$ e $d(s)$ e um vetor $\underline{\omega}$ contendo as frequências de interesse. Por exemplo, a função poder-se-ia chamar de `meunyq` sendo descrita por `gjw = meunyq(n,d,w)`.

4. Seja $s = \sigma + j\omega$ uma variável complexa e seja

$$F(s) = \frac{s}{s+2} \, .$$

Escreva um arquivo de comandos em MATLAB para calcular e representar graficamente o mapeamento de $F(s)$ sobre Γ, onde Γ é o contorno representado na figura 2.16.

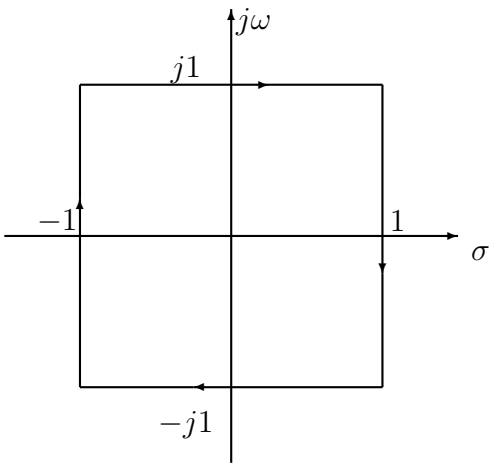


Figura 2.16: Contorno Γ para o exercício 2.4

5. Escrever um arquivo de função que permita somar dois polinômios $p(s) = p_0 s^m + p_1 s^{m-1} + \cdots + p_{m-1} s + p_m$ e $q(s) = q_0 s^n + q_1 s^{n-1} + \cdots + q_{n-1} s + q_n$. Teste a função para os polinômios $p(s) = s + 1$ e $q(s) = s^2 + 2s - 1$

6. Suponha que sejam dadas as funções de transferências da planta e do controlador

$$G(s) = \frac{n_G(s)}{d_G(s)} \text{ e } K(s) = \frac{n_K(s)}{d_K(s)}.$$

Escreva arquivos de funções que permitem calcular as seguintes funções de transferências para o sistema cujo diagrama de blocos está representado na figura 2.17: *(a)* $T_{RY}(s) = Y(s)/R(s)$, *(b)* $T_{RE}(s) = E(s)/R(s)$ e *(c)* $T_{RU}(s) = U(s)/R(s)$

Nota: As entradas para as funções serão os vetores formados pelos coeficientes dos polinômios $n_G(s)$, $d_G(s)$, $n_K(s)$ e $d_K(s)$ e as saídas serão vetores formados pelos coeficientes dos polinômios do numerador e do denominador da função de transferência considerada. Por exemplo, a função MATLAB que calcula o numerador e o denominador da função de transferência $T_{RY}(s)$, poderia ser definida como `[ntry,dtry] = try(ng,dg,nk,dk)`.

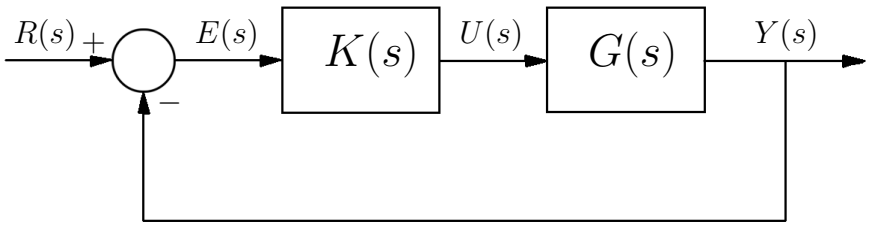


Figura 2.17: Diagrama de blocos para o exercício 2.6

7. Suponha que sejam conhecidas as coordenadas de dois pontos $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ e a abscissa de um terceiro ponto $P_3 = (x_3, y_3)$, onde $x_1 < x_3 < x_2$. Escreva um arquivo do tipo função que permita encontrar y_3 interpolando-se os pontos P_1 e P_2 por uma reta, isto é, a ordenada de P_3 será $y_3 = ax_3 + b$, onde a e b são, respectivamente, os coeficientes angular e linear da reta que passa pelos pontos P_1 e P_2.

8. Suponha que sejam conhecidas as coordenadas de dois pontos $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ e a ordenada de um terceiro ponto $P_3 = (x_3, y_3)$, onde $y_1 < y_3 <$

y_2. Escreva um arquivo do tipo função que permita encontrar x_3 interpolando-se os pontos P_1 e P_2 por uma reta.

9. Escreva um arquivo do tipo função que permita calcular: *(i)* os tempos de subida (t_r) e de acomodação (t_s) para um sistema superamortecido ou criticamente amortecido e *(ii)* os tempos de subida (t_r), de pico (t_p) e de acomodação (t_s) e o percentual de ultrapassagem (PO) para um sistema subamortecido.
   Nota: Assuma que sejam conhecidos t e $y(t)$, estando armazenados nos vetores $\underline{t} = \left[\begin{array}{cccc} t_1 & t_2 & \ldots & t_n \end{array}\right]$ e $\underline{y} = \left[\begin{array}{cccc} y(t_1) & y(t_2) & \ldots & y(t_n) \end{array}\right]$, respectivamente.
   Atenção: Use o comando `find` e interpolação.

10. Suponha que

$$Q(s) = G(s)K(s) = K\frac{n_Q(s)}{d_Q(s)} \, ,$$

    onde $n_Q(s)$ e $d_Q(s)$ são conhecidos e K deve ser arbitrado. Escreva um arquivo de funções que:

    (a) Verifique se o sistema realimentado (realimentação negativa) é estável para um dado valor de K.

    (b) Caso o sistema seja estável, encontre as margens de fase e de ganho do sistema.
    Nota: Use o comando `find` e faça interpolação para achar os valores corretos.

11. Dado um conjunto de pontos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, escreva um arquivo de funções em MATLAB que utilize o método dos mínimos quadrados para calcular o coeficiente angular da reta $y = \alpha x$, conforme a figura 2.18.
    Nota: Veja capítulo 3, equação **??**

12. Suponha que a partir de um experimento, obtém-se os vetores

$$\begin{array}{rcl} \underline{\omega} & = & \left[\begin{array}{cccc} \omega_0 & \omega_1 & \ldots & \omega_q \end{array}\right] \\ \underline{\phi} & = & \left[\begin{array}{cccc} \phi_0 & \phi_1 & \ldots & \phi_q \end{array}\right] \\ \underline{h} & = & \left[\begin{array}{cccc} \mid H(j\omega_0) \mid & \mid H(j\omega_1) \mid & \ldots & \mid H(j\omega_q) \mid \end{array}\right] \end{array}$$

    onde ω_i, $i = 0, 1, \ldots, q$ são as frequências angulares e $H(j\omega_i) = \mid H(j\omega_i) \mid \exp(j\phi_i)$. Escreva um arquivo de funções que utilize o método dos mínimos

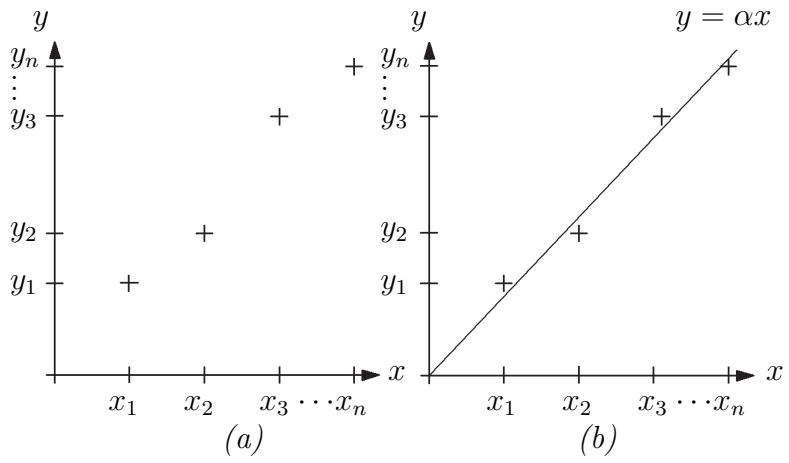


Figura 2.18: Representação dos pontos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ e da reta ajustada $y = \alpha x$ referentes ao exercício 2.11.

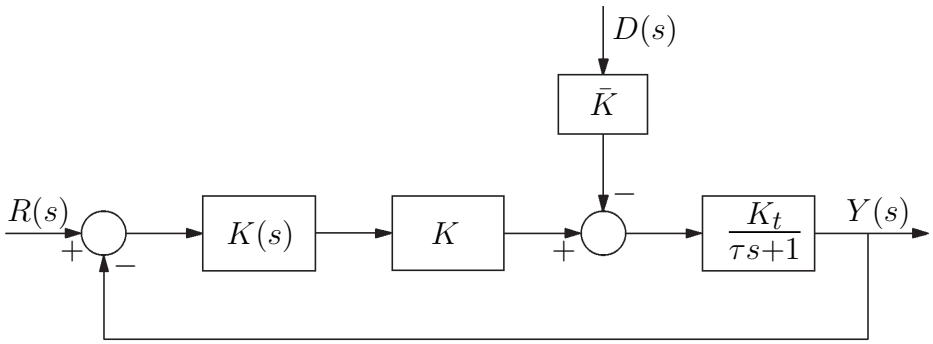


Figura 2.19: Diagrama de blocos para o sistema do exercício 2.13

quadrados (veja capítulo 3, algoritmo 5) para calcular os coeficientes da função de transferência

$$H(s) = \frac{b(s)}{a(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1} s + b_m}{s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n} \ ,$$

onde os graus dos polinômios do numerador e do denominador, m e n, respectivamente, são arbitrados.

13. Construa um arquivo SIMULINK para fazer a simulação do sistema da figura 2.19, onde $K = 127.9$, $K_t = 0.0056$, $\tau = 0.026$ e $\bar{K} = 451.8$, para as seguintes situações:

    (a) $r(t) = 12u_0(t)$ e $d(t) = 0$ e $K(s) = 5.4/s$, onde $u_0(t)$ representa um degrau unitário aplicado no instante $t = 0$.

    (b) $r(t) = 12u_0(t)$ e $d(t) = 0$ e $K(s) = (s + 30)/s$.

    (c) Reptita os itens *(a)* e *(b)* para $d(t) = 0.2u_0(t - 1)$

    Nota: Em todos os casos acima, o intervalo de simulação deve ser de -0.5 a $2s$.

Capítulo 3

Modelagem e identificação dos parâmetros da função de transferência motor CC

Neste capítulo iremos obter um modelo matemático para o motor CC controlado pela armadura. O modelo adotado será o de um sistema linear descrito por uma função de transferência cuja entrada é a tensão de armadura e cuja saída é a velocidade angular. Supor que o sistema como linear representa uma restrição muito forte pois exige que este tenha um comportamento linear em todas as faixas de operação. Na prática, contudo, o que ocorre é que os sistemas têm, em geral, comportamento aproximadamente linear em faixas de valores da variável de entrada. Assim sendo, torna-se necessário determinar experimentalmente uma faixa de operação na qual o comportamento do motor possa ser considerado linear. Uma importante ferramenta matemática utilizada na determinação da região linear é o chamado método dos mínimos quadrados, uma vez que permite ajustar um conjunto de pontos por uma função polinomial $p(x)$ cujo grau deve ser arbitrado pelo projetista. Após a determinação da região linear será possível proceder à identificação dos parâmetros da função de transferência. Para tanto, iremos realizar experimentos baseados tanto na resposta ao degrau como na resposta em frequência.

O restante desse capítulo está estruturado da seguinte forma. Na seção 3.1 vamos apresentar uma breve revisão dos principais fundamentos teóricos que serão utilizados no capítulo. Em seguida, na seção 3.2 iremos desenvolver o modelo

matemático do motor CC controlado pela armadura. Finalmente, na seção 3.4 iremos descrever os experimentos necessários para a obtenção dos dados a serem utilizados na determinação dos parâmetros do modelo.

3.1 Fundamentos teóricos

3.1.1 O método dos mínimos quadrados

Considere o problema de se resolver um sistema linear

$$A\underline{x} = \underline{b}, \tag{3.1}$$

em que a matriz A tem dimensão $m \times n$ ($m \gg n$). Esse sistema, em geral, não tem solução, isto é, $A\underline{x} \neq \underline{b}$, uma vez que o número de linhas de A é muito maior do que o de colunas. Assim, em raras ocasiões o vetor $\underline{b}$ irá pertencer ao espaço gerado pelas colunas de A.

Quando o sistema (3.1) não tem solução, é comum em engenharia, buscar um vetor $\underline{x}$ que, embora não seja solução do sistema, seja tal que minimize a norma quadrática do erro entre $A\underline{x}$ e $\underline{b}$. Seja, portanto,

$$\hat{\underline{b}} = A\underline{x}, \tag{3.2}$$

e defina

$$\underline{e} = \hat{\underline{b}} - \underline{b}. \tag{3.3}$$

Considere o seguinte problema: encontre $\underline{x}$ tal que $\|\underline{e}\|_2^2$ seja mínima. Note que, se existir $\underline{x}$ tal que $A\underline{x} = \underline{b}$ então $\|\underline{e}\|_2^2 = 0$, o que implica que mesmo no caso em que o sistema de equações (3.1) tem solução, o método dos mínimos quadrados também levará a essa solução.

A partir das equações (3.2) e (3.3) tem-se:

$$\begin{aligned} \|\underline{e}\|_2^2 &= < A\underline{x} - \underline{b}, A\underline{x} - \underline{b} > = (A\underline{x} - \underline{b})^t (A\underline{x} - \underline{b}) \\ &= \underline{x}^t A^t A\underline{x} - \underline{x}^t A^t \underline{b} - \underline{b}^t A\underline{x} + \underline{b}^t \underline{b}. \end{aligned} \tag{3.4}$$

Como $\underline{x}^t A^t \underline{b} \in \mathbb{R}$ então $\underline{x}^t A^t \underline{b} = (\underline{x}^t A^t \underline{b})^t = \underline{b}^t A\underline{x}$, o que nos permite escrever:

$$\|\underline{e}\|_2^2 = \underline{x}^t A^t A\underline{x} - 2\underline{b}^t A\underline{x} + \underline{b}^t \underline{b}. \tag{3.5}$$

Portanto, o seguinte problema de otimização deve ser resolvido:

$$\min_{\underline{x}\in\mathbb{R}^n} \|\underline{e}\|_2^2, \tag{3.6}$$

sendo $\|\underline{e}\|_2^2$ dada pela equação (3.5).

Contudo, antes de passarmos à solução do problema de otimização acima, alguns conceitos devem ser relembrados:

1. Seja $y = f(x_1, x_2, \ldots, x_n)$ uma função de $\mathbb{R}^n$ em $\mathbb{R}$. A derivada de y em relação ao vetor $\underline{x}$ é definida como:

$$\frac{\partial y}{\partial \underline{x}} = \begin{bmatrix} \dfrac{\partial y}{\partial x_1} \\ \dfrac{\partial y}{\partial x_2} \\ \vdots \\ \dfrac{\partial y}{\partial x_n} \end{bmatrix}. \tag{3.7}$$

2. De acordo com a notação introduzida na equação (3.7), tem-se que:

(a) Se $y = \underline{b}^t\underline{x}$, em que $\underline{b} \in \mathbb{R}^n$, então

$$\frac{\partial y}{\partial \underline{x}} = \underline{b}. \tag{3.8}$$

Prova: Seja $\underline{b}^t = \begin{bmatrix} b_1 & b_2 & \ldots & b_n \end{bmatrix}$. Então $y = \underline{b}^t\underline{x} = b_1x_1 + b_2x_2 + \ldots + b_nx_n$, e portanto:

$$\frac{\partial y}{\partial \underline{x}} = \begin{bmatrix} \dfrac{\partial y}{\partial x_1} \\ \dfrac{\partial y}{\partial x_2} \\ \vdots \\ \dfrac{\partial y}{\partial x_n} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \underline{b}. \tag{3.9}$$

(b) Se $y = \underline{x}^tH\underline{x}$, onde H é uma matriz simétrica, isto é $H^t = H$, então

$$\frac{\partial y}{\partial \underline{x}} = 2H\underline{x}. \tag{3.10}$$

Prova: Por indução finita sobre n.

(i) $n = 1$. Neste caso, $H = h_{11}$, $\underline{x} = x_1$ e, dessa forma

$$y = x_1 h_{11} x_1 = h_{11} x_1^2. \tag{3.11}$$

Portanto:

$$\frac{\partial y}{\partial \underline{x}} = \frac{dy}{dx_1} = 2h_{11}x_1 = 2H\underline{x}. \tag{3.12}$$

(ii) Suponha que a expressão (3.10) seja válida quando $H \in \mathbb{R}^{n\times n}$, isto é, se $y = \underline{x}^t H \underline{x}$ então $\partial y/\partial \underline{x} = 2H\underline{x}$.

(iii) Seja agora $H \in \mathbb{R}^{(n+1)\times(n+1)}$, isto é:

$$H = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} & h_{1,n+1} \\ h_{12} & h_{22} & \cdots & h_{2n} & h_{2,n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ h_{1n} & h_{2,n} & \cdots & h_{nn} & h_{n,n+1} \\ h_{1,n+1} & h_{2,n+1} & \cdots & h_{n,n+1} & h_{n+1,n+1} \end{bmatrix}. \tag{3.13}$$

Definindo

$$\begin{aligned} H_n &= \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{12} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & & \vdots \\ h_{1n} & h_{2,n} & \cdots & h_{nn} \end{bmatrix}, \\ \underline{h}_{n+1}^t &= \begin{bmatrix} h_{1,n+1} & h_{2,n+1} & \cdots & h_{n,n+1} \end{bmatrix}, \\ \underline{x}_n &= \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}, \end{aligned} \tag{3.14}$$

então $y = \underline{x}^t H \underline{x}$ é equivalente a:

$$\begin{aligned} y &= \begin{bmatrix} \underline{x}_n^t & x_{n+1} \end{bmatrix} \begin{bmatrix} H_n & \underline{h}_{n+1} \\ \underline{h}_{n+1}^t & h_{n+1,n+1} \end{bmatrix} \begin{bmatrix} \underline{x}_n \\ x_{n+1} \end{bmatrix} \\ &= \begin{bmatrix} \underline{x}_n^t H_n + x_{n+1}\underline{h}_{n+1}^t & \underline{x}_n^t \underline{h}_{n+1} + x_{n+1} h_{n+1,n+1} \end{bmatrix} \begin{bmatrix} \underline{x}_n \\ x_{n+1} \end{bmatrix} \\ &= \underline{x}_n^t H_n \underline{x}_n + x_{n+1}\underline{h}_{n+1}^t \underline{x}_n + \underline{x}_n^t \underline{h}_{n+1} x_{n+1} + x_{n+1}^2 h_{n+1,n+1} \\ &= \underline{x}_n^t H_n \underline{x}_n + 2x_{n+1}\underline{h}_{n+1}^t \underline{x}_n + x_{n+1}^2 h_{n+1,n+1}. \end{aligned} \tag{3.15}$$

Note que para a obtenção da última expressão usamos o fato de que como $\underline{x}_n^t \underline{h}_{n+1} \in \mathbb{R}$ então $\underline{x}_n^t \underline{h}_{n+1} = (\underline{x}_n^t \underline{h}_{n+1})^t = \underline{h}_{n+1}^t \underline{x}_n$.

Desta forma, $\partial y/\partial \underline{x}$ será:

$$\begin{aligned}\frac{\partial y}{\partial \underline{x}} &= \begin{bmatrix} \frac{\partial y}{\partial \underline{x}_n} \\ \frac{\partial y}{\partial x_{n+1}} \end{bmatrix} = \begin{bmatrix} 2H_n\underline{x}_n + 2x_{n+1}\underline{h}_{n+1} \\ 2\underline{h}_{n+1}^t\underline{x}_n + 2h_{n+1,n+1}x_{n+1} \end{bmatrix} \\ &= 2\begin{bmatrix} H_n & \underline{h}_{n+1} \\ \underline{h}_{n+1}^t & h_{n+1,n+1} \end{bmatrix}\begin{bmatrix} \underline{x}_n \\ x_{n+1} \end{bmatrix} = 2H\underline{x}. \end{aligned} \tag{3.16}$$

□

Vamos retornar ao problema de otimização (3.6). Usando os resultados (3.8) e (3.10), podemos escrever:

$$\frac{\partial}{\partial \underline{x}}\|\underline{e}\|_2^2 = 2(A^tA)\underline{x} - 2A^t\underline{b}, \tag{3.17}$$

e, portanto, o valor de $\underline{x}$ que minimiza $\|\underline{e}\|_2^2$ será tal que

$$2(A^tA)\underline{x} - 2A^t\underline{b} = \underline{0}. \tag{3.18}$$

Observe que, como, em geral, $m \gg n$, então o posto de A é, geralmente, igual a n e, portanto, A^tA é não singular, sendo desta forma inversível. Consequentemente, o vetor $\underline{x}$ que minimiza $\|\underline{e}\|_2^2$ será:

$$\underline{x} = (A^tA)^{-1}A^t\underline{b}. \tag{3.19}$$

3.1.2 Ajuste de um conjunto de p-pares cartesianos por um polinômio utilizando o método dos mínimos quadrados

Vamos considerar agora o seguinte problema: ajuste um conjunto de p-pares cartesianos (x_1, y_1), (x_2, y_2), ..., (x_p, y_p), por um polinômio $y(x)$ de grau n, isto é, encontre os coeficientes $a_0, a_1, \ldots, a_q$ de

$$y(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n, \tag{3.20}$$

em que $p >> n$, utilizando o método dos mínimos quadrados.

Substituindo os pontos (x_k, y_k), $k = 1, 2, \ldots, p$, na equação (3.20), obtém-se:

$$\begin{aligned} y_1 &= a_0x_1^n + a_1x_1^{n-1} + \ldots + a_{n-1}x_1 + a_n \\ y_2 &= a_0x_2^n + a_1x_2^{n-1} + \ldots + a_{n-1}x_2 + a_n \\ &\vdots \\ y_p &= a_0x_p^n + a_1x_p^{n-1} + \ldots + a_{n-1}x_p + a_n \end{aligned} \tag{3.21}$$

É fácil verificar que a equação (3.21) pode ser escrita na seguinte forma matricial:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \begin{bmatrix} x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ x_2^n & x_2^{n-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_p^n & x_p^{n-1} & \cdots & x_p & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} \tag{3.22}$$

Definindo

$$\underline{b} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}, \; A = \begin{bmatrix} x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ x_2^n & x_2^{n-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_p^n & x_p^{n-1} & \cdots & x_p & 1 \end{bmatrix} \text{ e } \underline{x} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix}, \tag{3.23}$$

então a equação (3.22) pode ser escrita como:

$$A\underline{x} = \underline{b}. \tag{3.24}$$

Como $p >> n$, a equação (3.24) não tem, em geral, solução. Utilizando o método do mínimos quadrados para encontrar a solução que minimiza a norma quadrática do erro $e = A\underline{x} - \underline{b}$, obtém-se, utilizando a equação (3.18), a seguinte solução

$$\underline{x} = (A^T A)^{-1} A^T \underline{b} \tag{3.25}$$

Note que se o ponto $(0, 0)$ for um dos pontos a serem ajustados, então o gráfico do polinômio resultante deve passar pela origem. Dessa forma $a_n = 0$, isto é, somente os n primeiros parâmetros devem ser encontrados. Nesse caso, é fácil verificar que a matriz A toma a seguinte forma:

$$A = \begin{bmatrix} x_1^n & x_1^{n-1} & \cdots & x_1 \\ x_2^n & x_2^{n-1} & \cdots & x_2 \\ \vdots & \vdots & & \vdots \\ x_p^n & x_p^{n-1} & \cdots & x_p \end{bmatrix} \tag{3.26}$$

Em resumo, o ajuste de um conjunto de p pontos $(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)$, por um polinômio de grau n, $n << p$, pode ser feito da seguinte forma:

Algoritmo 3.1

Entrada *Pontos a serem ajustados,* $(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)$*, e o grau desejado do polinômio,* n*.*

PASSO 1 *Forme o vetor* $\underline{b}$ *de acordo com a equação (3.23).*

PASSO 2 *Verifique se o ponto* $(0, 0)$ *é um dos pontos a serem ajustados. Em caso afirmativo, proceda da seguinte forma:*

- *Forme a matriz* A *de acordo com a equação (3.26).*
- *Calcule o vetor* $\underline{x}$ *de acordo com a equação (3.25)*
- *Faça*

$$\underline{x}^* = \begin{bmatrix} \underline{x} \\ 0 \end{bmatrix}.$$

- *Fim*

Em caso negativo, proceda da seguinte forma:

- *Forme a matriz* A *de acordo com a equação (3.23).*
- *Calcule o vetor* $\underline{x}$ *de acordo com a equação (3.25)*
- *Faça* $\underline{x}^* = \underline{x}$*.*
- *Fim*

Saída *Vetor* $\underline{x}^* = \begin{bmatrix} a_0 & a_1 & \cdots & a_n \end{bmatrix}$*, cujas componentes são os coeficientes do polinômio.*

3.1.3 Expansão em série de Fourier de funções periódicas

Seja $f(t)$ uma função definida no intervalo $[t_0, t_0 + T]$, em que $T = 2\pi/\omega_0$. Então $f(t)$ tem a seguinte expansão em série de Fourier:

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t), \tag{3.27}$$

sendo

$$a_0 = \frac{1}{T}\int_{t_0}^{t_0+T} f(t)dt \tag{3.28}$$

$$a_n = \frac{2}{T}\int_{t_0}^{t_0+T} f(t)\cos(n\omega_0 t)dt \tag{3.29}$$

$$b_n = \frac{2}{T}\int_{t_0}^{t_0+T} f(t)\sin(n\omega_0 t)dt \tag{3.30}$$

Às vezes, é mais conveniente escrever a expansão em série de Fourier como:

$$f(t) = a_0 + \sum_{n=1}^{\infty} c_n \sin(n\omega_0 t + \phi_n), \tag{3.31}$$

em que os coeficientes c_n e ϕ_n são, respectivamente, o módulo e a fase (em radianos) do número complexo $z_n = b_n + ja_n$, isto é, $z_n = c_n e^{j\phi_n}$.

Na prática, a expansão em série de Fourier é usada para encontrar as harmônicas de um sinal periódico — as mais importantes são a primeira harmônica, utilizada na obtenção da resposta em frequência de um sistema, e a terceira harmônica, muito utilizada em sistemas de energia elétrica. Nesses casos, o sinal $f(t)$ não é descrito analiticamente mas sim por um conjunto de pares ordenados $(t_k, f(t_k))$, sendo $t_k = t_0 + kh$, $k = 0, 1, \ldots, p$, em que h denota o intervalo de amostragem e $T = ph$. Nessas condições, os valores de a_0, c_n e ϕ_n serão calculados numericamente de acordo com o seguinte algoritmo.

Algoritmo 3.2

Entrada *N : número de harmônicos a serem determinados*

Passo 1 *Obtenha um conjunto de $p+1$ pontos $(t_k, f(t_k))$, em que $t_k = t_0 + kh$, $k = 0, 1, \ldots, p$, que descrevem a função no intervalo $[t_0, t_0 + T]$ e armazene os pontos t_k e $f(t_k)$ nos vetores* `t` *e* ft*, respectivamente.*

Passo 2 *Utilizando a função* `trapz` *do Matlab, calcule numericamente a_0 definido de acordo com a equação (3.28) da seguinte forma:*
```
a0 = trapz(t,ft)/T
```

Passo 3 *Para n variando de 1 a N, calcular:*

- `an = (2/T)*trapz(t,(cos(2*π*n)*t/T).*ft)`
- `bn = (2/T)*trapz(t,(sin(2*π*n)*t/T).*ft)`
- `zn = bn + j*an` ⇒ `cn = abs(zn)`, ϕ`n = angle(zn)`

Saída *Vetores* $\underline{c} = \left[\begin{array}{cccccc} a_0 & c_1 & c_2 & \cdots & c_N \end{array}\right]$ *e* $\underline{\phi} = \left[\begin{array}{cccccc} \phi_1 & \phi_2 & \cdots & \phi_N \end{array}\right]$

3.1.4 Obtenção da resposta em frequência de um sistema linear invariante no tempo estável

Consideremos um sistema linear invariante no tempo estável com entrada e saída denotadas por $u(t)$ e $y(t)$, respectivamente, e cuja função de transferência é denotada por $G(s)$. Tem-se, portanto, que:

$$G(s) = \frac{Y(s)}{U(s)}, \tag{3.32}$$

em que $U(s)$ e $Y(s)$ são as transformadas de Laplace de $u(t)$ e $y(t)$, respectivamente. Não é difícil demonstrar que, como $G(s)$ é estável, então a resposta de regime permanente a uma entrada senoidal $u(t) = U_{m_k} Sen(\omega_k t + \theta_k)$ é igual a $y_{ss}(t) = Y_{m_k} Sen(\omega_k t + \phi_k)$ em que

$$Y_{m_k} = |G(j\omega_k)| U_{m_k} \text{ e } \phi_k = \theta_k + \angle G(j\omega_k). \tag{3.33}$$

Assim, o valor de $G(j\omega_k) = |G(j\omega_k)| e^{j\angle G(j\omega_k)}$ pode ser determinado diretamente a partir da equação (3.33) sendo:

$$|G(j\omega_k)| = \frac{Y_{m_k}}{U_{m_k}} \text{ e } \angle G(j\omega_k) = \phi_k - \theta_k. \tag{3.34}$$

Quando o experimento é repetido para diversas frequências ω_k, $k = 1, 2, \ldots, p$, cobrindo um espectro de frequências que permite verificar o comportamento do sistema, os valores de $G(j\omega_k)$ determinam a chamada resposta em frequência de $G(s)$. Aos gráficos $\omega \times |G(j\omega_k)|$ e $\omega \times \angle G(j\omega_k)$ dá-se o nome de curvas de módulo e de fase da resposta em frequência de $G(s)$.

Na prática, contudo, o sinal de saída é geralmente corrompido por ruídos introduzidos pelos sensores de medição, o que faz com que o sinal $y_{ss}(t)$ não tenha uma forma senoidal exata. Para contornar esse problema, faz-se a expansão em

série de Fourier desse sinal e toma-se o valores do módulo e da fase da frequência fundamental como os valores de Y_{m_k} e ϕ_k como a amplitude e a fase do valor de regime permanente da resposta. Assim, a obtenção da resposta em frequência de um sistema linear invariante no tempo estável pode ser feita de acordo com o seguinte algoritmo.

Algoritmo 3.3

Passo 1 *Selecione um número finito p de frequências angulares ω_k, $k = 1, 2, \ldots, p$. A frequência mais baixa é ditada pelo equipamento destinado a gerar o sinal senoidal de entrada enquanto a frequência mais alta é determinada pela razão entre as amplitudes do sinal de saída e do ruído de medição.*

Passo 2 *Aplique entradas senoidais da forma $u_k(t) = U_{m_k} Sen(\omega_k t + \theta_k)$, $k = 1, 2, \ldots, p$, e faça a aquisição dos pontos referentes aos sinais de entrada e de saída, $(t_i, u_k(t_i))$ e $(t_i, y_k(t_i))$, respectivamente, para cada um dos sinais de entrada aplicados e correspondentes saídas.*

Passo 3 *Utilize o algritmo 3.2 para obter os termos fundamentais das séries de Fourier dos sinais de entrada e saída, dados respectivamente por:*

$$u_{k_f}(t) = C_{u_k} Sen(\omega_k t + \phi_{u_k}) \tag{3.35}$$

$$y_{k_f}(t) = C_{y_k} Sen(\omega_k t + \phi_{y_k}) \tag{3.36}$$

Passo 4 *Calcule $G(j\omega_k) = |G(j\omega_k)| e^{\phi_k}$, sendo*

$$|G(j\omega_k)| = \frac{C_{y_k}}{C_{u_k}} \tag{3.37}$$

$$\phi_k = \phi_{y_k} - \phi_{u_k} \tag{3.38}$$

3.1.5 Identificação dos parâmetros K e τ da função de transferência de primeira ordem de um SLIT utilizando resposta ao degrau

Considere um SLIT com resposta ao degrau monotonicamente crescente com entrada $u(t)$ e saída $y(t)$. Uma das formas de se modelar esse sistema é por meio de uma função de transferência de primeira ordem

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1}, \tag{3.39}$$

sendo K o ganho DC e τ a constante de tempo do sistema.

Uma das formas de se determinar os parâmetros K e τ da função de transferência (3.39) é por meio da resposta ao degrau. Note que, aplicando-se um degrau de amplitude A ao sistema descrito pela equação (3.39), obtém-se a seguinte resposta $y(t)$:

$$y(t) = KA(1 - e^{-t/\tau}),\ t \geq 0. \tag{3.40}$$

Vamos considerar, inicialmente, a determinação do parâmetro K.

Determinação do parâmetro K

Note, inicialmente que:

$$\lim_{t\to\infty} y(t) = KA = y_\infty. \tag{3.41}$$

Portanto:

$$K = \frac{y_\infty}{A}. \tag{3.42}$$

Assim como no caso da resposta senoidal, na prática, porém, o sinal de resposta é corrompido por ruídos o que impede que valor de regime permanente de $y(t)$ seja determinado diretamente como na equação (3.42). Assim, o seguinte algoritmo deve ser usado para a determinação do parâmetro K.

Algoritmo 3.4 *(Determinação do parâmetro K)*

Passo 1. *Aplique ao sistema um degrau de amplitude A e faça a aquisição da resposta $y(t)$.*

Passo 2. *Determine um instante t_s a partir do qual se considera que o sistema esteja em regime permanente.*

Passo 3. *Encontre*

$$y_\infty = \frac{1}{N}\sum_{i=1}^{N} y(t_i),$$

sendo N o número de pontos $(t_i, y(t_i))$, $t_i \geq t_s$, e calcule

$$K = \frac{y_\infty}{A}.$$

Determinação do parâmetro τ

A determinação do parâmetro τ pode ser feita utilizando os seguintes métodos:

1. Método da área

Considere a área A_0 na figura 3.1, cuja curva representa a resposta real $y(t)$ de um sistema que desejamos modelar de acordo com a função de transferência (3.39). Note que A_0 pode ser escrita como:

$$A_0 = \int_0^\infty [KA - y(t)]dt.$$

Substituindo $y(t)$ na equação acima pela equação (3.40) e, após manipulações algébricas simples, obtém-se:

$$A_0 = KA\tau.$$

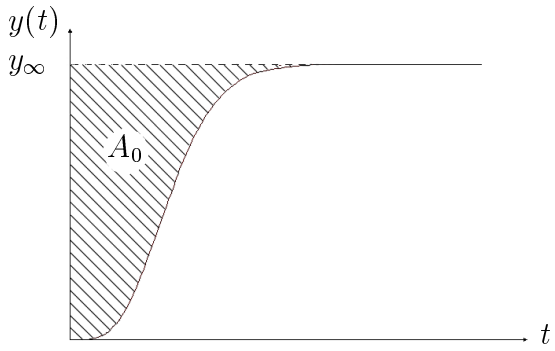


Figura 3.1: Resposta ao degrau monotonicamente crescente.

Assim, o seguinte algoritmo pode ser usado para a determinação do parâmetro τ.

Algoritmo 3.5 *(Determinação do parâmetro τ pelo método da área)*

Passo 1. *Aplique ao sistema um degrau de amplitude A e faça a aquisição da resposta $y(t)$.*

Passo 2. *Determine um instante t_s a partir do qual se considera que o sistema esteja em regime permanente.*

Passo 3. *Calcule a área A_0 utilizando a função* `trapz` *(*`a0 = trapz(t,yt)`*) do Matlab, na qual* `t` *é o vetor formado por todos os instantes de tempo $t_i \leq ts$ e* `yt` *o vetor formado pelos correspondentes valores de $y(t_i)$.*

Passo 4. *Calcule*

$$\tau = \frac{A_0}{y_\infty}.$$

2. Método do logaritmo neperiano

Um outra forma de se determinar a constante de tempo τ é por meio do chamado método do logaritmo neperiano. Para tanto, considere os instantes t tais que $y(t) < y_\infty = KA$. Manipulações algébricas simples, permitem escrever a equação (3.40) como:

$$\frac{1}{\tau}t = \ln\left(\frac{y_\infty}{y_\infty - y(t)}\right), \tag{3.43}$$

que pode, ainda, ser reescrita da seguinte forma:

$$at = b, \tag{3.44}$$

em que

$$a = \frac{1}{\tau} \tag{3.45}$$

e

$$b = b[y(t)] = \ln\left(\frac{y_\infty}{y_\infty - y(t)}\right). \tag{3.46}$$

Assim, o seguinte algoritmo pode ser usado para a determinação do parâmetro τ.

Algoritmo 3.6 *(Determinação do parâmetro τ pelo método do logaritmo neperiano)*

Passo 1. *Aplique ao sistema um degrau de amplitude A e fazer a aquisição da resposta $y(t)$.*

Passo 2. *Determine um instante t_r tal que $y(t_r) < y_\infty$ e isole os pontos $(t_i, y(t_i))$, $t_i \leq t_r$*

Passo 3. *Forme os vetores colunas $\underline{t}$ e $\underline{b}$ cujos i-ésimos elementos são, respectivamente, t_i e $b[y(t_i)]$, em que $(t_i, y(t_i))$, $t_i \leq t_r$, foram obtidos no passo 4.*

Passo 4. *Calcule a utilizando o método dos mínimos quadrados da seguinte forma:*

$$a = \frac{\underline{t}^T \underline{b}}{\|\underline{t}\|^2},$$

e, em seguida, determine

$$\tau = \frac{1}{a}.$$

Observação 3.1 *Uma outra forma de se determinar o parâmetro τ é por meio do método da tangente, que consiste na determinação da abscissa do ponto de interseção da reta tangente de maior coeficiente angular à curva de resposta ao degrau. Esse método não será, contudo, aqui considerado por ser extremamente dependente do traçado da reta tangente e, portanto, muito susceptível a erros.*

3.1.6 Identificação dos parâmetros K e τ da função de transferência de primeira ordem de um SLIT utilizando diagrama de módulo de Bode

Uma outra maneira de se obter a função de transferência de um SLIT estável

$$Y(s) = G(s)U(s), \tag{3.47}$$

em que $G(s)$ possui somente polos e zeros reais, é utilizando as curvas de resposta em frequência. Nesse caso, ao invés de utilizar as curvas de módulo e fase, $\omega \times |G(j\omega)|$ e $\omega \times \angle G(j\omega)$, utilizamos os diagramas de Bode de módulo e fase, $\omega(\log) \times |G(j\omega)|_{dB}$ e $\omega(\log) \times \angle G(j\omega)$, nos quais o eixo das abscissas é representado em escala logarítmica, $|G(j\omega)|_{dB} = 20 \log |G(j\omega)|$ e $\angle G(j\omega)$ é dada em graus. A grande vantagem de se utilizar os diagramas de Bode é que é possível determinar assíntotas da curva de módulo cujas interseções determinam as frequências de canto que correspondem aos polos e zeros de $G(s)$.

Considere novamente a equação (3.39) repetida abaixo:

$$G(s) = \frac{K}{\tau s + 1}. \tag{3.48}$$

Como

$$G(j\omega) = \frac{K}{1 + j\omega\tau} = \frac{K/\tau}{1/\tau + j\omega},$$

então é fácil verificar que

$$|G(j\omega)|_{dB} = 20\log\frac{K/\tau}{|1/\tau + j\omega|} = 20\log(K/\tau) - 20\log|1/\tau + j\omega|. \tag{3.49}$$

As assíntotas do diagrama de módulo podem ser, então, determinadas da seguinte forma:

$$|G(j\omega)|_{dB,ass} = \begin{cases} 20\log(K/\tau) - 20\log(1/\tau) = 20\log K, & \omega \ll 1/\tau, \\ 20\log(K/\tau) - 20\log\omega, & \omega \gg 1/\tau. \end{cases} . \tag{3.50}$$

Não é difícil verificar que as assíntotas se interceptam no ponto $(1/\tau, 20\log K)$, isto é, a abscissa do ponto de interseção das assíntotas, denominada frequência de canto, é numericamente igual ao polo de $G(s)$. A partir da equação (3.49, vemos que $|G(j1/\tau)|_{dB} = 20\log K - 20\log|1+j| \approx 20\log K - 3$. Assim sendo,

$$|G(j1/\tau)|_{dB,ass} - |G(j1/\tau)|_{dB} = 3\text{dB}. \tag{3.51}$$

Com base no exposto acima, podemos propor o seguinte algoritmo para a obtenção dos parâmetros K e τ de um sistema de primeira ordem estável a partir da sua resposta em frequência.

Algoritmo 3.7

Entrada *Conjunto de pontos* $(\omega_k, |G(j\omega_k)|)$, $k = 1, 2, \ldots, p$, *obtidos utilizando o algoritmo (3.3).*

Passo 1 *Forme os vetores* `w` *e* `gjwdb` *cujas componentes são, respectivamente,* ω_k *e* $20\log|G(j\omega_k)|$, $k = 1, 2, \ldots, p$ *e represente graficamente esses pontos (*`semilogx(w,gjwdb,'+')`*).*

Passo 2 *Ajuste os pontos de baixa frequência por uma reta horizontal utilizando o método dos mínimos quadrados (algoritmo (3.1)) e represente graficamente essa reta no mesmo gráfico obtido no passo anterior. Denote esse valor por* `Kdb`.

Passo 3 *Ajuste os pontos definidos pelos vetores* `w` *e* `gjwdb` *por um polinômio utilizando o método dos mínimos quadrados cujo melhor grau deve ser determinado por tentativa-e-erro da seguinte forma:*

3.1 *Defina um vetor* `ww` *de 200 frequências espaçadas logariticamente cujas primeira e última componentes são, respectivamente,* ω_1 *e* ω_p.

3.2 *Escolha um grau* n_0 *e ajuste os pontos* `w` *e* `gjwdb` *por um polinômio* `g` *de grau* n_0 *utilizando o algoritmo (3.1)*

3.3 *Calcule* `gww = polyval(g,ww)` *e represente esse pontos no gráfico definido nos passos 1 e 2 (*`semilogx(ww,gww)`*).*

3.4 *Se a curva determinada no passo 3.3 for uma boa aproximação para os pontos representados no passo 1, então vá para o passo seguinte. Caso contrário, retorne ao passo 3.2.*

Passo 4 *Determine a abscissa do ponto pertencente ao gráfico representado no passo 3.4 cuja ordenada é igual a* `Kdb - 3`*. Denote a abscissa encontrada por* `wc`.

Passo 5 *Calcule* $K = 10^{Kdb/20}$ *e constante de tempo* $\tau = wc$.

Saida *Ganho* K *e* τ

3.2 Modelo matemático de um motor CC controlado pela armadura

Considere o circuito equivalente de um motor CC controlado pela armadura da figura 3.2, na qual $i_a(t)$ e $v_a(t)$ denotam, respectivamente, a corrente e a tensão de armadura, V_f e I_f representam, respectivamente, a tensão e a corrente de campo (constantes, por hipótese), $\omega(t)$ é a velocidade angular do motor e J e f são o momento de inércia da carga e o coeficiente de atrito nos mancais, respectivamente.

Sabe-se que o torque produzido pelo motor, $t_m(t)$, é proporcional ao fluxo magnético no entreferro ($\phi(t)$) e à corrente de armadura ($i_a(t)$), isto é,

$$t_m(t) = K_f \phi(t) i_a(t) = K_m i_a(t) \tag{3.52}$$

em que $K_m = K_f \phi = K_f K_\phi I_f$, com K_f, K_ϕ e I_f constantes. Aplicando-se a lei das tensões de Kirchhoff ao circuito da armadura, obtém-se:

$$v_a(t) = R_a i_a(t) + L_a \frac{d}{dt} i_a(t) + e(t) \tag{3.53}$$

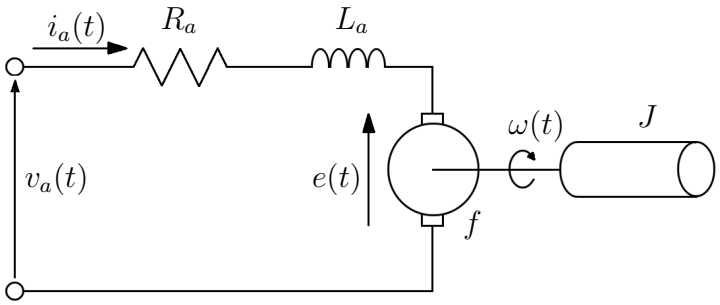


Figura 3.2: Circuito equivalente de um motor CC controlado pela armadura

sendo $e(t)$ a força contra-eletromotriz, que é proporcional à velocidade angular do motor, sendo dada por:

$$e(t) = K_e\omega(t) \tag{3.54}$$

Finalmente, usando-se a lei de Newton para o movimento rotacional, pode-se escrever:

$$t_m(t) - t_d(t) - f\omega(t) = J\frac{d}{dt}\omega(t) \tag{3.55}$$

onde $t_d(t)$ representa um torque externo (perturbação).

As equações (3.52) a (3.55) nos permitem obter a função de transferência que relaciona as transformadas de Laplace da entrada ($V_a(s)$) da saída ($W(s)$). Para tanto, aplicando-se a transformada de Laplace a ambos os membros das equações (3.52) a (3.55), resulta:

$$\left.\begin{array}{l} T_m(s) = K_m I_a(s) \\ V_a(s) = R_a I_a(s) + L_a s I_a(s) + E(s) \\ E(s) = K_e W(s) \\ T_m(s) - T_d(s) - fW(s) = JsW(s) \end{array}\right\} \tag{3.56}$$

e após alguma manipulação algébrica, obtém-se:

$$\begin{aligned} W(s) \;=\; & \frac{K_m/(R_a f)}{(\tau_e s+1)(\tau_m s+1) + K_e K_m/(R_a f)} V_a(s) - \\ & - \frac{(\tau_e s+1)/f}{(\tau_e s+1)(\tau_m s+1) + K_e K_m/(R_a f)} T_d(s) \end{aligned} \tag{3.57}$$

onde $\tau_e = L_a/R_a$ e $\tau_m = J/f$. Note que a função de transferência da equação (3.57) modela o motor CC como um sistema de 2a ordem. Porém, este sistema pode ser bem aproximado por um modelo de 1a ordem, se levarmos em conta que $L_a/R_a \ll 1$ e portanto, $\tau_e s + 1 \approx 1$ para as freqüências de interesse. Desta forma, o modelo matemático do motor CC que iremos adotar será o seguinte:

$$W(s) = \frac{K_a}{\tau s + 1} V_a(s) - \frac{K_d}{\tau s + 1} T_d(s) \tag{3.58}$$

onde $K_a = K_m/(R_a f + K_e K_m)$ e $K_d = R_a/(R_a f + K_e K_m)$ e $\tau = J R_a/(R_a f + K_e K_m)$.

Para o caso do nosso laboratório, consideraremos que não existe perturbação externa e, portanto o torque externo é nulo ($t_d(t) = 0$). Assim, a equação 3.58 pode ser simplificada para:

$$W(s) = \frac{K_a}{\tau s + 1} V_a(s). \tag{3.59}$$

É importante ressaltar que a medição da velocidade angular é feita por meio de tacômetros, conforme mostrado na figura 3.3. Um tacômetro nada mais é do que

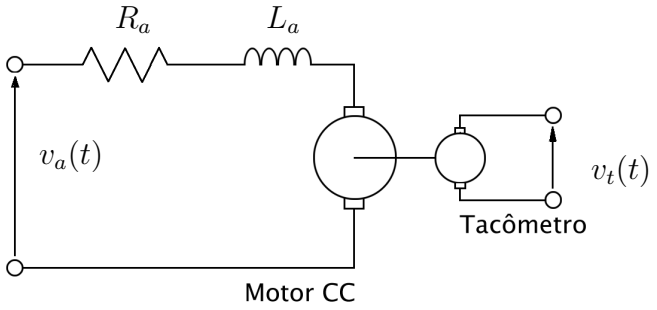


Figura 3.3: Circuito elétrico equivalente para o motor CC com sensor de velocidade (tacômetro)

um gerador CC de pequena potência, cuja tensão gerada é constante e proporcional à velocidade do eixo ao qual ele está acoplado. Desta forma, a tensão nos terminais do tacômetro, $v_t(t)$, será:

$$v_t(t) = K_t \omega(t), \tag{3.60}$$

onde K_t é uma constante. Portanto, a relação entre $V_t(s)$ e $V_a(s)$ pode ser expressa pela seguinte equação:

$$V_t(s) = \frac{K_a K_t}{\tau s + 1} V_a(s), \tag{3.61}$$

cujo diagrama de blocos está representado na figura 3.4.

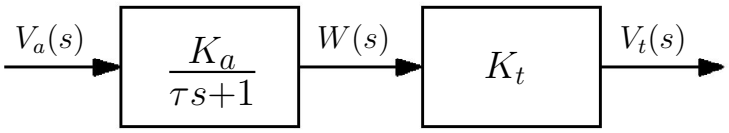


Figura 3.4: Diagrama de blocos representativo do modelo matemático do motor CC-tacômetro

3.3 Identificação dos parâmetros do motor CC

O modelo matemático do motor CC dado pela equação 3.61 considera que a planta seja linear. Entretanto, em geral, isto não é sempre verdade para todos os valores de $v_a(t)$ e, portanto, os estudantes devem obter a faixa de valores de $v_a(t)$ para que o sistema seja linear. Uma vez que a região linear de operação for definida, os estudantes podem executar o experimento para determinar os ganhos do sistema e a constante de tempo.

3.3.1 Identificação da região linear de operação

A região linear pode ser determinada da seguinte forma.

Algoritmo 3.8

Passo 1 *Aplique uma tensão constante no terminal de armadura V_a e meça o valor correspondente em regime permanente da tensão do tacômetro V_t;*

Passo 2 *Obtenha o gráfico $V_a \times V_t$ e utilize o método dos mínima quadrados para ajustar os pontos em um polinômio $p(V_a)$ de ordem desejada;*

Passo 3 *Calcule a derivada de $p(V_a)$ em relação a V_a. Portanto, a região linear corresponde ao intervalo em que a derivada é quase plana.*

3.3.2 Identificação de K_t

Para a determinação de K_t, observe a equação (3.60),

$$v_t(t) = K_t\omega(t). \tag{3.62}$$

Aplicando um degrau de tensão de amplitude V_a nos terminais do motor, isto é,

$$v_a(t) = \begin{cases} 0, & t < 0 \\ V_a, & t \geq 0 \end{cases} \tag{3.63}$$

e considere que $V_a(s) = V_a/s$, e portanto a equação (3.59) pode ser re-escrita da seguinte forma:

$$W(s) = \frac{K_a V_a}{s(\tau s + 1)}. \tag{3.64}$$

A resposta do sistema à entrada descrita em (3.63) pode ser obtida calculando-se a transformada inversa de Laplace da equação (3.64). Procedendo desta forma obtemos:

$$\omega(t) = K_a V_a(1 - e^{-\frac{1}{\tau}t}), \;\; t \geq 0 \tag{3.65}$$

Quando $t \to \infty$, a equação acima se reduz a (ver equação (3.41))

$$\omega(t) = W = K_a V_a, \tag{3.66}$$

o que mostra que quando uma tensão de valor constante é aplicada a um motor CC, a velocidade angular em regime permanente também será constante e proporcional ao valor da tensão aplicada.

Portanto, retornando a determinar do ganho do tacômetro K_t, considere a equação (3.62), cujo valor de tensão, em estado permanente, para uma entrada igual ao degrau de amplitude W é dado por:

$$V_t = K_t W. \tag{3.67}$$

Isto sugere que a identificação dos parâmetros de K_t pode ser feita pelo ajuste dos pontos (W_i, V_{t_i}) por uma reta passando pela origem, conforme a figura 3.5. Portanto, basta um único procedimento para obter tanto a região linear de operação e K_t.

Algoritmo 3.9

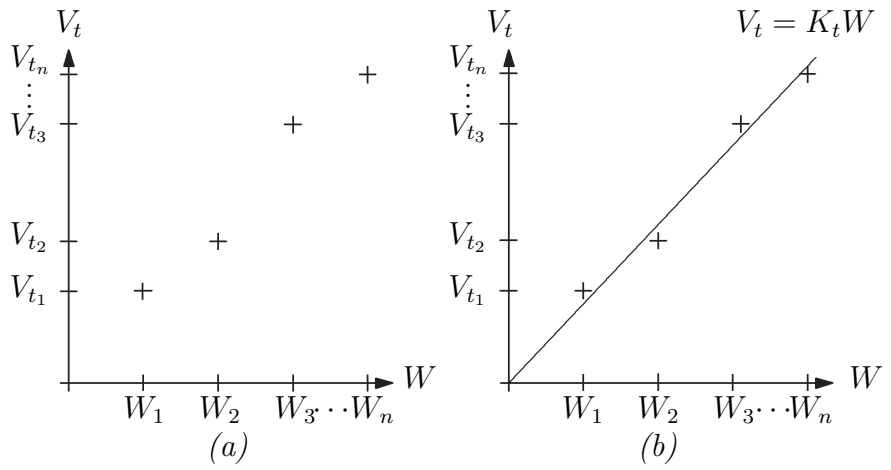


Figura 3.5: Representação *(a)* cartesiana dos pontos (W_1, V_{t_1}), (W_2, V_{t_2}), ..., (W_n, V_{t_n}) e *(b)* da reta $V_t = K_t W$

Passo 1 *Excite o motor CC com tensões constantes e iguais a* $V_{a_1}, V_{a_2}, \ldots, V_{a_n}$, *e meça os valores de tensão correspondentes,* $V_{t_1}, V_{t_2}, \ldots, V_{t_n}$, *nos terminais do tacômetro e as velocidades angulares do motor* W_1, W_2, ..., W_n (rpm), *utilizando-se tacômetros digitais ópticos.*

Passo 2 *Utilizando os pontos* (V_{a_1}, V_{t_1}), (V_{a_2}, V_{t_2}), ..., (V_{a_n}, V_{t_n}) *calcule a região linear pelo algoritmo 3.8.*

Passo 3 *Use o método dos mínimos quadrados para ajustar por uma reta passando pela origem* ($V_t = K_t W$) *aos pontos obtidos no passo anterior (veja figura 3.5).*

3.3.3 Identificação de $\mathbf{K_a}$ e τ

Uma vez que se dispõe da região linear e do ganho K_t, o próximo passo é a identificação do ganho K_a e da constante de tempo τ. Definindo $K = K_a K_t$, então a equação (3.61) se torna idêntica á equação (3.39 em que $V_t(s) = Y(s)$ e $V_a(s) = U(s)$. Assim, para a determinação do parâmetro K_a basta determinar o valor de K utilizando os algoritmos das seções 3.1.5 e 3.1.6. Os mesmos algoritmos podem ser usados para a determinação do parâmetro τ, conforme descrito a seguir:

1. Utilizando a resposta ao degrau

   - Determinação de K: algoritmo 3.4;
   - Determinação de τ utilizando os métodos da área e do logaritmo neperiano: algoritmos 3.5 e 3.6, respectivamente.

2. Utilizando a resposta em frequência

   - Determinação de K e τ: algoritmo 3.7.

3.4 Experimento para identificação da função de transferência motor CC

Uma vez que já dispomos do modelo matemático do sistema e do embasamento teórico necessário para a determinação de seus parâmetros, podemos agora descrever os ensaios para obter os dados experimentais necessários para a identificação os parâmetros da função de transferência do motor CC. Este experimento é composto, basicamente de três partes: *(i)* excitação do motor CC com tensões constantes com vistas à identificação da região linear e do ganho K_t; *(ii)* excitação do motor CC com diferentes tensões constantes com vistas à identificação K_a e τ; *(iii)* excitação do sistema com sinais senoidais de diferentes freqüências objetivando-se obter a resposta em freqüência do sistema para que, a partir da curva de módulo dos diagramas de Bode identificar K_a e τ.

No experimento para a identificação da função de transferência do motor CC, serão utilizados os seguintes equipamentos:

1. Osciloscópio digital com, pelo menos, dois canais;
2. Gerador de funções;
3. Fonte de tensão CC (regulável);
4. Tacômetro óptico;
5. Multímetros digitais (3);
6. Amplificador de potência;

3.4.1 Experimento para determinação da região linear e do ganho K_t

São os seguintes os passos necessários para a obtenção dos dados que levam à determinação da região linear e do parâmetro K_t:

1. Conecte as saídas da fonte de tensão CC aos terminais do motor.

2. Conecte um multímetro à saída da fonte de tensão e um outro multímetro aos terminais do tacômetro.

3. Excite o motor com o primeiro valor de tensão (V_a) sugerido na primeira coluna da tabela 3.4.1, anotando na segunda coluna o valor da tensão efetivamente aplicada. Em seguida, meça a tensão nos terminais do tacômetro (V_t) e a velocidade angular do motor (ω), utilizando um tacômetro óptico, preenchendo, respectivamente, as colunas 3 e 4 da tabela 3.4.1.

4. Repita o passo 3 para os demais valores de tensão sugeridos na tabela 3.4.1.

3.4.2 Experimento para determinação do ganho K_a e τ utilizando resposta ao degrau

Para a obtenção da curva de resposta ao degrau do sistema é necessário excitar o motor com uma degrau de tensão. Os geradores de funções de que o laboratório dispõe não têm potência suficiente para fazer o motor funcionar. Para superar este problema, faz-se passar o sinal fornecido pelo gerador de funções por um amplificador de potência, que fornecerá, então, a potência necessária para o motor girar. Com isso em mente, siga os passos seguintes para obter os dados necessários para o levantamento da resposta ao degrau do sistema:

1. Conecte os terminais do gerador de funções aos terminais de entrada do amplificador de potência. Em seguida conecte os terminais de saída do amplificador de potência aos terminais do motor.

2. Conecte os terminais de saída do amplificador de potência ao canal 1 do digitalizador e os terminais do tacômetro ao canal 2 do digitalizador.

3. Excite o motor com uma tensão de onda quadrada com dois estágios, um de valor acima da região linear e outro de aproximadamente $6,7,8V$ de amplitude. Em seguida, armazene a resposta $v_(t)$ para cada caso.

V_a (V) sugerida	V_a (V) medida	V_t (V)	ω (rpm)
0,0			
0,5			
1,0			
1,5			
2,0			
2,3			
2,6			
2,9			
3,3			
3,6			
3,9			
4,0			
5,0			
6,0			
7,0			
8,0			
9,0			
10,0			
11,0			
12,0			
13,0			
14,0			
15,0			

Tabela 3.1: Experimento para determinação do parâmetro K_t

3.4.3 Experimento para obtenção da resposta em freqüência

Para a obtenção da curva de resposta em freqüência do sistema é necessário excitar o motor com uma tensão senoidal. Para tanto, siga os passos seguintes para obter os dados necessários para o levantamento da curva de resposta em freqüência do sistema:

1. Conecte os terminais do gerador de funções aos terminais de entrada do amplificador de potência. Em seguida conecte os terminais de saída do amplificador de potência aos terminais do motor.

2. Conecte os terminais de saída do amplificador de potência ao canal 1 do digitalizador e os terminais do tacômetro ao canal 2 do digitalizador.

3. Encontre qual a menor e a maior frequência que conseguiremos observar $v_t(t)$ e, em seguida, determine a faixa de frequência utilizando o comando *logspace*. Após isso, calcule qual seria o período de amostragem se quisermos armazenar dois período em 500 pontos para cada frequência.

4. Excite o motor com uma tensão senoidal de aproximadamente $8V$ de amplitude, variando a freqüência da tensão senoidal encontrada no item acima.

3.5 Validação do modelo

Tendo sido determinados os parâmetros da função de transferência do sistema, o passo seguinte é a validação do modelo obtido, *i.e*, verificar se os valores calculados a partir dos experimentos de identificação são tais que o modelo matemático adotado represente, com precisão aceitável, o sistema real. No caso do motor CC (sistema adotado para este laboratório) foram realizados ensaios para se determinar os parâmetros K_a, K_t, K e τ. Note que:

1. Os ganhos K_a e K_t foram identificados a partir do ajuste dos coeficientes angulares de retas de acordo com o algoritmo 3.9. A exatidão dos valores encontrados para K_a e K_t pôde ser verificada a partir da comparação com o produto K_aK_t e K.

2. A constante de tempo τ foi calculada de duas formas: *(i)* a partir da determinação da freqüência de canto do diagrama de módulo de Bode e *(ii)* a partir do ajuste do diagrama polar de $K_aK_t/(\tau s + 1)$ a um conjunto de pontos obtidos experimentalmente. O valor de τ será escolhido a partir de simulações em computador, comparando-se os resultados dessas com o desempenho do sistema real.

Assim sendo, para a validação do modelo matemático do motor CC, dado na figura 3.4, proceda da seguinte forma:

1 Encontre o erro percentual ($E_{K_aK_t}(\%)$) entre os produtos de K_a eK_t e de K, isto é:

$$E_{K_aK_t}(\%) = \frac{K_aK_t - K}{K_aK_t}100(\%) \tag{3.68}$$

Se o erro for menor que 1%, os valores de K_a e K_t obtidos experimentalmente podem ser adotados como representativos do modelo, com razoável grau de confiança. Caso o erro seja maior que 1%, deve-se, inicialmente, verificar se os resultados obtidos nos ensaios de laboratório foram de fato utilizados no cálculo de K_a, K_t e K_aK_t. Isto permitirá que se encontre possíveis discrepâncias entre os valores obtidos pelos três métodos, evitando que todos os experimentos realizados para a determinação de K_a, K_t e K tenham que ser realizados novamente. Caso não tenha sido encontrado qualquer erro de manipulação dos dados, o aluno deve retornar ao laboratório para realizar novamente aqueles experimentos cujos valores obtidos inicialmente apresentam maiores discrepâncias.

2 Excite o motor CC com um degrau de tensão de amplitude igual a $7V$ e faça a aquisição dos sinais[1] de entrada ($v_a(t)$) e de saída ($v_t(t)$) e dos correspondentes instantes de tempo (t).

3(a) Com os valores de K_a e K_t, obtidos no passo 1, e com o valor de τ, obtido a partir do diagrama de módulo de Bode, construa um modelo em SIMULINK equivalente ao diagrama de blocos representativo motor CC dado pela figura **??** no qual o sinal de entrada será $t \times v_a(t)$ (obtidos no passo 2). Para tanto, crie dois vetores colunas `t` e `va`, cujas componentes são as correspondentes abscissas e ordenadas do sinal de entrada e utilize o bloco `from workspace` como entrada, sendo `t` e `va` os parâmetros. Realize uma simulação tendo como instantes inicial e final, `t(1)` e `t(length(t))`, respectivamente. Note que, após a simulação serão gerados dois vetores: `ts`, que corresponde aos instantes de tempo utilizados na simulação e `vts`, que corresponde à tensão nos terminais do tacômetro para o modelo obtido. Represente, em um mesmo gráfico, as curvas $t \times v_a(t)$, $t \times v_t(t)$ e $t_s \times v_t(t_s)$, onde t_s é um intervalo cujos extemos são `ts(1)` e `ts(length(ts)`.

3(b) Proceda de forma idêntica à anterior, porém com o valor de τ obtido a partir do ajuste do diagrama polar.

[1]Esta aquisição de dados pode ser feita utilizando-se um computador com placas de aquisição de dados ou um osciloscópio digital com capacidade de armazenamento em disco.

3(c) Entre as curvas $t_s \times v_t(t_s)$ (representadas graficamente nos item 3(a) e 3(b)), escolha aquela que mais se aproxima da resposta do sistema real $t \times v_t(t)$. O valor de τ que corresponde ao modelo que produziu a curva mais próxima será o escolhido.

É de se esperar que as curvas obtidas no trabalho de simulação não sejam exatamente coincidentes com as curvas obtidas para o sistema real. Porém, para que o modelo adotado possa ser considerado satisfatório, essas curvas devem ser bastante próximas. Qualquer discrepância mais acentuada pode desqualificar o experimento de identificação ou até mesmo o modelo matemático; por exemplo, levando à necessidade de escolha de um modelo não-linear.

Capítulo 4

Projeto do controlador de velocidade

Obtido um modelo matemático para o sistema, o passo seguinte é projetar um controlador que satisfaça as seguintes exigências: *(i)* estabilidade; *(ii)* rastreamento assintótico de um sinal de referência (*i.e.* para uma dada velocidade, o motor deve, em estado permanente, girar nessa velocidade); *(iii)* rejeição assintótica de perturbações (*i.e.* quando uma carga for inserida nos terminais do gerador, o motor, após um 'pequeno' transitório, deve permanecer com a mesma rotação de antes da introdução da carga); *(iv)* o sistema compensado não deve ter o seu desempenho comprometido por eventuais erros no modelo, cometidos durante a fase de identificação, o que significa dizer que o sistema tem elevada robustez ou equivalentemente pouca sensibilidade a variações nos parâmetros da função de transferência da planta) e, finalmente, *(v)* bom desempenho transitório.

O objetivo deste capítulo é desenvolver o projeto de um controlador de velocidade para o grupo motor-gerador que satisfaça as condições *(i)* a *(v)* acima. Como finalidade didática, será, inicialmente, considerada a possibilidade de se fazer a compensação em malha aberta para, em seguida, projetar um sistema de controle realimentado. Esta abordagem terá a finalidade de ilustrar os benefícios da realimentação num sistema de controle.

Este capítulo está estruturado da seguinte forma: na seção 4.1 serão apresentados os fundamentos teóricos necessários para o projeto de um controlador que rastreie assintoticamente um sinal de referência, $R(s)$, e rejeite (também assintoticamente) uma perturbação, $D(s)$, conforme a figura 4.1. Outro tópico a ser abordado na seção 4.1 é a chamada sensibilidade de um sistema em relação à variação

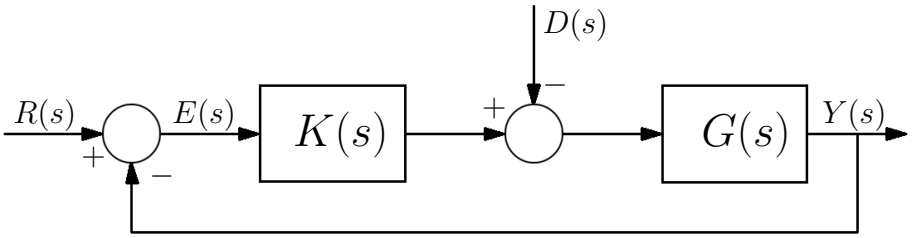


Figura 4.1: Diagrama de blocos para o projeto de controladores com objeticos de rastreamento assintótico de um sinal referência e rejeição assintótica de perturbação

de parâmetros da função de transferência da planta. Na seção 4.2 será considerado o projeto de um controlador que satisfaça as exigências *(i)* a *(v)*. Vale lembrar que um bom domínio do método do lugar das raízes é fundamental para o projeto de compensadores, principalmente no que se refere à melhoria da resposta transitória. Outros conceitos também importantes são aqueles refentes à resposta transitória de um sistema de segunda ordem. Caso esses assuntos não estejam bem sedimentados, os alunos são aconselhados a revisá-los.

4.1 Fundamentos teóricos

4.1.1 Rastreamento e rejeição assintótica de sinais de dinâmica conhecida

Considere o sistema realimentado da figura 4.1 onde, $G(s)$ e $K(s)$ representam as funções de transferências da planta e do controlador, respectivamente, sendo

$$G(s) = \frac{n_G(s)}{d_G(s)} \text{ e } K(s) = \frac{n_K(s)}{d_K(s)}, \tag{4.1}$$

$n_G(s)$ e $d_G(s)$ são polinômios conhecidos e $n_K(s)$ e $d_K(s)$ são polinômios a serem determinados. Sejam $R(s)$, $D(s)$ e $Y(s)$ as transformadas de Laplace dos sinais de referência, de perturbação externa e de saída, onde

$$R(s) = \frac{\alpha(s)}{\beta(s)} \text{ e } D(s) = \frac{\gamma(s)}{\delta(s)}, \tag{4.2}$$

$\beta(s)$ e $\delta(s)$ são polinômios conhecidos (dinâmicas de $R(s)$ e $D(s)$, respectivamente), $\alpha(s)$ e $\gamma(s)$ são também polinômios, porém, como será visto mais adiante, não são necessariamente conhecidos.

A partir do diagrama de blocos da figura 4.1, pode-se escrever:

$$Y(s) = -G(s)D(s) + G(s)K(s)R(s) - G(s)K(s)Y(s) \tag{4.3}$$

e, conseqüentemente:

$$Y(s) = \frac{G(s)K(s)}{1 + G(s)K(s)}R(s) - \frac{G(s)}{1 + G(s)K(s)}D(s). \tag{4.4}$$

A partir da equação (4.4), pode-se observar que o sinal de saída $Y(s)$ possui duas componentes: *(i)* $Y_R(s)$, que é devida ao sinal de referência $R(s)$ e *(ii)* $Y_D(s)$, que se deve à perturbação externa $D(s)$. Portanto, $Y(s)$ pode ser escrito como:

$$Y(s) = Y_R(s) - Y_D(s) \tag{4.5}$$

onde

$$Y_R(s) = \frac{G(s)K(s)}{1 + G(s)K(s)}R(s) \text{ e } Y_D(s) = \frac{G(s)}{1 + G(s)K(s)}D(s) \tag{4.6}$$

O rastreamento e a rejeição de perturbações externas (assintoticamente) requerem que $y(t) \to r(t)$ quando $t \to \infty$, o que equivale a exigir que

$$\lim_{t\to\infty} y_D(t) = 0 \tag{4.7}$$

e

$$\lim_{t\to\infty} e_R(t) = 0 \tag{4.8}$$

onde

$$e_R(t) = y_R(t) - r(t). \tag{4.9}$$

Os problemas do rastreamento assintótico de um sinal de referência e da rejeição assintótica de um sinal externo de perturbação (ambos de dinâmicas conhecidas) serão abordados considerando-se as condições impostas pelas equações (4.7)–(4.9), acima. Antes, porém, considere a seguinte fatoração: seja $p(s)$ um polinômio e fatore $p(s)$ como $p(s) = p^-(s)p^+(s)$, onde $p^-(s)$ é um polinômio de Hurwitz cujos zeros são os zeros de $p(s)$ com parte real negativa e $p^+(s)$ é um polinômio cujos zeros são os zeros de $p(s)$ com parte real positiva ou nula. Por exemplo, o polinômio

$p(s) = s^3 - 2s^2 - 3s$ pode ser fatorado como $p^-(s)p^+(s)$, onde $p^-(s) = s + 1$ e $p^+(s) = s(s - 3)$.

Condições necessárias e suficientes para que o sistema da figura 4.1 rejeite assintoticamente um sinal $D(s) = \gamma(s)/\beta(s)$, $\beta(s)$ conhecido, são apresentadas no seguinte teorema.

Teorema 4.1 Seja $K(s)$ um controlador que estabiliza $G(s)$, isto é, $K(s)$ é tal que o sistema realimentado da figura 4.1 é estável e seja $D(s) = \gamma(s)/\delta(s)$, $\delta(s)$ conhecido. Então

$$\lim_{t\to\infty} y_D(t) = 0, \tag{4.10}$$

isto é, o sistema rejeitará assintoticamente o sinal de perturbação $d(t)$, se e somente se

$$n_G(s)d_K(s) = \chi(s)\delta^+(s) \tag{4.11}$$

onde $\delta^+(s)$ é um polinômio formado com os zeros de $\delta(s)$ com parte real positiva ou nula e $\chi(s)$ é um polinômio qualquer.

Prova: Usando as notações das equações (4.1) e (4.2), tem-se que, a expressão (4.6) pode ser escrita como:

$$Y_D(s) = \frac{n_G(s)d_K(s)}{[n_G(s)d_K(s) + d_G(S)d_K(s)]} \frac{\gamma(s)}{\delta^+(s)\delta^-(s)} \tag{4.12}$$

Note que

$$n_G(s)d_K(s) + d_G(S)d_K(s) = p_C(s), \tag{4.13}$$

onde $p_C(s)$ denota o polinômio característico de malha fechada. Como, por hipótese, $K(s)$ estabiliza $G(s)$ então $p_C(s)$ é um polinômio de Hurwitz e, portanto,

$$\lim_{t\to\infty} y_D(t) = \lim_{s\to 0} sY_D(s) = 0 \tag{4.14}$$

se e somente se $\delta^+(s)$ for um divisor de $n_G(s)d_K(s)$, ou equivalentemente, se existir um polinômio $\chi(s)$ tal que $n_G(s)d_K(s) = \chi(s)\delta^+(s)$ □

0 teorema 4.1 acima mostra que para que haja rejeição assintótica de um sinal externo de perturbação aplicado na entrada da planta, as dinâmicas desse sinal que possuem parte real positiva ou nula devem ser zeros da planta ou pólos do

controlador. Vale ressaltar que, somente em casos especiais, essas dinâmicas serão também zeros da função de transferência da planta e portanto, para se conseguir a rejeição assintótica de um sinal de perturbação, é mais comum fazer com que essas dinâmicas sejam também pólos do controlador, isto é,

$$K(s) = \frac{1}{\delta^+(s)}\bar{K}(s) \tag{4.15}$$

onde $\bar{K}(s) = \bar{n}_K(s)/\bar{d}_K(s)$, $\bar{n}_K(s)$ e $\bar{d}_K(s)$ serão escolhidos de tal forma que $n_G(s)\bar{n}_K(s) + d_G(s)\bar{d}_K(s)\delta^+(s)$ seja um polinômio de Hurwitz.

Uma vez obtida uma condição necessária e suficiente para a rejeição assintótica de um sinal externo de perturbação, o passo seguinte é considerar o problema do rastreamento assintótico de um sinal de referência. A abordagem deste problema é feita de forma análoga à anterior, levando à condição necessária e suficiente do teorema seguinte.

Teorema 4.2 Seja $K(s)$ um controlador para o qual o sistema realimentado da figura 4.1 seja estável e assuma que a transformada de Laplace do sinal de referência $r(t)$ é $R(s) = \alpha(s)/\beta(s)$, $\beta(s)$ conhecido. Então

$$\lim_{t\to\infty} e_R(t) = 0, \tag{4.16}$$

isto é, $y(t)$ rastreia assintoticamente $r(t)$, se e somente se

$$d_G(s)d_K(s) = \eta(s)\beta^+(s) \tag{4.17}$$

onde $\beta^+(s)$ é um polinômio formado com os zeros de $\beta(s)$ com parte real positiva ou nula e $\eta(s)$ é um polinômio qualquer.

Prova: A partir das equações (4.6) e (4.9), pode-se escrever:

$$\begin{aligned} E_R(s) &= R(s) - Y_R(s) \\ &= \frac{1}{1 + G(s)K(s)}R(s) \end{aligned} \tag{4.18}$$

e usando a notação da equação (4.1) tem-se:

$$E_R(s) = \frac{d_G(s)d_K(s)}{[n_G(s)n_K(s) + d_G(s)d_K(S)]}\frac{\alpha(s)}{\beta^+(s)\beta^-(s)}. \tag{4.19}$$

Como, por hipótese, $G(s)$ estabiliza $K(s)$, então $p_c(s) = n_G(s)n_K(s) + d_G(s)d_K(s)$ é um polinômio de Hurwitz e, portanto,

$$\lim_{t\to\infty} e_R(t) = \lim_{s\to\infty} sY_R(s) = 0 \tag{4.20}$$

se e somente se $d_G(s)d_K(s)$ for múltiplo de $\beta^+(s)$, ou equivalentemente, se existir um polinômio $\eta(s)$ tal que $d_G(s)d_K(s) = \eta(s)\beta^+(s)$. □

De acordo com o teorema 4.2, para que haja rastreamento assintótico de um sinal de referência, as dinâmicas desse sinal com parte real positiva ou nula devem ser pólos do controlador ou da planta. Observe que, somente em casos muito especiais haverá coincidência dos pólos da planta com essas dinâmicas. Portanto, de uma forma geral, o rastreamento assintótico será obtido fazendo-se:

$$K(s) = \frac{1}{\beta^+(s)}\bar{K}(s) \tag{4.21}$$

onde $\bar{K}(s) = \bar{n}_K(s)/\bar{d}_K(s)$, $\bar{n}_K(s)$ e $\bar{d}_K(s)$ serão tais que $n_G(s)\bar{n}_K(s)+d_G(s)\bar{d}_K(s)\beta^+(s)$ seja estável.

Finalmente, note que os objetivos de rastreamento assintótico de um sinal de referência $R(s) = \alpha(s)/[\beta^-(s)\beta^+(s)]$ e a rejeicão assintótica de um sinal externo de perturbação $D(s) = \gamma(s)/[\delta^+(s)\delta^-(s)]$ serão, em geral, atingidos simultaneamente, com um mesmo controlador, fazendo-se

$$K(s) = \frac{1}{\text{mmc}[\beta^+(s), \delta^+(s)]}\bar{K}(s) \tag{4.22}$$

onde $\text{mmc}[\beta^+(s), \delta^+(s)]$ denota o mínimo múltiplo comum de $\beta^+(s)$ e $\delta^+(s)$ e $\bar{K}(s) = \bar{n}_K(s)/\bar{d}_K(s)$, com $\bar{n}_K(s)$ e $\bar{d}_K(s)$ sendo calculados de forma que

$$p_C(s) = n_G(s)\bar{n}_K(s) + d_G(s)\bar{d}_K(s)\text{mmc}[\beta^+(s), \delta^+(s)] \tag{4.23}$$

seja um polinômio de Hurwitz.

Exemplo 4.1: Suponha que para o sistema realimentado da figura 4.1, $R(s) = A/s$ e $D(s) = B/s$, onde $A, B \in \mathbb{R}$ são as amplitudes dos degraus de referência e de perturbação, respectivamente. Portanto, para que sejam atingidos, simultaneamente, os objetivos de rastreamento assintótico de $r(t)$ e rejeição, também assintótica, de $d(t)$, o controlador deve possuir ação integral, isto é,

$$K(s) = \frac{1}{s}\bar{K}(s) \tag{4.24}$$

$\bar{K}(s)$ sendo calculado de tal sorte que o sistema realimentado seja estável com bom desempenho transitório. Guarde bem este resultado! Ele será importante quando do projeto de um controlador de velocidade para o grupo motor-gerador.

4.1.2 Sensibilidade

A sensibilidade de um sistema está associada à não manutenção do desempenho e da estabilidade em presença de variações nos parâmetros da planta. Essas variações são, em geral, devidas a: *(i)* erros de identificação da função de transferência do sistema, uma vez que os valores dos parâmetros, obtidos a partir da identificação, jamais representarão exatamente o sistema real e *(ii)* envelhecimento dos componentes do sistema. Desta forma, é importante analisar como o sistema irá se comportar em presença de tais variações (comumente denominadas de incertezas no modelo), seja no que se refere ao desempenho do sistema compensado, ou com relação à manutenção da estabilidade.

A sensibilidade de um sistema em relação à variação em um determinado elemento é quantificada pela razão entre a variação percentual da função de transferência do sistema e a variação percentual da função de transferência do elemento considerado, *i.e.*,

$$S_G^T(s) = \frac{\frac{\Delta T(s)}{T(s)}100\%}{\frac{\Delta G(s)}{G(s)}100\%} \tag{4.25}$$

onde $T(s)$ representa a função de transferência global do sistema e $G(s)$ é a função de transferência do elemento considerado (em geral, a planta). A implicação imediata da equação (4.25) acima é que, quão mais perto de 1 for S_G^T, mais sensível a variações nos parâmetros de $G(s)$ será o sistema como um todo. Observe que a equação (4.25) pode ser escrita como:

$$S_G^T(s) = \frac{\Delta T(s)}{\Delta G(s)}\frac{G(s)}{T(s)} \tag{4.26}$$

e, portanto, quando $\Delta G(s) \to 0$ tem-se que $\Delta T(s) \to 0$ e conseqüentemente,

$$S_G^T(s) = \frac{G(s)}{T(s)}\frac{\partial T(s)}{\partial G(s)}. \tag{4.27}$$

A redução da sensibilidade do sistema à variação dos parâmetros da função de transferência da planta é uma das razões para se usar a realimentação. Este fato será evidenciado no exemplo seguinte.

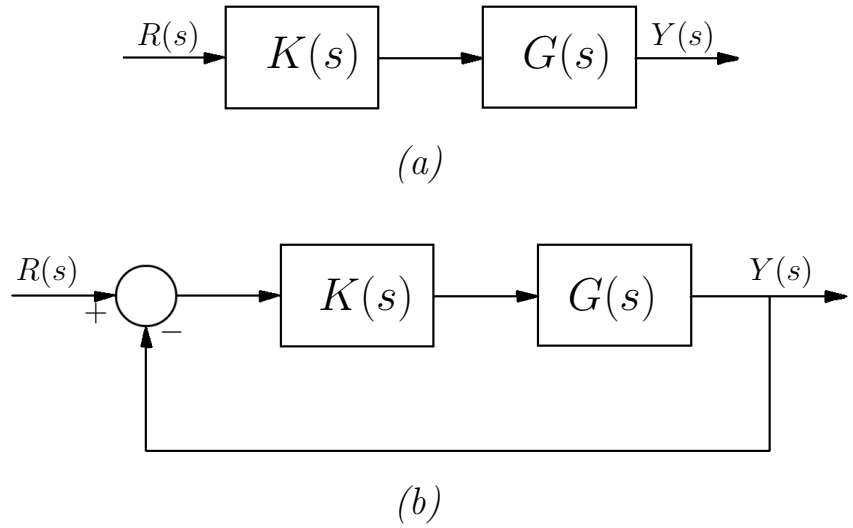


Figura 4.2: Diagrama de blocos: *(a)* sistema em malha aberta; *(b)* sistema realimentado

Exemplo 4.2: Sejam os sistemas em malha aberta e fechada representados na figura 4.2. Calcule S_G^T para ambos os casos.

Consideremos inicialmente o sistema em malha aberta. Neste caso $T(s) = G(s)K(s)$ e, portanto:

$$S_G^T = \frac{G}{T}\frac{\partial T}{\partial G} = \frac{G}{GK}K = 1 \tag{4.28}$$

que revela que qualquer erro de identificação ou variação nos parâmetros de $G(s)$ irá implicar num erro, de mesma magnitude na função de transferência (e por conseqüinte, no desempenho) do sistema como um todo.

Para o sistema em malha fechada, tem-se que $T(s) = G(s)K(s)/[1+G(s)K(s)]$, e então:

$$\begin{aligned} S_G^T &= \frac{G}{T}\frac{\partial T}{\partial G} \\ &= G\frac{1+GK}{GK}\frac{(1+GK)K - GK^2}{(1+GK)^2} \\ &= \frac{1}{1+GK} \end{aligned} \tag{4.29}$$

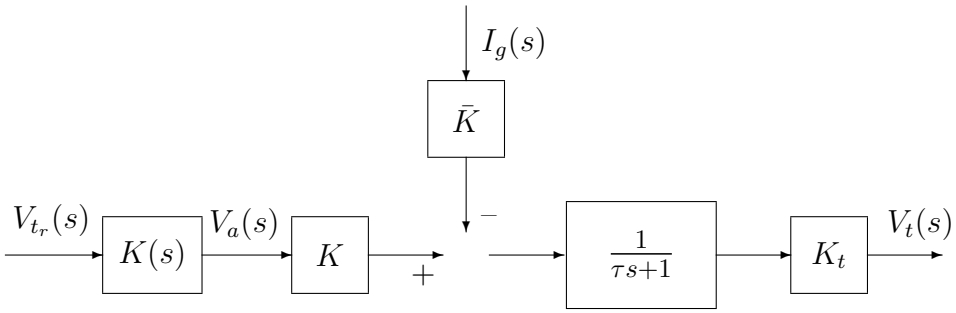


Figura 4.3: Diagrama de blocos para o sistema em malha aberta

Suponha que $K(s)$ estabiliza o sistema da figura 4.2(b) então, para uma dada freqüência ω_0, S_G^T será reduzida de 1 (sistema em malha aberta) para

$$\mid S_G^T(j\omega_0) \mid = \frac{1}{\mid 1 + G(j\omega_0)K(j\omega_0) \mid}, \tag{4.30}$$

que é menor que 1, evidenciando o benefício da realimentação no que se refere à diminuição da sensibilidade do sistema à variação nos parâmetros de $G(s)$. □

4.1.3 Complementos

(a) Compensação em malha aberta

Suponha que se deseje fazer uma compensação em malha aberta conforme o diagrama de blocos da figura 4.3 e seja $K(s) = K_p$. Nessas condições:

(1) Para quais valores de K_p, o sistema será estável?

(2) Entre os valores de K_p que tornam o sistema estável, encontre aquele que faz com que o sistema tenha um erro de regime permanente nulo para uma entrada igual ao degrau de amplitude V_r.

(b) Compensação em malha fechada (controlador integral)

Suponha, agora, que o objetivo seja projetar um sistema de controle realimentado para o sistema, conforme o diagrama de blocos da figura 4.4. Conforme visto na seção 4.1.1, exemplo 4.1, a estrutura mais simples para $K(s)$ de tal sorte que o sistema realimentado rastreie e rejeite assintoticamente um degrau é $K(s) =$

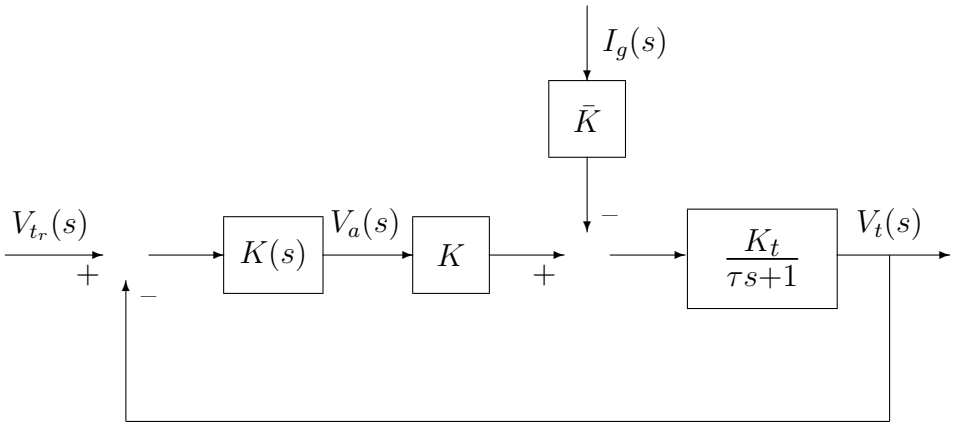


Figura 4.4: diagrama de blocos para o sistema realimentado

K_I/s, onde K_I é calculado para que o sistema realimentado seja estável e com bom desempenho transitório. Construa o gráfico do lugar das raízes do sistema e responda:

(1) Para quais valores de K_I, o sistema realimentado será estável?

(2) Para quais valores de K_I, o sistema realimentado será: *(i)* superamortecido; *(ii)* criticamente amortecido; *(iii)* sub-amortecido e *(iv)* subamortecido com percentual menor ou igual a 5%.

(c) Compensação em malha fechada (controlador proporcional + integral)

Seja, agora

$$K(s) = K_p \frac{s+z}{s} = K_p \left(1 + \frac{z}{s}\right) = K_p \left(1 + \frac{1}{T_i s}\right) \tag{4.31}$$

onde $T_i = 1/z$, K_p e $z > 1/\tau$ a serem determinados. Esboce o gráfico do lugar das raízes do sistema realimentado da figura 4.4 com o controlador (4.31) e responda:

(1) Para quais valores de K_p, o sistema realimentado será estável?

(2) Para qual valor de K_p, os pólos do sistema realimentado serão complexos e com parte real igual a $-1/\tau$?

4.2 Projeto do controlador de velocidade

Como o objetivo do sistema de controle a ser aqui desenvolvido é controlar a velocidade de rotação do motor, isto é, mantê-la em um determinado valor, é natural considerar como sinal de referência um degrau de amplitude V_r Volts, isto é:

$$v_{t_r}(t) = \begin{cases} V_r\ (V), & t \geq 0 \\ 0, & t < 0 \end{cases} \tag{4.32}$$

Note que está implícito no objetivo de rastreamento do sinal $v_r(t)$, a necessidade do sistema ser estável. Além desses objetivos (estabilidade e rastreamento assintótico do sinal de referência) o sistema deve:

1. Rejeitar sinais de perturbação, que podem ser modelados como degraus de amplitude I_g Ampères, isto é:

$$i_g(t) = \begin{cases} I_g\ (A), & t \geq t_0 \\ 0, & t < t_0 \end{cases} \tag{4.33}$$

   onde, $t_0 \geq 0$.

2. Baixa sensibilidade a erros de identificação dos parâmetros do modelo do grupo motor gerador.

3. Bom desempenho transitório. Como se trata de uma planta didática, o desempenho do sistema será definido unicamente um termos do tempo de acomodação da resposta ao degrau do sistema em malha aberta.

É sabido que os objetivos de um sistema de controle somente serão alcançados com sistemas realimentados. Porém, como forma de ilustrar os benefícios da realimentação, iremos, inicialmente, considerar a possibilidade de se usar um controle em malha aberta e, em seguida, projetar um sistema de controle realimentado.

4.2.1 Sistema de controle em malha aberta

A partir do diagrama de blocos da figura 4.3 pode-se escrever:

$$V_t(s) = \frac{K}{\tau s + 1} K(s) V_{t_r}(s) - \frac{\bar{K}}{\tau s + 1} I_g(s). \tag{4.34}$$

É fácil verificar que o sistema de controle em malha aberta será estável se e somente se $K(s)$ for estável. Como, por simplicidade foi adotado como satisfatório o tempo de acomodação do sistema sem compensação, pode-se utilizar um controlador estático, isto é:

$$K(s) = K_P \tag{4.35}$$

onde K_P deve ser determinado de acordo com a seção 4.1.3(a) de tal sorte que, em regime permanente, $v_t(t) = V_r$.

Uma vez obtido K_P, o passo seguinte é fazer a análise do desempenho do sistema compensado, utilizando o SIMULINK. Para tanto, construa um modelo, em SIMULINK, equivalente ao diagrama de blocos da figura 4.3. Em seguida, tendo $0s$ e $2s$ como instantes inicial e final de simulação, proceda à seguinte simulação:

1. Para uma corrente $i_g(t) = 0\ (A)$, aplique um degrau de amplitude igual a $10\,V$ com início em $t = 0$s. Em seguida, encontre o erro de estado permanente e o valor do tempo de acomodação (t_s) da resposta.

2. Suponha, agora, que tenha havido um erro de 10% na identificação de K_a, isto é $K_{a_{\text{real}}} = 0,9K_a$. Ainda com $i_g(t) = 0\ (A)$, aplique um degrau de amplitude igual a $10\ V$ com início em $t = 0$s e encontre o erro de estado permanente.

3. Retorne K_a ao valor obtido na identificação e, em seguida, aplique simultaneamente as entradas $v_{t_r}(t)$ e $i_g(t)$, ambas degraus, com amplitudes respectivamente iguais a $10\ V$ e $0,450\,A$ e inícios em $t = 0$ e $t = 1\,s$ (s). Realizada a simulação, encontre o erro de regime permanente.

4.2.2 Sistema de controle com realimentação

Você deve ter observado que nas simulações 2 e 3 realizadas na seção anterior, o erro de regime permanente foi diferente de zero, isto é, o sistema compensado em malha aberta não foi capaz de rejeitar um sinal externo de perturbação e foi, também, sensível a erros de identificação. A realimentação surge, então, como a única alternativa para superar essas limitações do controlador em malha aberta. Conforme visto no exemplo 4.1, para que o sistema realimentado seja capaz de rastrear assintoticamente um degrau e rejeitar, também assintoticamente, sinais externos de perturbação do tipo degrau, o controlador deve ser dinâmico e, mais

crucial ainda, ter um pólo em $s = 0$. Portanto, a forma mais simples para esse controlador é o tipo integral (I), sendo dado por:

$$K(s) = \frac{K_I}{s} \tag{4.36}$$

onde K_I deve ser calculado de tal forma que o sistema realimentado seja estável com desempenho transitório satisfatório, isto é, que satisfaça as especificações de desempenho. Lembre-se de que, no presente caso, requer-se que o tempo de acomodação da resposta ao degrau para o sistema realimentado seja aproximadamente igual ao do sistema em malha aberta.

Para verificar os benefícios da introdução da realimentação acrescida de uma compensação dinâmica, vamos inicialmente considerar os seguintes casos: *(i)* sistema realimentado criticamente amortecido e *(ii)* sistema realimentado subamortecido com percentual de ultrapassagem menor ou igual a 5%. Para tanto, proceda da seguinte forma:

1. Calcule o valor de K_I de tal forma que o sistema realimentado seja criticamente amortecido.

2. Com o valor de K_I, calculado no item anterior, e com os valores de K, K_t, $\bar{K}$ e τ, identificados para o grupo motor-gerador, construa um modelo em SIMULINK equivalente ao diagrama de blocos da figura 4.4 e ajuste os tempos inicial e final de simulação em 0 e $2s$, respectivamente.

3. Para uma corrente $i_g(t) = 0$ (A), aplique um degrau de amplitude igual a $10\,V$ com início em $t = 0$s. Em seguida, encontre o erro de estado permanente e o valor do tempo de acomodação (t_s) da resposta.

4. Suponha, agora, que tenha havido um erro de 10% na identificação de K_a, isto é $K_{a_{\text{real}}} = 0,9K_a$. Ainda com $i_g(t) = 0$ (A), aplique um degrau de amplitude igual a $10\ V$ com início em $t = 0$s e encontre o erro de estado permanente.

5. Retorne K_a ao valor obtido na identificação e, em seguida, aplique simultaneamente as entradas $v_{t_r}(t)$ e $i_g(t)$, ambas degraus, com amplitudes respectivamente iguais a $10\ V$ e $0,450\,A$ e inícios em $t = 0$ e $t = 1\,s$ (s). Realizada a simulação, encontre o erro de regime permanente.

6. Calcule, agora, o valor de K_I de tal forma que o sistema realimentado seja subamortecido com percentual de ultrapassagem menor ou igual a 5%.

7. Repita os itens 2 a 5, acima.

O aluno deve ter observado nas simulações realizadas que o sistema realimentado com o controlador integral, em ambos os casos (criticamente amortecido e subamortecido com percentual de ultrapassagem menor ou igual que 5%), eliminou a sinal externo de perturbação e teve erro de regime permanente nulo, mesmo quando se supôs um erro de 10% no ganho da função de transferência da planta. Contudo, o tempo de acomodação da resposta ao degrau para o sistema realimentado é, para ambos os casos, o dobro do tempo de acomodação da resposta ao degrau do sistema em malha aberta. Para se justificar esse fato, construa o diagrama do lugar das raízes do sistema realimentado para $K(s) = K_I/s$ e, a partir do valor da parte real dos pólos, mostre que, de fato, o tempo de acomodação da resposta ao degrau deveria ser o dobro daquele obtido para o sistema em malha aberta.

Assim sendo, para que o desempenho do sistema realimentado seja aproximadamente igual ao do sistema em malha aberta, os pólos de malha fechada devem ter parte real aproximadamente igual a $-1/\tau$ (pólo da planta). Isto implica que o diagrama dos lugar das raízes deve se deslocar para a esquerda e, para tanto, é necessário que exista um zero igual a $-z$ ($z > 0$) à esquerda de $-1/\tau$. Portanto, $K(s)$ deve ter a seguinte forma:

$$K(s) = \frac{K_p(s+z)}{s} = K_p\left(1 + \frac{z}{s}\right) = K_p\left(1 + \frac{1}{T_i s}\right) \tag{4.37}$$

onde $T_i = 1/z$. Note, pela equação 4.37 acima, que a introdução do zero no controlador equivale a dotá-lo de ação proporcional, resultando num controlador PI.

Finalmente, para verificar se tal controlador, de fato, tornou o sistema realimentado mais rápido, proceda da seguinte forma:

1. Calcule o valor de K_p e T_i de tal sorte que os pólos do sistema realimentado tenham parte real igual a $-1/\tau$.

2. Substitua a função de transferência do bloco referente ao controlador pela equação 4.37, com os valores de K_p e T_i calculados no item anterior.

3. Para uma corrente $i_g(t) = 0\ (A)$, aplique um degrau de amplitude igual a $10\,V$ com início em $t = 0$s. Em seguida, encontre o valor do tempo de acomodação (t_s) da resposta.

4.3 Comentários finais

Uma vez obtida uma função de transferência para o controlador, o passo final é a sua implementação no sistema real. A implementação da função de transferência 4.37 pode ser feita utilizando dispositivos eletrônicos analógicos ou digitais. Neste laboratório utilizar-se-á um controlador analógico, deixando o controlador digital para um laboratório futuro. A implementação do controlador no sistema real será o assunto do próximo capítulo.

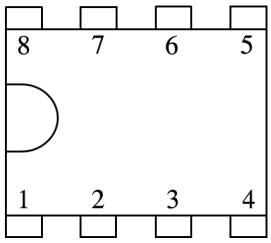
Capítulo 5

Implementação do controlador de velocidade

Neste capítulo trataremos da implementação do controlador projetado no capítulo anterior. Como se trata de um laboratório para um primeiro curso de Sistemas de Controle, será desenvolvido um controlador analógico, em cuja construção serão utilizados amplificadores operacionais[1]. Esses amplificadores, pelas suas características de resposta e preço, são freqüentemente utilizados na implementação de controladores analógicos.

Antes de entrarmos na implementação do controlador faremos um breve estudo introdutório sobre amplificadores operacionais. Neste estudo não consideraremos detalhes construtivos (que geralmente é feito nos cursos de Eletrônica), nos preocupando somente com os aspectos externos do dispositivo e com as suas características que, quando exploradas, nos permitirão construir circuitos que tenham um determinado comportamento. Isso será realizado na seçcão 5.1. Caso o aluno já possua conhecimentos suficientes desse assunto, pode omitir esse estudo e caso o aluno deseje obter maiores detalhes sobre teoria e prática de amplificadores operacionais deve consultar [4]. Dotado dos fundamentos necessários à utilização dos amplificadores operacionais, o passo seguinte será a construção de um circuito eletrônico para o controlador, o que será feito na seção 5.2.

[1]Este mesmo controlador poderia ser implementado digitalmente, isto é, utilizando-se computadores digitais.



1 - NC (não conectar)
2 - Entrada inversora
3 - Entrada não inversora
4 - Tensão de polarização ($-V_{cc}$)
5 - NC
6 - Saída
7 - Tensão de polarização($+V_{cc}$)
8 - NC

Figura 5.1: Representação esquemática de um amplificador operacional

5.1 Amplificadores Operacionais

5.1.1 Conceitos básicos

Amplificadores operacionais são amplificadores que têm elevado ganho e são geralmente utilizados para amplificar sinais que se estendem sobre uma ampla faixa de freqüências. Na figura 5.1 está representado um amplificador operacional de oito pinos, com as respectivas indicações das conecções que devem ser efetuadas em cada um dos seus terminais e, na figura 5.2, o símbolo mais comumente adotado para representá-lo em um circuito. Observe na figura 5.1 o chanfro na parte esquerda do retângulo. Esse chanfro é utilizado nos circuitos integrados encapsulados para servir como referência para a numeração dos pinos. Note, ainda na figura 5.1, a necessidade das tensões externas de polarização ($\pm V_{CC}$), que são devidas ao fato dos amplicadores operacionais serem, na verdade, circuitos integrados construídos com diversos estágios de transistores, que devem, portanto, ser polarizados.

A maioria dos amplificadores operacionais possui dois terminais de entrada e um terminal de saída, conforme representado na figura 5.2. A entrada (-) é denominada inversora enquanto a entrada (+) 'e chamada não- inversora. Isto faz com que esses amplificadores sejam conhecidos como diferenciais, uma vez a tensão de saída é proporcional à diferença entre as tensões aplicadas nos seus terminais de entrada. Utilizando-se a notação da figura 5.2, essa característica é matematicamente descrita pela seguinte equação:

$$v_0 = A(v_2 - v_1) \tag{5.1}$$

onde A é o ganho de malha aberta ($A > 45.000$), sendo definido pelos transistores

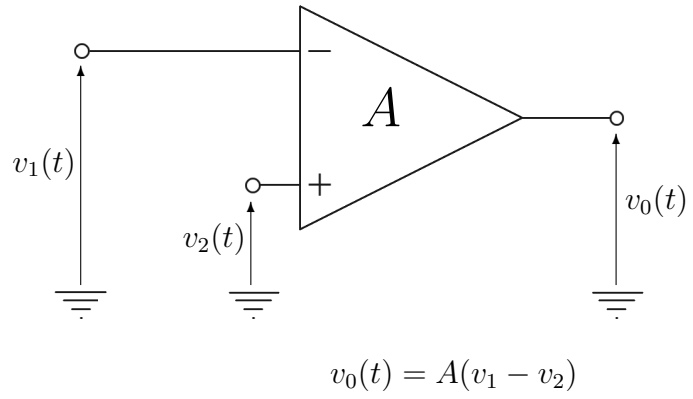


Figura 5.2: Simbologia adotada para amplificadores operacionais

internos do amplificador. Usualmente, na prática, este ganho é considerado infinito ($A \to \infty$). Outras características, geralmente adotadas na prática são: *(i)* $v_0 = 0$ quando $v_1 = v_2$ (tensão 'offset' de entrada nula); *(ii)* impedância de entrada infinita, que permite baixa ou nenhuma interferência com os elementos colocados em cascata no circuito ($Z_{in} = \infty$); *(iii)* impedância de saída nula ($Z_{out} = 0$); *(iv)* largura de faixa infinita; e *(v)* atraso de resposta nulo. Na tabela 5.1 são mostrados os valores reais de algumas dessas grandezas para os amplificadores operacionais 741 e LF356. É importante ressaltar que o valor da impedância de entrada para o LF356 'e aproximadamente $10^{12}\Omega$, isto é, praticamente infinita[2]. Em face disso, o LF356 será utilizado na implementação do controlador de velocidade do grupo motor-gerador.

5.1.2 Configurações básicas

A configuração em malha aberta da figura 5.2 é raramente utilada. Ao se construir um circuito utilizando um amplificador operacional são geralmente utilizados outros elementos de circuitos (resistores, capacitores, indutores, etc) para formar redes de realimentação. Essas configurações farão com que o novo circuito tenha características diferentes daquelas do amplificador em malha aberta, como por exemplo: modificações do ganho, da resposta em freqüência e a possibilidade de se efetuar

[2]A razão dessa elevada impedância de entrada reside no fato desse amplificador ser construído com transistores de efeito de campo.

	741	LF356
Ganho de malha aberta (V/mV)	200	200
Tensão 'offset' de entrada (mV)	2	3
Impedância de entrada $(M\Omega)$	2	10^6
Tensão de polarização (V)	± 15	± 15
Banda passante (MHz)	1	1
Corrente de saída máxima (mA)	5	5

Tabela 5.1: Características dos amplificadores operacionais 741 e LF356

operações matemáticas como soma, integração, derivação e etc. Algumas dessas configurações são apresentadas a seguir:

Circuito inversor

Considere o circuito da figura 5.3. Aplicando-se a lei das correntes de Kirchhoff ao nó 1 e lembrando que, como o amplificador operacional tem impedância de entrada infinita, a corrente que flui para a entrada inversora é nula, pode-se escrever:

$$\begin{cases} \frac{e_i(t)-e_s(t)}{R_i} + \frac{e_o(t)-e_s(t)}{R_f} = 0 \\ e_o(t) = -Ae_s(t) \ \ (A \to \infty) \end{cases} \tag{5.2}$$

Substituindo $e_s(t) = -e_o(t)/A$ na primeira equação, resulta:

$$\frac{e_i(t)}{R_i} + e_o(t)(\frac{1}{R_f} + \frac{R_i}{A} + \frac{R_f}{A}) = 0. \tag{5.3}$$

Finalmente, fazendo $A \to \infty$, obtém-se:

$$e_o(t) = -\frac{R_f}{R_i}e_i(t) \tag{5.4}$$

As seguintes observações podem ser feitas a partir da equação 5.4 acima: *(i)* o ganho não é mais determinado pelo amplificador e sim pelos elementos externos R_i e R_f; *(ii)* quando o sinal de entrada for limitado, isto é, $|e_1(t)| < E_i$, tem-se que $|e_0(t)| < (R_f/R_i)E_i = E_0$ e, portanto limitado. Como $A \to \infty$, resulta:

$$|e_s(t)| = \frac{|e_0(t)|}{A} \to 0. \tag{5.5}$$

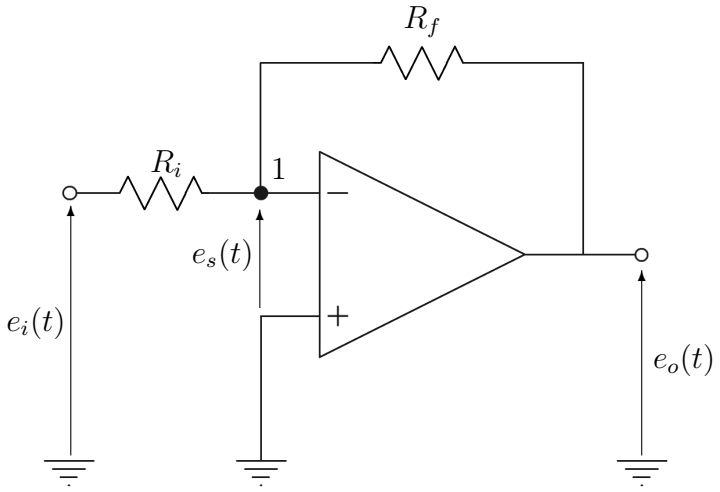


Figura 5.3: Circuito inversor

Por essa razão, os terminais referentes às entradas inversora e não-inversora são dito estarem em terra virtual, isto é, para efeitos de análises de redes contendo amplificadores, $e_s(t)$ pode ser feito identicamente nulo.

Circuito não-inversor

Considere agora o circuito da figura 5.4(a), que difere do circuito inversor da figura 5.3 principalmente pelo fato do sinal externo $e_i(t)$ estar agora conectado à entrada não-inversora. Aplicando a lei das correntes de Kirchhoff ao nó 1 e levando em conta os fatos de que a corrente que entra no amplicador operacional é nula e que os terminais + e − estão em terra virtual pode-se escrever:

$$\frac{e_i(t)}{R_i} + \frac{e_i(t) - e_o(t)}{R_f} = 0. \tag{5.6}$$

Após algumas manipulações algébricas simples, obtém-se:

$$e_o(t) = \frac{R_f + R_i}{R_i} e_i(t) \tag{5.7}$$

Note, na equação 5.7 acima que o ganho de malha fechada será sempre maior ou igual a 1 e é, mais uma vez, determinado por R_i e R_f. Observe ainda que, se R_f

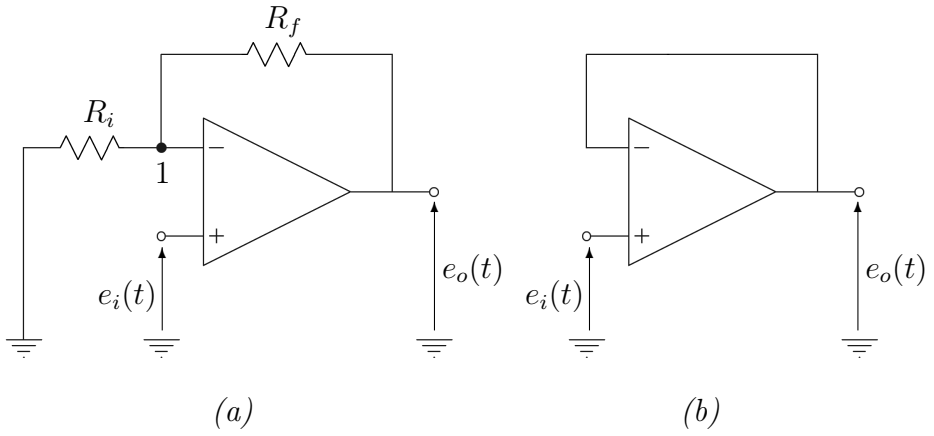


Figura 5.4: *(a)* Circuito não-inversor; *(b)* Seguidor de tensão

é feito igual a zero (curto-circuito) e $R_i = \infty$ (circuito aberto), conforme mostrado na figura 5.4(b), tem-se que o ganho será exatamente igual a 1. Nesse caso, o amplificador atua como um 'seguidor de tensão', isto é, a tensão de saída segue exatamente a tensão de entrada. A configuração de seguidor de tensão é largamente utilizada para prover isolamento entre os sinais da fonte e da carga, evitando assim interações indesejáveis.

Circuito comparador

Uma das maneiras de se fazer a comparação entre dois sinais é utilizando-se o circuito da figura 5.5. Aplicando-se a lei das correntes de Kirchhoff ao nó A e lembrando que os pontos A e B têm o mesmo potencial (terra virtual), pode-se escrever:

$$\begin{cases} e_B(t) = e_A(t) = \frac{R_3}{R_2+R_3} e_2(t) \\ \frac{e_1(t)-e_B(t)}{R_1} + \frac{e_o(t)-e_B(t)}{R_f} = 0. \end{cases} \tag{5.8}$$

Após algumas manipulações algébricas elementares, obtém-se:

$$e_o(t) = -\frac{R_f(R_2+R_3)}{R_1(R_2+R_3)} e_1(t) + \frac{R_3(R_1+R_f)}{R_1(R_2+R_3)} e_2(t). \tag{5.9}$$

Finalmente, fazendo $R_1 = R_2 = R_3 = R_f = R$ na equação 5.9 acima, resulta:

$$e_o(t) = e_2(t) - e_1(t). \tag{5.10}$$

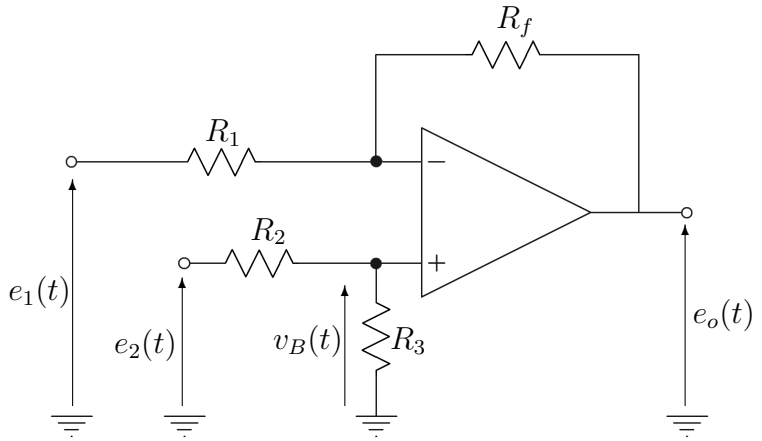


Figura 5.5: Circuito comparador genérico

Deve ser ressaltado que, na prática, é quase impossível ter quatro resistores de valores exatamente iguais e, portanto, a igualdade (5.10) não pode, a princípio, ser implementada fisicamente. Este problema pode ser superado inserindo-se potenciômetros de valores R'_1, R'_2, R'_3 e R'_f em série com os resistores R_1, R_2, R_3 e R_f. Esses potenciômetros devem ser ajustados de tal sorte que $R_1+R'_1 = R_2+R'_2 = R_3+R'_3 = R_f + R'_f$. Para se verificar de que a igualdade (5.10) está sendo verificada (a menos de um 'offset'), aplica-se sinais iguais em $e_1(t)$ e $e_2(t)$ e mede-se o sinal $e_o(t)$, que nesse caso deve ser aproximadamente igual a zero (a menos de um 'offset').

Circuito integrador

O circuito da figura 5.6(a) representa uma maneira de se calcular analogicamente a integral de um sinal. Pode-se mostrar que a função de transferência do circuito é dada por:

$$\frac{E_o(s)}{E_i(s)} = -\frac{1}{R_i C_f s} \tag{5.11}$$

e, portanto,

$$e_o(t) = -\frac{1}{R_i C_f}\int_0^t e_i(\lambda)d\lambda. \tag{5.12}$$

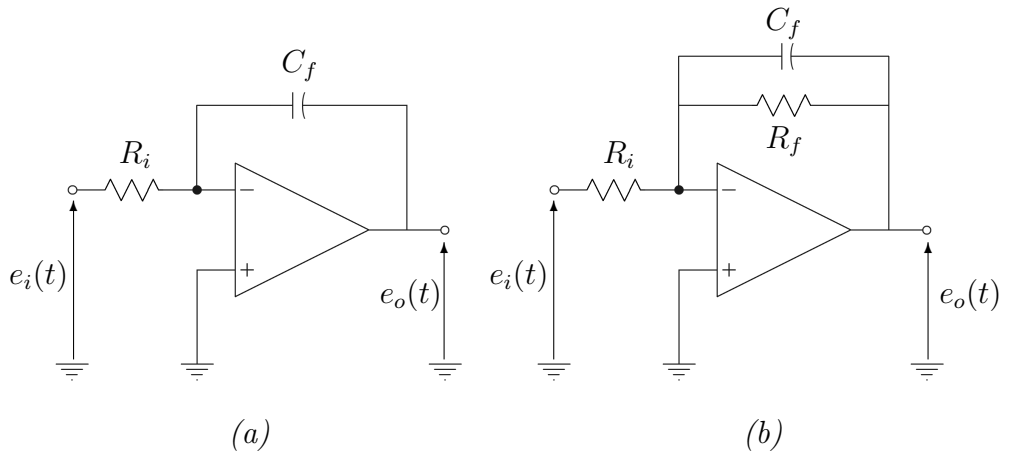


Figura 5.6: Circuitos integradores: *(a)* ideal; *(b)* prático.

A realimentação composta unicamente por um capacitor torna a configuração da figura 5.6(a) bastante sensível para alguns amplificadores operacionais (inclusive o 741), o que torna a sua implementação problemática. Para solucionar este problema, introduz-se um resistor em paralelo com o capacitor, conforme representado na figura 5.6(b). Neste caso, a função de transferência do sistema passa a ser a seguinte:

$$\frac{E_o(s)}{E_i(s)} = -\frac{R_f}{R_i}\frac{1}{R_f C_f s + 1} \tag{5.13}$$

cujas assíntotas do diagrama de módulo de Bode estão representadas na figura 5.7. Note que aumentando-se o valor do resistor R_f, o pólo $-1/R_f C_f$ se aproxima de zero ao mesmo tempo em que o ganho DC torna-se cada vez maior. Nestas condições, a resposta em freqüência do sistema assemelha-se cada vez mais à de um integrador ideal e, portanto, o circuito da figura 5.6(b) pode ser usado com razoável precisão para se efetuar a integração de um sinal.

Circuito derivador

A derivação de um sinal analógico pode ser feita idealmente utilizando-se o circuito da figura 5.8(a). Para esse circuito, tem-se que:

$$e_o(t) = -R_f C_i \frac{d}{dt} e_i(t). \tag{5.14}$$

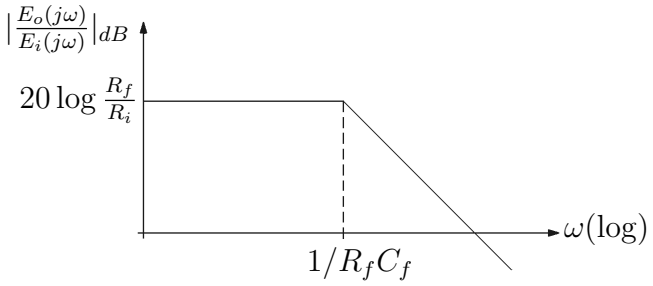
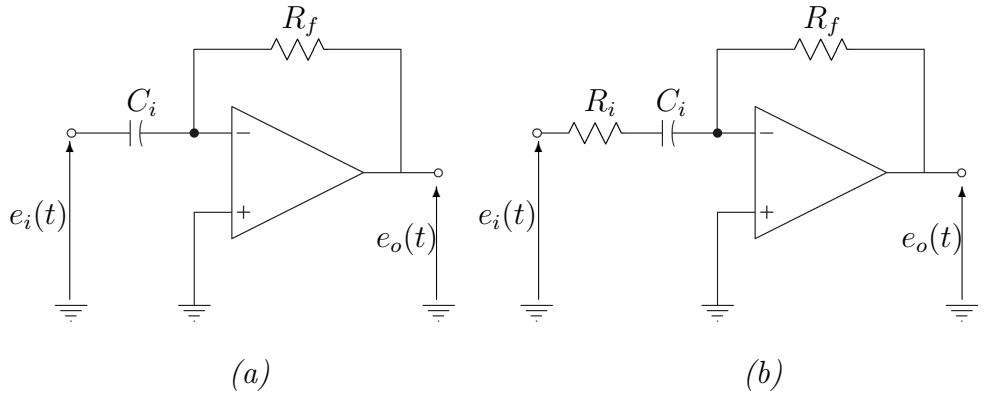


Figura 5.7: Assíntotas do diagrama de módulo de Bode para o circuito integrador prático da figura 5.6(b)



(a) *(b)*

Figura 5.8: Circuitos derivadores: *(a)* ideal; *(b)* prático

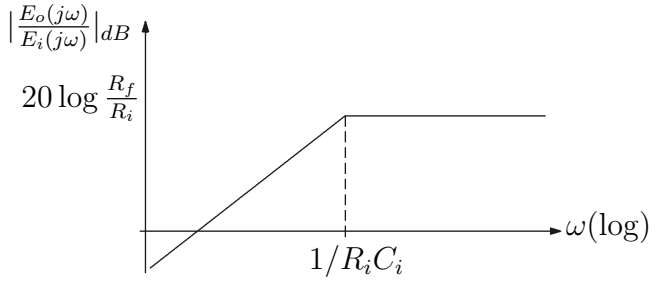


Figura 5.9: Assíntotas do diagrama de módulo de Bode para o circuito derivador prático da figura 5.8(b)

A utilização desse circuito como um derivador pode trazer sérios problemas no que se refere a sinais com componentes de freqüências muito altas, uma vez que tais sinais seriam amplificados por valores infinitos (pelo menos teoricamente). Para solucionar este problema, faz-se uso do circuito da figura 5.8(b), cuja função de transferência é dada por:

$$\frac{E_o(s)}{E_i(s)} = -\frac{sR_fC_i}{R_iC_is+1}, \tag{5.15}$$

cujas assíntotas do diagrama de módulo de Bode estão representadas na figura 5.9. Note que, diminuindo R_i, o pólo $-1/R_iC_i$ se afasta cada vez mais de zero, fazendo com que o ganho de alta freqüência aumente, o que torna a resposta em freqüência desse sistema mais próxima daquela de um derivador ideal. Assim sendo, o circuito da figura 5.8(b) pode ser utilizado para se efetuar a derivação de um sinal.

Um circuito para controladores PID

Um circuito cuja saída $e_o(t)$ é proporcional à entrada $e_i(t)$, à sua integral e à sua derivada, isto é,

$$e_o(t) = K_p\left(e_i(t) + \frac{1}{T_i}\int_0^t e_i(\lambda)d\lambda + T_d\frac{d}{dt}e_i(t)\right) \tag{5.16}$$

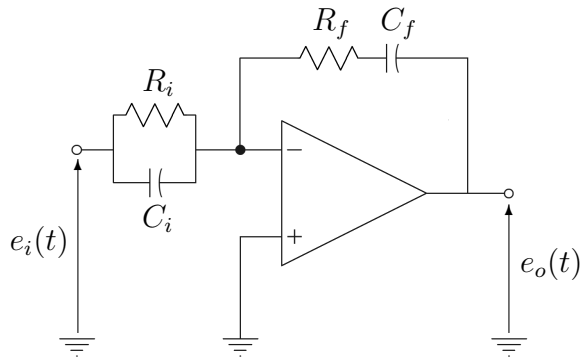


Figura 5.10: Circuito para controladores PID

está representado na figura 5.10. Pode-se mostrar que

$$\begin{aligned} K_p &= \frac{R_f}{R_i} \\ T_d &= R_i C_i \\ T_i &= R_f C_f. \end{aligned} \tag{5.17}$$

para $T_i >> T_d$.

Em geral, os valores de K_p, T_i e T_d são determinados durante a fase de projeto do controlador. O problema passa, então, a ser o de encontrar R_i, R_f, C_i e C_f de tal forma que o circuito do controlador PID tenha valores de K_p, T_i e T_d o mais próximo possível dos valores calculados teoricamente. Para tanto, deve-se resolver o sistema de três equações e quatro incógnitas (5.17). O número maior de incógnitas, faz com que a escolha dos valores das capacitâncias e resistências não seja única, o que é bom do ponto de vista prático, uma vez que pode-se 'jogar' com os capacitores e resistores disponíveis no laboratório.

5.2 Controlador automático de velocidade

Após a breve introdução sobre amplificadores operacionais apresentada na seção anterior, estamos em condições de construir um circuito eletrônico analógio para o controlador de velocidade do grupo motor-gerador. O diagrama de blocos para o

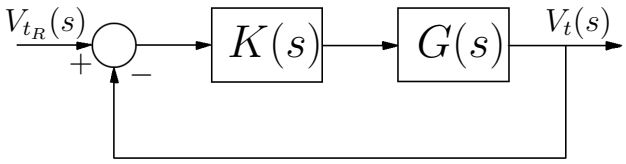


Figura 5.11: Diagrama de blocos correspondente ao sistema de controle do grupo motor-gerador

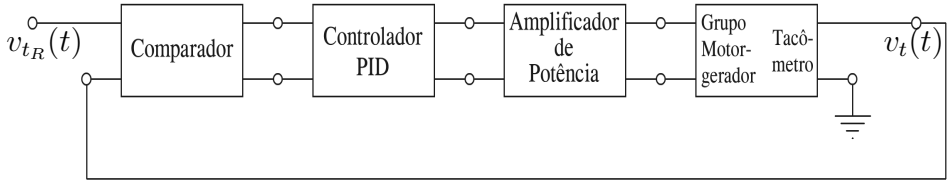


Figura 5.12: Representação esquemática do sistema de controle de velocidade do grupo motor-gerador

sistema está representado na figura 5.11, onde $G(s)$ representa a função de transferência da planta e

$$K(s) = K_p \frac{s+z}{s} = K_p \left(1 + \frac{1}{T_i s}\right) \tag{5.18}$$

onde os valores de K_p e T_i foram determinados ao final do capítulo 4.

A partir do diagrama da figura 5.11, vê-se que a implementação do sistema de controle requer a construção de um circuito comparador e de um circuito para o controlador PID. Em princípio, nenhum outro elemento seria necessário e bastaria o engenheiro de controle conectar a saída do controlador ao motor. Porém, amplificadores operacionais têm corrente de saída muito baixa (aproximadamente $5mA$, conforme mostrado na tabela 5.1). Isto faz com que seja necessário acoplar na saída do amplificador operacional um amplificador de potência para poder 'drenar' a corrente necessária para o motor, o que faz com que a implementação do sistema de controle de velocidade para o grupo motor-gerador seja feita conforme mostrado no diagrama de blocos da figura 5.12

5.2.1 Implementação do comparador

O circuito utilizado para se efetuar a comparação entre o sinal de referência ($v_{t_R}(t)$) e o sinal do tacômetro ($v_t(t)$) será aquele representado na figura 5.5. Para sua implementação, proceda da seguinte forma:

1. Selecione quatro resistores de mesmo valor (de acordo com seu código de barras coloridas) com valores entre $1k\Omega$ e $10k\Omega$. Valores de resistores mais baixo que os sugeridos podem levar o amplificador a não executar exatamente a operação que dele esperamos.

2. Meça a resistência de cada um dos resistores R_1, R_2, R_3 e R_4 e anote-as.

3. Ajuste os quatro potenciômetros R'_1, R'_2, R'_3 e R'_4 de tal modo que $R_1 + R'_1 = R_2 + R'_2 = R_3 + R'_3 = R_f + R'_f$.

4. Monte o circuito da figura 5.5 utilizando um amplificador LF356, efetuando-se as conecções de seus terminais de acordo com a figura 5.1. Lembre-se de que o amplicador deve ser polarizado com as tensões de $\pm 15V$ (ver tabela 5.1).

5. Para verificar se o circuito somador está, de fato, efetuando a subtração dos sinais, aplique sinais iguais a ambos os terminais. Isto pode ser feito, por exemplo, conectando-se ambas as entradas ao ponto de potencial zero (terra). Neste caso, de acordo com a tabela 5.1 a tensão 'offset' de saída deverá ser de, aproximadamente, $2mV$. Caso a tensão seja muito maior que $2mV$, varie suavimente cada um potenciômetro, observando se tal movimento implica numa diminuição da tensão de saída. Termine este procedimento quando a tensão de saída se aproximar de $2mV$ ou quando qualquer variação no potenciômetro resultar num aumento da tensão de 'offset'.

6. Finalmente, como $e_o(t) = e_2(t) - e_1(t)$ (equação 5.10), faça $e_1(t) = v_t(t)$ (a tensão de saída do tacômetro) e $e_2(t) = v_{t_R}(t)$ (a tensão de referência).

5.2.2 Implementação do controlador

O circuito da figura 5.10 permite a implementação de um controlador PID. Porém, o controlador de velocidade desenvolvido ao final do capítulo 4 é do tipo PI. Assim sendo, caso o circuito da figura 5.10 seja usado para a implementação do controlador

da equação 5.18, o ganho derivativo deve ser feito igual a zero. Com isso em mente, proceda da seguinte forma para a implementação do controlador:

1. Faça, inicialmente, $C_i = 0F$. Em seguida, utilizando um resistor R_f de aproximadamente $10k\Omega$, encontre um capacitor C_f e um resistor R_i, de tal forma que, a partir das equações 5.17, sejam obtidos valores de K_p e T_i aproximadamente iguais aos encontrados no final do capítulo 4.

2. Utilizando um amplificador LF356 monte o circuito da figura 5.10.

3. Para verificar se o controlador PI foi, de fato, implementado corretamente, aplique ao controlador um sinal de onda quadrada de freqüência igual a $0,3Hz$, amplitude igual a $1V$ e valor médio igual a zero. Como o circuito tem ações proporcional e integral, o sinal de saída deve ser uma rampa com inclinação definida pela constante de tempo de integração, T_i, e deslocada verticalmente, sendo este deslocamente determinado pelo ganho proporcional.

5.2.3 Amplificador de potência

O amplificador de potência utilizado no laboratório é também inversor e tem um ganho igual a cinco. Portanto, o aluno deve tomar o cuidado de certificar-se de que o ganho em malha aberta do sistema real (somador + PI + amplificador de potência) seja igual ao projetado e as inversões levem a uma realimentação negativa.

5.3 Experimento final: funcionamento do sistema de controle

Implementados o comparador e o controlador PI e tendo sido feito o ajuste para que o amplificador de potência tenha ganho igual a -1, estamos em condições de verificar na prática se o sistema de controle projetado irá funcionar satisfatoriamente. Para tanto proceda da seguinte forma:

1. Conecte o terminal de saída do comparador ao ponto correspondente à entrada do controlador PI e a saída do controlador PI à entrada do amplificador de potência. Conecte ainda a saída do tacômetro à entrada correspondente do circuito comparador.

2. Aplique um sinal de onda quadrada de freqüência igual a $0.3Hz$, amplitude igual a $9V$ e tensão mínima igual a zero ao terminal do comparador correspondente à tensão de referência.

3. Conecte a saída do amplificador de potência aos terminais do motor e observe no osciloscópio os sinais de referência ($v_{t_R}(t)$) e da saída do tacômetro ($v_t(t)$). Responda às seguintes questões:
   (i) Houve rastreamento do sinal de referência?
   (ii) Em caso afirmativo, qual o tempo de acomodação aproximado da resposta?
   (iii) O tempo de acomodação para o sistema real está próximo daquele obtido durante a simulação realizada no capítulo 4?

4. Conecte agora uma carga nos terminais do gerador e feche este circuito num instante próximo do meio do pulso. Verifique se o sistema rejeita esta perturbação assim como na simulação realizada no capítulo 4.

5.4 Comentários finais

Este laboratório procurou abordar do ponto de vista prático, os principais conceitos introduzidos na disciplina Sistemas de Controle I. Espera-se que os alunos tenham verificado que o ferramental teórico pode ser aplicado, efetivamente, na prática, para levar o sistema real a ter um comportamento bastante próximo daquele previsto durante a simulação. Outro aspecto a ser ressaltado, ao fim desse curso, é que as ferramentas utilizadas na identificação da função de transferência do grupo motor-gerador e no projeto do controlador de velocidade não se aplicam somente ao sistema utilizado nesse laboratório, podendo ser utilizadas em uma grande variedade de outros sistemas físicos e devem se usadas no futuro quando ao aluno for solicitado o projeto de um sistema de controle.

Referências Bibliográficas

[1] N. A. Kheir, K. J. Aström, D. Auslander, K. C. Cheok, G. F. Franklin, M. Masten, and M. Rabins, "Control systems engineering education," *Automatica*, vol. 32, pp. 147–166, 1991.

[2] R. M. Stephan, *Laboratório de Controle I.* UFRJ - Escola de Engenharia, 1991.

[3] MathWorks, *Matlab Prime R2012a.* The Math Works Inc., 2012.

[4] J. G. Graeme, G. E. Tobei, and L. P. Huelsman, *Operational Amplifiers: Design and Applications.* McGraw-Hill Book Company, 1981.