

INTRODUÇÃO AOS SISTEMAS DE CONTROLE (LABORATÓRIO)

Professores: Silas do Amaral
Celso José Faria de Araújo

Joinville, fevereiro de 2020.

Sumário

1 Introdução ao Matlab	1
1.1 Introdução	1
1.2 Fundamentos do Matlab	1
1.3 Descrição de Comandos Básicos.....	1
1.3.1 Operações Matemáticas Básicas.....	1
1.3.2 Polinômios, Vetores e Matrizes.....	2
1.3.3 Variáveis e Funções Predefinidas.....	4
1.4 Gráficos no Matlab.....	4
1.4.1 Criando Gráficos	4
1.4.2 Melhorando a Apresentação dos Gráficos.....	5
1.4.4 Imprimindo e Copiando Gráficos	6
1.5 Usando o Help.....	7
1.6 Salvando e Recuperando Variáveis	7
1.7 Programando no MATLAB – Uma introdução	8
1.8 Definindo os Caminhos de Procura de Arquivos do Matlab	9
1.9 Tarefa	10
1.10 Exercícios Sugeridos	11
2 Introdução ao Simulink	12
2.1 Introdução	12
2.2 Fundamentos do Simulink.....	12
2.2.1 Descrição dos Grupos Básicos do Simulink	13
2.2.2 Descrição das Opções do Menu da Janela do Modelo.....	14
2.2.3 Modificando os Parâmetros da Simulação	16
2.3 Exercícios	18
3 Linearização de Sistemas Não-lineares	19
3.1 Introdução	19
3.2 Sistema de Nível de Líquido	19
3.3 Parte Teórica (Fazer antes da realização da experiência)	20
3.4 Parte Experimental	20
4 Análise da Resposta Transitória de Sistemas de 1^a Ordem.....	22
4.1 Introdução	22
4.2 Exercícios	23
5 Análise da Resposta Transitória de Sistemas de 2^a Ordem	25
5.1 Introdução	25
5.2 Exercícios	25
6 Análise da Resposta Transitória de Sistemas de 2^a Ordem e Superior	27
6.1 Introdução	27
6.2 Exercícios	27
7 Análise da Resposta em Freqüência Através dos Diagramas de Bode	31
7.1 Introdução	31
7.2 Traçando Diagramas de Bode no Matlab	31
7.3 Exercícios	31

1 INTRODUÇÃO AO MATLAB

1.1 Introdução

Ao se projetar equipamentos eletrônicos, máquinas, processos químicos, sistemas de controle, etc., é fundamental avaliar o comportamento dinâmico do sistema. No caso de sistemas simples, muitas vezes a experiência do projetista é suficiente para tomar boas decisões durante o projeto, as quais influenciarão o desempenho e a operação do sistema. Entretanto, quando se trata de um sistema complexo, deve-se lançar mão de “ferramentas” que permitam antecipar problemas e auxiliar nas decisões.

O Matlab (*Matrix Laboratory - MATLAB®*) é um *software* de simulação digital importante para análise e projeto de sistemas de controle. O objetivo principal desta experiência consiste em familiarizar o aluno com algumas de suas funções básicas, úteis para a análise de sistemas lineares e que serão usadas nas experiências subsequentes.

1.2 Fundamentos do Matlab

O Matlab é um poderoso programa interativo voltado à solução de problemas científicos e de engenharia. Como o nome já diz, o Matlab é um *software* específico para manipulação e processamento de matrizes. Assim, a matriz é o elemento básico de armazenamento de informações.

Além do módulo principal, o Matlab possui módulos auxiliares (*toolboxes*) voltados para diversas áreas, tais como: Telecomunicações, Eletrônica Digital, Eletrônica Analógica, Sistemas Não Lineares, Processamento de Sinais, etc. Novas rotinas e até mesmo *toolboxes* podem ser criadas pelo usuário para resolver os seus problemas específicos.

No decorrer das explicações, são apresentados exemplos que o aluno deve executar. Para facilitar sua identificação e compreensão, estão colocados dentro de molduras (caixas) e cada comando é acrescido de um comentário, explicando sua função.

1.3 Descrição de Comandos Básicos

1.3.1 Operações Matemáticas Básicas

O Matlab segue regras semelhantes às da matemática formal para as expressões. Por exemplo,

```
>> a=3/4  
a = 0.75
```

Caso não se queira mostrar o valor da resposta, o caractere ‘ ; ’ deve ser incluído ao final da expressão. É permitido, também, inserir comentário, digitando antes do texto o caracter ‘ % ’.

```
>> b=2           % sem o caractere ';' no final da sentença o resultado é apresentado.
      b = 2
>> c=3;          % com o caractere ';' no final da sentença o resultado não é apresentado.
>> d=b+c        % o resultado é armazenado na variável 'd' e é apresentado.
      d = 5
>> b+c          % se nenhum nome é atribuído a uma variável, ela é armazenada em "ans".
ans = 5
```

Os nomes das variáveis no Matlab devem começar sempre com uma letra, que pode ser seguida de letras e/ou números. *Maiúsculas* e *minúsculas* são consideradas diferentes. Portanto, uma variável definida com *letra minúscula* deve ser referenciada sempre desta forma, para ser reconhecida pelo Matlab.

As operações matemáticas no Matlab são realizadas da esquerda para direita na seguinte ordem:

^	potenciação			
*	multiplicação	e	/	divisão
+	adição	e	-	subtração

Os parênteses podem afetar a ordem das operações.

```
>> 1+2^3/4*2          % 1+{[(2^3)/4]*2}
      ans = 5
>> 1+2^3/(4*2)        % 1+[(2^3)/(4*2)]
      ans = 2
>> (1+2)^3/(4*2)      % [(1+2)^3]/(4*2)
      ans = 3.3750
```

1.3.2 Polinômios, Vetores e Matrizes

Um polinômio pode ser formado, digitando-se seus coeficientes, conforme mostrado abaixo. Normalmente, utilizam-se letras maiúsculas para identificar polinômios, vetores e matrizes.

```
>> P=[1 6 8];          % declaração do polinômio P = s2 + 6s + 8
```

Para obter as raízes de um polinômio qualquer, procede-se da seguinte forma:

```
>> R=roots(P)
      R = -4
            -2
```

A operação inversa também é possível, ou seja, a partir das raízes de um polinômio, pode-se determinar seus coeficientes. Para tanto, utiliza-se o comando ***poly***, como segue.

```
>> S=poly([-2 -4])      % raízes do polinômio
      S = 1   6   8
```

Para multiplicar dois ou mais polinômios, usa-se o comando ***conv***, conforme abaixo:

```
>> P1=[1 4 8];          % polinômio P1 = s2 + 4s + 8
>> P2=[1 2];            % polinômio P2 = s + 2
>> PM=conv(P1,P2)       % PM = (s2 + 4s + 8)*(s + 2)
      PM = 1   6   16   16  % PM = s3 + 6s2 + 16s + 16
```

Para dividir dois polinômios, é utilizado o comando ***deconv***. Esta operação nem sempre é exata, podendo existir um resto; por isso, indica-se usá-la no seguinte formato:

```
>> [Q, R]=deconv(P1, P2)      % equivale a operação (s^2 + 4s + 8)/(s + 2)
    Q = 1   2
    R = 0   0   4               % Q = s + 2
                                % R = 4
```

A variável *Q* armazena o quociente da divisão, ao passo que o resto é armazenado na variável *R*.

O comando **[r, p, k] = residue(num, den)** encontra os resíduos (*r*), os pólos (*p*) e o valor direto (*k*) associados à razão entre dois polinômios. O uso deste comando é ilustrado a seguir na obtenção da expansão em frações parciais da razão de polinômios abaixo:

$$\frac{N(s)}{D(s)} = \frac{3s^3 + 9s^2 - 11s - 57}{s^3 + s^2 - 9s - 9}$$

Inicialmente, são definidos os polinômios do numerador (*N*) e do denominador (*D*); em seguida, é chamada a função **residue**, obtendo-se os seus resíduos, pólos e termo direto, como ilustra o próximo quadro.

```
>> N=[3 9 -11 -57];          % polinômio N = 3s^3 + 9s^2 - 11s - 57
>> D=[1 1 -9 -9];          % polinômio D = s^3 + s^2 - 9s - 9
>> [r, p, k]=residue(N, D)  % Encontra os resíduos da razão polinomial N(s)/D(s)
r
    -2
    3
    5               % os resíduos são -2, 3 e 5
p =
    -3
    3
    -1             % O polo -3 corresponde à constante -2, o polo 3 ao resíduo 3 e o -1 ao 5
k =
    3               % valor direto da expansão em frações parciais
```

Com isso, é facilmente obtida a expansão em frações parciais da referida razão de polinômios:

$$\frac{3s^3 + 9s^2 - 11s - 57}{s^3 + s^2 - 9s - 9} = \frac{-2}{s+3} + \frac{3}{s-3} + \frac{5}{s+1} + 3$$

O comando **[num,den]=residue(r,p,k)** converte a expressão em frações parciais de volta para a razão de polinômios num/den. No exemplo, obtém-se de volta *N*(*s*)/*D*(*s*).

Um vetor é declarado da mesma forma que um polinômio, isto é, especificando seus componentes dentro de colchetes, conforme mostrado abaixo.

```
>> A=[1 6 8]                  % declaração do vetor A = [ 1 6 8]
    A = 1   6   8
```

Alternativamente, um vetor pode ser criado usando o operador ":" , na forma:

```
>> B=0:1:5                   % declaração do vetor contendo elementos de 0 à 5 com intervalos de 1 em 1
    B = 0   1   2   3   4   5
```

A declaração de uma matriz é feita de modo semelhante, isto é:

```
>> C=[1 2; 4 8]              % declaração da matriz C = [ 1 2 ]
    C = 1   2
        4   8
```

As propriedades matemáticas de cada operação envolvendo polinômios, vetores ou matrizes são válidas no Matlab.

1.3.3 Variáveis e Funções Predefinidas

O Matlab possui variáveis predefinidas, apresentadas a seguir:

- i e $j = \sqrt{-1}$ (se usar i e j como variáveis, limpar após o uso com o comando `clear`)
- $\pi = \pi$ (3,1416)
- $Inf = \infty$
- $Nan = \text{não número}$ (ex.: $0/0$)

```
>> 2*pi
    ans = 6.2832
>> d=4/Inf
    d = 0
>> z=2+2i
    z = 2.0000 + 2.0000i
```

Além disso, diversas funções predefinidas, tais como as funções trigonométricas, as funções exponencial, raiz quadrada, etc, estão implementadas no MatLab. Alguns exemplos são fornecidos no próximo quadro.

```
>> u=sin(3*pi/4)          % o Matlab usa como default a medida de ângulo em radianos
    u = 0.7071
>> v=sqrt(4)              % o comando sqrt (square root) executa a operação raiz quadrada
    v = 2
>> abs(z)                 % valor absoluto (módulo) da variável z definida anteriormente
    ans = 2.8284
>> angle(z)               % ângulo (fase) do número variável z definida anteriormente
    ans = 0.7854
>> exp(-1)                 % exponencial
    ans = 0.3679
>> log10(100)              % logaritmo na base 10
    ans = 2
```

1.4 Gráficos no Matlab

O Matlab é um *software* capaz de traçar vários tipos de gráficos, sejam eles em duas ou mais dimensões, em escalas lineares, logarítmicas ou semilogarítmicas. Nesta experiência, serão estudados apenas os gráficos bidimensionais com escalas lineares.

1.4.1 Criando Gráficos

Para criar um gráfico linear da forma (x,y) , onde x é uma variável e y é uma função, ou seja, $y=f(x)$, digita-se o comando `plot(x,y)`. É importante, no entanto, fornecer uma faixa de variação da variável x , o que deve ser feito conforme mostrado abaixo.

```
>> x=-15:0.05:15;           % x assumirá valores entre -15 e +15 com intervalos de 0,05
>> y=sin(x);
>> plot(x,y)
```

No caso de se desejar mais curvas em um mesmo gráfico, pode-se usar o comando **`hold`** (ou **`hold on`**) de forma a fazer com que a escala do próximo gráfico se adapte à escala do gráfico atual. Continuando o exemplo anterior:

```
>> y2=cos(x);
>> hold                  % pode-se utilizar também o comando hold on
current plot held         % gráfico corrente "travado"
>> plot(x,y2);
```

Observe que todas as curvas são traçadas com linha contínua azul, pois este é o estilo de linha padrão. Note que a semelhança entre as curvas pode ser um problema para sua identificação.

Digitando **hold** novamente (ou **hold off**), o gráfico é liberado, dando lugar a um novo.

```
>> hold
current plot released % gráfico corrente "liberado"
```

Outra forma de traçar mais de uma curva no mesmo gráfico é mostrada a seguir.

```
>> y3 = sin(2*x);
>> plot(x,y,x,y2,x,y3);
```

Observe que, deste modo, as curvas são traçadas em cores distintas, facilitando sua identificação. A ordem das cores apresentadas é a seguinte: *azul, verde, vermelho, ciano, magenta, amarelo e preto*.

1.4.2 Melhorando a Apresentação dos Gráficos

No entanto, se a impressora utilizada for monocromática, é importante que as curvas sejam traçadas em estilos diferentes, especificando o *tipo de linha* e, se necessário, utilizando *símbolos* para marcar cada valor calculado pelo Matlab, de acordo com a convenção estabelecida na Tabela 1.1.

Tabela 1.1 – Especificações extras para o traçado de gráficos

Tipos de Linha	Símbolos	Cores de Linha
- contínua	o com círculos	y amarelo (yellow)
-- tracejada	.	m magenta (magenta)
: pontilhada	x com caracteres x	c ciano (cyan)
-. traço e ponto	+ com caracteres +	r vermelho (red)
	* com caracteres *	g verde (green)
	s quadrado	b azul (blue)
	d diamante	w branco (white)
	v triângulo	k preto (black)
	p estrela com cinco arestas	
	h estrela com seis arestas	

Estas melhorias e outras mais são conseguidas de uma forma mais fácil, acessando os menus na própria janela do gráfico (Ambiente *Figure*). Por exemplo, ativando o modo **Edit Plot** no menu **Tools** e clicando duas vezes, rapidamente, sobre a curva, abre-se uma janela, que permite ajustar diversos parâmetros como os exibidos na tabela acima.

A forma mais adequada para traçar gráficos com mais de uma curva é a seguinte:

```
>> plot(x,y,'-r') % o gráfico y será traçado com linha contínua vermelha.
>> hold on % retém a execução do gráfico.
>> plot(x,y2,'--b') % o gráfico y2 será traçado com linha tracejada azul.
```

Ou então, usando apenas um comando, conforme exemplificado abaixo:

```
>> plot(x,y,'-r',x,y2,'--b') % o gráfico obtido será idêntico ao anterior.
```

Para adicionar linhas de grade no gráfico, dar nome aos eixos, além de inserir título no gráfico e mudar o valor das escalas dos eixos, são usados os seguintes comandos:

grid	insere linhas de grade no gráfico
xlabel 'Nome do eixo x';	insere o nome do eixo x
ylabel 'Nome do eixo y';	insere o nome do eixo y
title 'título do gráfico';	insere o título do gráfico

<pre>axis([xmin,xmax,ymin,ymax]); text(posição no eixo x, posição no eixo y,'Texto'); legend('Texto 1', 'Texto 2', n);</pre>	muda as escalas dos eixos x e y insere um texto dentro do gráfico insere a legenda no gráfico
--	---

Dica: Estes comandos são facilmente acionados através do menu **Insert**, que faz parte do ambiente **Figure**.

O comando **text** possui o inconveniente de, uma vez colocado o texto, não poder ser movido com facilidade. Existe, porém, um comando **gtext** com o qual a escolha da localização do texto é feita com auxílio do mouse. Seguem alguns exemplos:

<pre>>> grid >> xlabel 'x' >> ylabel 'y = f(x)' >> title 'Exercício' >> axis([-10,10,-1.1,1.1]) >> legend('sen(x)', 'cos (x)')</pre>	% insere linhas de grade no gráfico % insere o nome do eixo x % insere o nome do eixo y % insere o título do gráfico % muda as escalas dos eixos x e y % insere legenda no gráfico
--	---

Obs.: A legenda pode ser movida com o auxílio do mouse, arrastando-a para a posição desejada.

1.4.4 Imprimindo e Copiando Gráficos

A impressão dos diagramas pode ser feita de duas maneiras: impressão direta do gráfico para a impressora, ou transportando o gráfico para um editor de texto (Word, LaTeX, etc.)

Para imprimir diretamente o gráfico, clique no menu "**File**" – "**Print**". Desta forma, o gráfico será impresso usando toda a página. Caso a impressão não esteja da forma desejada, deve-se configurar a impressora no menu "**File**" – "**Print Setup**" e a página no menu "**File**" – "**Page Setup**".

Para copiar um gráfico para um editor de textos, escolhe-se antes a opção de cópia. Esta escolha é feita no menu "**Edit**" – "**Copy Options**", como mostrado na *Figura 1.1*. Nesta janela, escolhe-se inicialmente o tipo de figura: *Metafile* ou *Bitmap*.

Cada uma das opções tem suas vantagens e desvantagens, sendo que a principal vantagem de uma figura tipo *Metafile* é que ela ocupa menos espaço que uma do tipo *Bitmap* (quase a metade). Por outro lado, uma figura do tipo *Metafile* pode ser distorcida quando, após colada no editor de textos, seu tamanho for modificado somente num dos sentidos (horizontal ou vertical). Já uma figura do tipo *Bitmap* preserva suas características mesmo quando suas dimensões são completamente alteradas dentro do editor de textos.

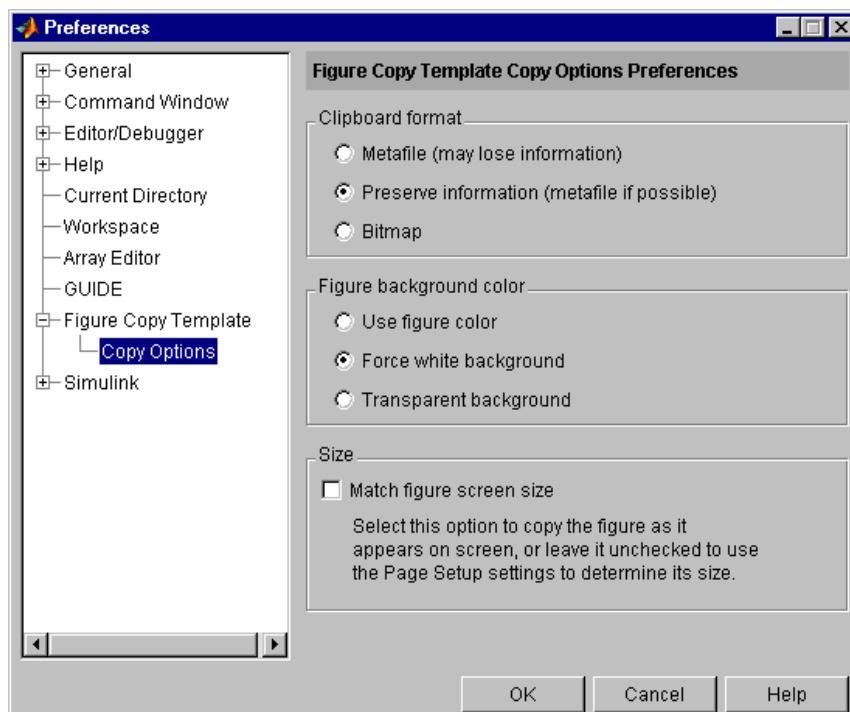


Figura 1.1: Janela de opções para copiar figuras

Pode-se escolher também a cor de fundo entre três opções: usar a cor da figura, a cor branca ou fundo transparente. Finalmente, é possível ajustar o tamanho da figura e sua posição na página.

Após selecionar as opções de cópia, escolha a opção "*Copy Figure*" no menu "*Edit*" do gráfico para copiar o gráfico no Matlab e depois use o comando "*Editar*" – "*Colar especial*" no MS-Word. É importante salientar aqui que, dependendo das opções de cópia, surgirão opções diferentes no momento de colar a figura no MS-Word. Aconselha-se não usar a opção "flutuar sobre texto".

Uma vez colado no MS-Word, é possível ajustar o tamanho do gráfico clicando uma vez em cima deste e redefinindo o tamanho com o auxílio do *mouse*.

1.5 Usando o Help

O *help on line* do Matlab deve ser usado para obter informações sobre os comandos, funções operadores e *toolboxes* que compõem o Matlab. Existem duas formas de obter *help on line*.

Digitando-se *help*, o Matlab apresenta uma lista de tópicos de ajuda dentro da janela de comando. Para obter informações sobre um tópico ou comando em especial digite *help + nome do tópico* que se deseja aprender.

```
>> help plot
```

Para obter ajuda sobre uma *toolbox* em especial, digite *help + nome da toolbox*. Por exemplo, para conhecer os comandos existentes na área de controle, ou seja, sobre a "toolbox control", digite:

```
>> help control
```

Helpwin apresenta os tópicos de ajuda numa janela fora da janela de comando do Matlab. Este comando é útil quando não se quer misturar o trabalho feito na janela de comando com o texto do tópico de ajuda. Assim, para obter o mesmo texto de ajuda do comando *plot* em uma janela separada digite:

```
>> helpwin plot
```

1.6 Salvando e Recuperando Variáveis

Para salvar os valores das variáveis usadas numa sessão do Matlab, a fim de que possam ser utilizadas em outro momento, utiliza-se o comando *Save*. Este comando salva todas as variáveis geradas pelo usuário num arquivo *matlab.mat* no diretório de trabalho do Matlab. É conveniente, entretanto, que o nome deste arquivo e o diretório no qual ele será colocado, sejam pessoais, como exemplificado a seguir.

```
>> save c:\exp1\teste % o nome do arquivo será teste.mat e ele será gravado no diretório c:\exp1
```

Obs.: Este comando salva apenas os valores das variáveis definidas numa sessão, não salvando gráficos, ou qualquer outro tipo de procedimento adotado no Matlab.

Tendo salvado as variáveis neste arquivo, o Matlab pode ser fechado para, numa próxima sessão, reutilizá-las, recuperando-as através do comando *load*, isto é:

```
>> load c:\exp1\teste % é necessário informar todo o caminho onde se encontra o arquivo
```

Ob.: Para maiores detalhes a respeito destes comandos, consulte o *help* do Matlab.

Existem ainda outros comandos que permitem trabalhar com as variáveis utilizadas. Os comandos *who* e *whos* fornecem uma lista das variáveis correntes. O comando *clear* limpa as variáveis.

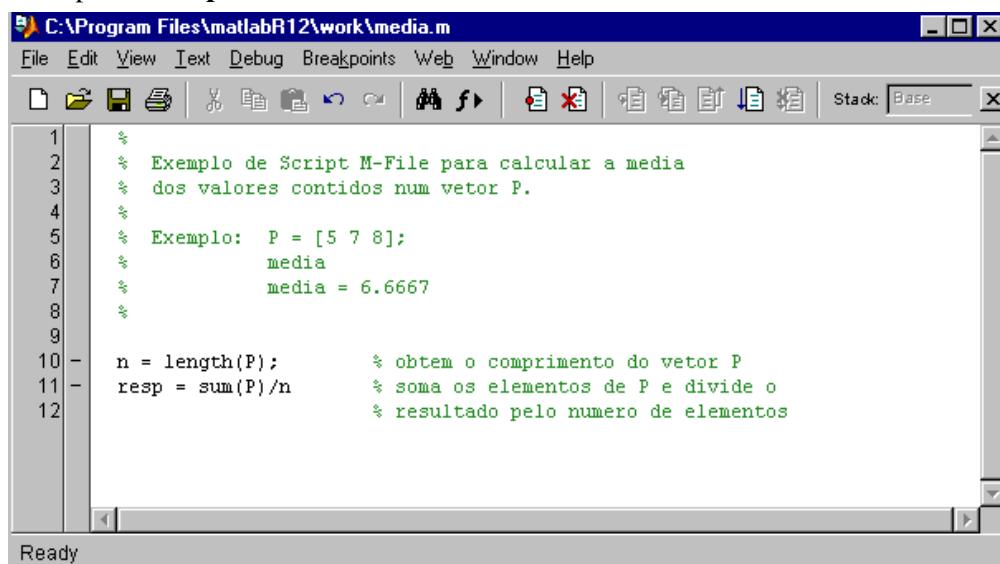
1.7 Programando no MATLAB – Uma introdução

Arquivos que contêm código de linguagem MATLAB são chamados de *M-files*. M-files podem ser *funções*, que aceitam argumentos e produzem um resultado, ou podem ser *scripts*, que executam uma série de declarações do MATLAB. O nome de um arquivo M-file deve ser seguido por “.m”.

Para escrever um *M-file* deve-se selecionar “*File+New+M-File*” ou clicar no botão de atalho . Com isso, o Editor/Debugger é aberto e um programa *M-file* pode ser criado.

Script M-files	Function M-files
➤ Não aceitam argumentos de entrada.	➤ Podem aceitar argumentos de entrada.
➤ Operam sobre dados definidos no <i>workspace</i> do Matlab.	➤ Por <i>default</i> , as variáveis internas são locais.
➤ Úteis para automatizar uma série de passos que necessitam ser repetidos várias vezes.	➤ Úteis para aumentar a linguagem do Matlab.

Um exemplo de Script M-file é dado abaixo:



```
C:\Program Files\matlabR12\work\media.m
File Edit View Text Debug Breakpoints Web Window Help
 1 % Exemplo de Script M-File para calcular a media
 2 % dos valores contidos num vetor P.
 3 %
 4 %
 5 % Exemplo: P = [5 7 8];
 6 %           media
 7 %           media = 6.6667
 8 %
 9 %
10 - n = length(P);      % obtém o comprimento do vetor P
11 - resp = sum(P)/n     % soma os elementos de P e divide o
12 % resultado pelo numero de elementos

Ready
```

Para salvar o *M-File* do tipo *script* ou do tipo *function*, utilize a opção “*Save*” do menu “*File*” do MATLAB Editor/Debugger, o que abrirá a janela mostrada abaixo (Windows 98). Escolha então um diretório apropriado para salvar seu arquivo, o nome do arquivo seguido da extensão “.m”.



Os comentários inseridos no arquivo servem para auxiliar os usuários que não sabem utilizar tal arquivo. Assim, pode-se utilizar o help do Matlab conforme mostrado a seguir.

```
>> help media
```

Uma propriedade muito importante de um *Script M-File* é que suas variáveis são globais, o que implica na necessidade de declará-las no Matlab. Note que se, no exemplo anterior, for usado um nome diferente para a variável P, por exemplo T = [1 3 5], não será possível obter o resultado, pois internamente a variável T não é utilizada e a variável P não existe.

```
>> T=[0 1 3];
>> media
??? Undefined function or variable P.
```

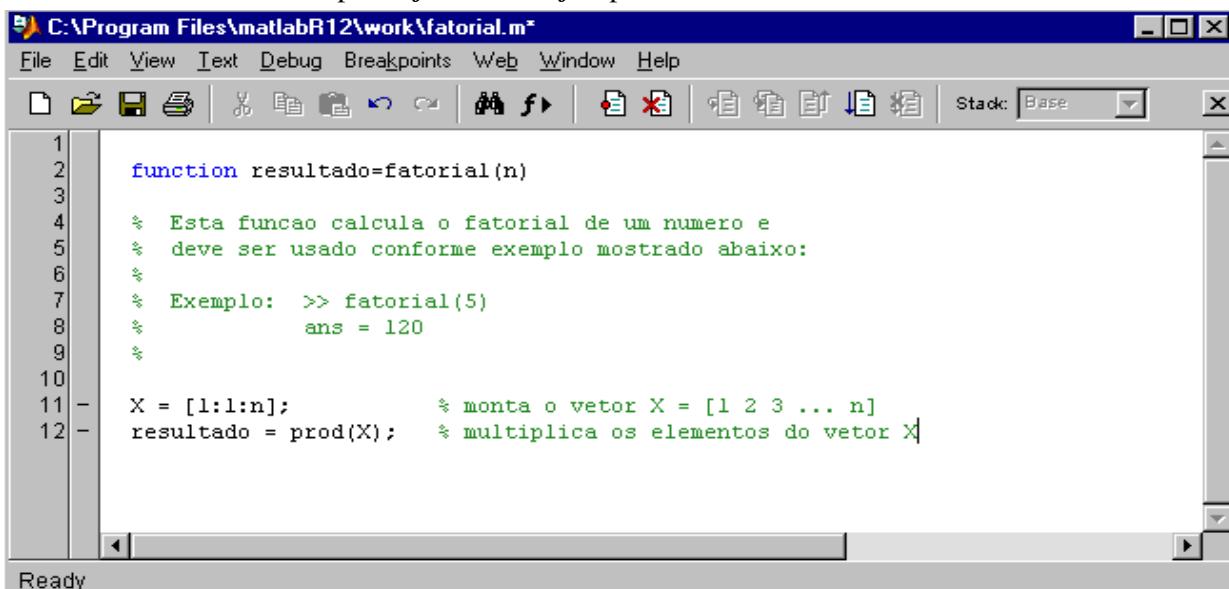
É necessário que, na declaração das variáveis, sejam utilizados os mesmos nomes internos ao arquivo. Desta forma, tem-se:

```
>> P=[0 1 3];
>> media
resp = 1.3333
```

Utilize agora este arquivo para calcular a média entre alguns valores e testar seu funcionamento!!!

Obs.: Para executar um arquivo script *M-File*, basta digitar seu nome na janela de comandos do Matlab.

Vamos fazer um exemplo de *function M-file* para treinar.



Salve esta função M-file com o nome de fatorial.m. Digitando *fatorial(n)* na janela de comando do Matlab, onde *n* é um número inteiro positivo, o factorial de tal número é obtido. Assim, por exemplo, ao digitar-se *fatorial(5)*, obtém-se a seguinte resposta:

```
>> fatorial (5)
      resultado = 120
```

A função do exemplo acima tem alguns elementos que são comuns a todas as funções M-file. A primeira linha define o nome da função, o número e a ordem dos argumentos de entrada e saída (*function[argumentos de saída] = nome da função [argumentos de entrada]*). Os termos entre colchetes são opcionais. O texto ajuda é aquele que se encontra depois do caractere %.

Digite *help fatorial* na janela de comando do Matlab e observe o resultado.

```
>> help fatorial
```

1.8 Definindo os Caminhos de Procura de Arquivos do Matlab

É importante ressaltar que os arquivos de usuários devem ser salvos num diretório que esteja no

caminho de procura do Matlab, ou seja, no seu **Matlab Path**. Para ver todos os diretórios em que o Matlab pesquisa arquivos do tipo M-file (ou outros), escolha “**File+Set Path**” na janela de comandos do Matlab. Com isso, a janela da *Figura 1.2* é exibida.

É possível adicionar caminhos (*paths*), clicando no botão **Add Folder ...** (*Figura 1.2*). No laboratório de ASL, por exemplo, é conveniente incluir o caminho G:\ASL\ e colocar os arquivos criados neste diretório ou num subdiretório do usuário.

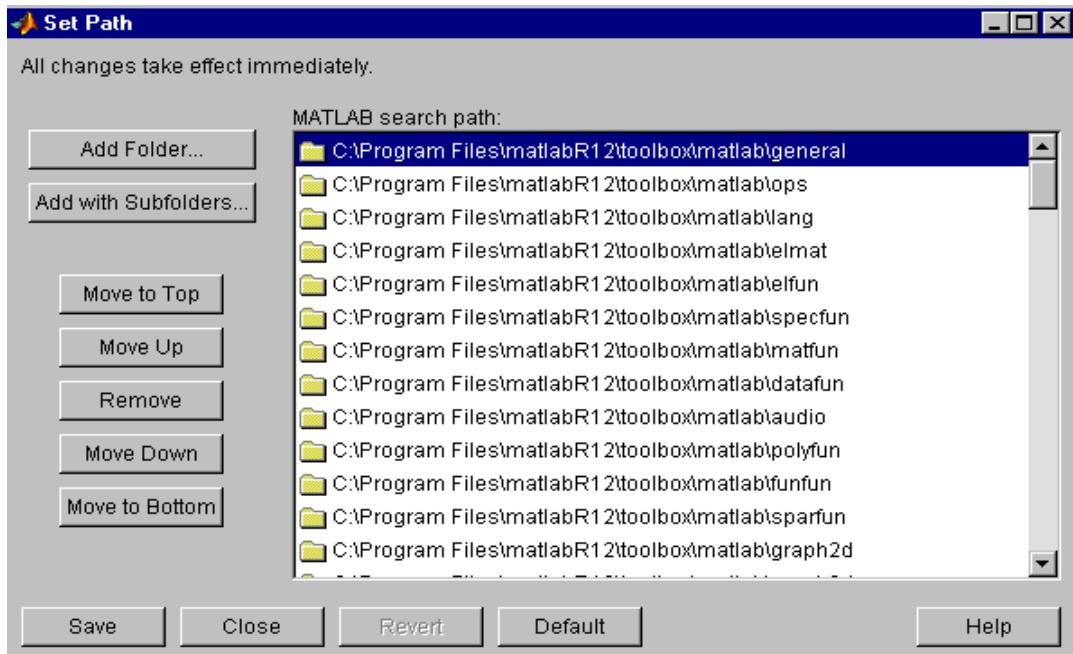


Figura 1.2: Janela do Matlab Path.

Numa **function**, podem ser empregados várias das estruturas normalmente usadas em linguagens de programação, tais como **if - else - end**, **for - end**, **while - end**. A seguir, são listadas algumas das mais usadas para realização dos exercícios propostos nessa experiência:

Comando	Sintaxe
fprintf	fprintf('O valor medido é %d e o valor calculado é %d',x,y)
input	N = input('Digite o número de provas ')
if else elseif	<pre>if x >= 5 fprintf('Texto 1') else fprintf('Texto 2') end</pre>

1.9 Tarefa

Faça um programa MatLab, que peça para introduzir os coeficientes de dois polinômios e apresente os seguintes resultados:

- a) Raízes de cada um dos polinômios;
- b) Gráfico de cada um dos polinômios;
- c) Divisão dos polinômios;
- d) Expansão em frações parciais da relação dos polinômios.
- O programa deve ser repetitivo, parando somente sob a ação de um comando específico.
- A escolha das funções deve ser feita através de um menu.
- Será valorizado o uso de **functions** para algumas das tarefas envolvidas.

1.10 Exercícios Sugeridos

Exercício 1 – Obtenha as raízes dos seguintes polinômios:

a) $s^2 + 4s + 5 = 0$

b) $s^3 + 6s^2 + 25s + 8 = 0$

Exercício 2 – Obtenha os polinômios que possuem as seguintes raízes:

a) $s = -2 \pm j2$

b) $s_1 = -10$ e $s_2 = -1 \pm j2$

Exercício 3 – Obtenha os polinômios resultantes das seguintes operações:

a) $(s^2 + 5) * (s + 2) / (s + 3)$

b) $(s + 1) * (s + 2) * (s + 3) * (s + 4)$

Obs.: Nas operações com divisão, mostre tanto o quociente como o resto obtido na operação.

Exercício 4 – Expanda as seguintes $B(s)/A(s)$ em frações parciais utilizando o Matlab:

a)
$$\frac{B(s)}{A(s)} = \frac{s^2 + 2s + 3}{(s + 1)^3}$$

b)
$$\frac{B(s)}{A(s)} = \frac{2s^3 + 5s^2 + 3s + 6}{s^3 + 6s^2 + 11s + 6}$$

Exercício 5 – Trace em um mesmo gráfico as funções $y_1 = x^2$ e $y_2 = x^{0.5}$. Considere uma variação de x entre 0 e 10 com intervalos de 0,01. Utilize os recursos do *software* para diferenciar as curvas em cor e tipo de linha; ascendente, também, legenda, título e o nome dos eixos.

Dica: Note que x é um vetor linha e que não é possível obter a operação matemática x^2 para esse tipo de vetor. Sendo assim, deve-se utilizar o recurso mostrado abaixo, de forma que a operação matemática seja executada sobre cada elemento escalar do vetor, e não sobre o vetor propriamente dito.

```
>> y1 = x.^2; % observe que foi utilizado um ponto logo após a variável x
```

Exercício 6 – Obtenha, num mesmo gráfico, as curvas de resposta das funções $y_1 = \sin(t)$ e $y_2 = \sin(2t + \pi/4)$ e verifique o período de cada um dos sinais. Considere que t varia de 0 a 4π . Não esqueça de usar os recursos do *software* para diferenciar as curvas e colocar título e nome nos eixos, além da legenda. É necessário mostrar apenas um gráfico, o qual deve conter as duas curvas traçadas, bem como uma indicação gráfica de como foi obtido o período de uma das senóides.

Exercício 7 – Crie um arquivo *script file* (M-File) que trace três curvas no mesmo gráfico. Este arquivo deve ainda fazer com que o gráfico tenha fundo branco, grid, e nome na escala x como sendo "Tempo (s)" e na escala y como sendo "Amplitude". Note que a faixa de variação de valores para o eixo das abscissas, bem como as funções que serão usadas para obter os gráficos devem ser fornecidas externamente no momento de uso do M-File, não podendo ser definidas internamente no arquivo.

Exercício 8 – Crie uma *function M-file* que calcule a média aritmética de um vetor P qualquer. Guie-se pelo arquivo *Script M-File media.m*, feito anteriormente nesta experiência.

Exercício 9 – Crie uma função que calcule a média semestral, considerando que são fornecidas as notas das quatro provas e a média do laboratório. Esta função deve retornar a média do aluno juntamente com o resultado "Aprovado" ou "Exame". Caso o aluno não seja aprovado, deve fornecer ainda a nota necessária no exame para obter aprovação.

2 INTRODUÇÃO AO SIMULINK

2.1 Introdução

É uma ferramenta para modelar, analisar e simular sistemas matemáticos e físicos, incluindo os não lineares, de uma forma simples, pois representa os sistemas na forma de diagramas de blocos, ao invés de equações matemáticas. O *Simulink* usa as funções definidas no Matlab numa forma gráfica e fácil de ser compreendida.

A seguir são descritos os fundamentos desta ferramenta.

2.2 Fundamentos do Simulink

Software de Simulação de Sistemas Dinâmicos - SIMULINK

Simulink é um programa para a simulação de sistemas dinâmicos. Como uma extensão do Matlab, o Simulink incorpora diferentes aspectos específicos de sistemas dinâmicos enquanto mantém a funcionalidade apresentada pelo Matlab.

O Simulink acrescenta uma nova classe de janelas. Nessas janelas, *modelos* na forma de diagrama de blocos são montados e editados principalmente por comandos feitos diretamente do mouse.

Para ter acesso ao Simulink, digite

```
>> simulink
```

na *janela de comando* do Matlab, ou clique no botão  na barra de ferramentas do mesmo.

Em seguida, é aberto um *browser* para a biblioteca do **Simulink** (Figura 2.1), onde são encontrados os blocos necessários para modelar um determinado sistema, via diagrama de blocos. Com isso, o Simulink está pronto para ser usado.

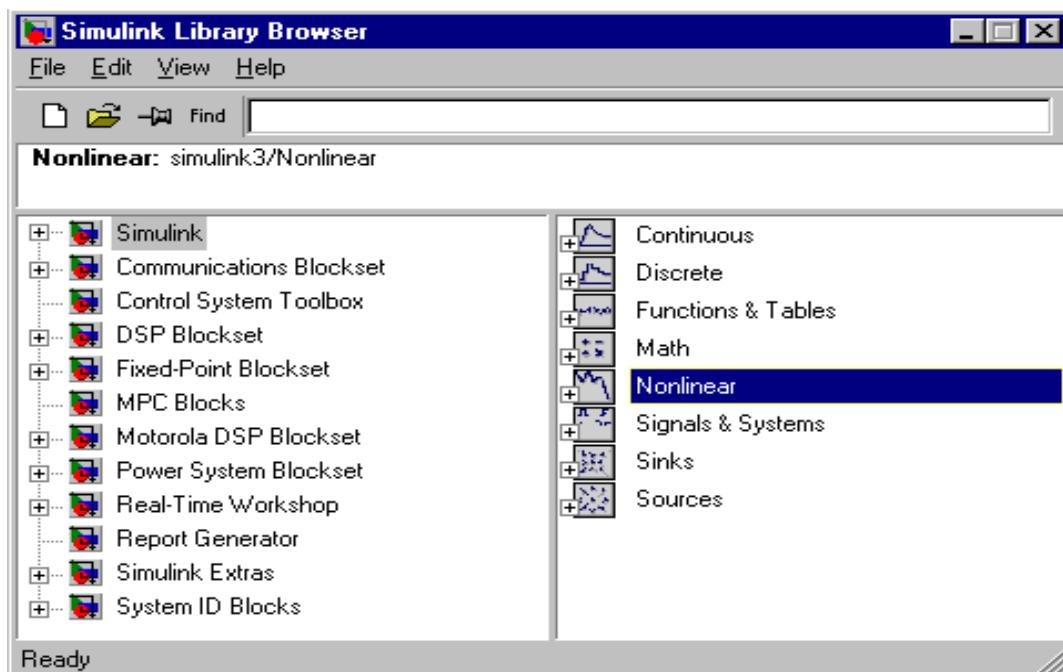


Figura 2.1: Biblioteca do Simulink

A Biblioteca do Simulink contém diversas sub-bibliotecas com conjuntos de funções aplicáveis aos

mais variados propósitos. Cada uma destas sub-bibliotecas possui grupos de blocos, alguns dos quais aparecem na janela da direita: **Continuous, Discrete, Functions & Tables, Math, Nonlinear, Signals & Systems, Sinks and Sources**. O menu superior do *browser* apresenta quatro opções: **File, Edit, View e Help**. A mais importante é **File**, que inclui a criação de um novo modelo (*New-Model*) ou de uma nova biblioteca (*New-Library*), a abertura de modelos ou bibliotecas (**Open**) já existentes e um conjunto de preferências (**Preferences**).

O uso do Simulink pode ser abordado em duas etapas: (i) a definição do modelo e (ii) a análise do modelo. Uma sessão típica inicia-se pela definição do modelo ou carregando um modelo pré-existente, para em seguida proceder à sua análise. Na prática, estas duas etapas são realizadas interativamente quando o projetista cria e modifica um modelo para determinar o seu comportamento.

Para criar um diagrama de blocos qualquer, primeiramente deve-se abrir um novo modelo (**File-New-Model**). Uma nova janela é aberta, na qual será criado o novo modelo, correspondente a um sistema físico. A seguir, copiam-se os blocos necessários das bibliotecas do Simulink, arrastando-os para a janela criada. Após copiados, os blocos devem ser interligados com auxílio do *mouse*.

A tabela abaixo apresenta uma lista com as ações mais freqüentes feitas com o mouse ou teclado para manipular blocos, linhas e outros. BDM significa clicar botão direito do mouse; BEM significa clicar com o botão esquerdo do mouse.

Tarefa	Microsoft Windows
Selecionar um bloco	BEM
Selecionar vários blocos	Shift + BEM
Copiar bloco de outra janela	Arrastar bloco com o BEM ou BDM
Mover bloco	Arrastar bloco com o BEM
Duplicar bloco	BDM e arrastar o bloco duplicado
Conectar blocos	BEM
Desconectar blocos	Shift + arrastar bloco com BEM ou BDM
Selecionar uma linha	BEM
Selecionar varias linhas	Shift + BEM
Desenhar ramo	BDM e arrastar linha
Criar anotação	clique duplo no diagrama, e depois escrever texto
Copiar anotação	BDM e arrastar anotação
Mover anotação	arrastar anotação com BEM

Depois de construído o novo diagrama de blocos, salve o arquivo com o nome desejado num diretório que você tenha permissão para tal. Os arquivos criados desta forma terão a extensão **.mdl** (nome.mdl) e poderão ser abertos futuramente. Para abrir um arquivo existente, no menu **File** clique em **Open** e escolha o arquivo que deseja abrir, indicando o diretório onde o mesmo se encontra.

Nota: Jamais modifique a biblioteca do Simulink e, se ao fechar a janela do próprio Simulink surgir a mensagem "Save Simulink before closing?", escolha a opção Não.

2.2.1 Descrição dos Grupos Básicos do Simulink

A sub-biblioteca **Simulink** contém as principais funções necessárias para projeto e análise de sistemas dinâmicos. Estas funções estão divididas em grupos distintos, que estão descritos a seguir.

Continuous: este grupo consiste de alguns blocos destinados à criação de sistemas contínuos, tais como o integrador, a função de transferência, etc.

Discrete: é composto de blocos semelhantes aos do grupo anterior, visando à implementação de sistemas discretos, os quais serão estudados na disciplina denominada Sistemas de Controle II.

Functions & Tables: Um bloco importante neste grupo é $f(u)$, o qual permite criar expressões simples, tais como $y=u^2$ e $y=sqrt(u)$ e expressões complexas, como $y = u^2 + abs(u) - sqrt(u)+ log10(u)$. É importante observar que este bloco permite a utilização de uma única variável independente (entrada) e requer que ela seja denotada pela letra u . Portanto, para obter a operação $y=x^2$ utiliza-se o bloco u^2 .

Math: disponibiliza várias funções matemáticas úteis para a criação de sistemas lineares e não-lineares.

Nonlinear: conjunto de blocos representando não-linearidades típicas: saturação, zona morta, etc.

Signals & Systems: este grupo possui blocos que permitem importar e exportar dados através de portas de entrada (Import) e portas de saída (Outport), além de blocos para fazer multiplexação de dados (***mux***) e demultiplexação de dados (***demux***). O bloco de multiplexação de dados pode ser usado, por exemplo, para possibilitar a visualização de mais de um sinal (curva) num mesmo gráfico.

Sinks: aqui estão os blocos para apresentação dos resultados de forma gráfica como Scope (tipo osciloscópio), To Workspace, display (resultado numérico), entre outros.

Sources: este grupo consiste na biblioteca de *fontes de sinais* tais como senóide, degrau (step), constante, gerador de sinais (onde podem ser gerados senóides, ondas quadradas, ruídos, etc.), entre outros.

2.2.2 Descrição das Opções do Menu da Janela do Modelo

Conforme colocado anteriormente, na janela aberta para criação de um novo modelo (diagrama de blocos) ou para utilização de um modelo já existente, existe um menu principal, cujas principais opções são descritas a seguir.

File:

- New... Model***: abre nova janela para montar um modelo;
- New... Library***: abre nova janela para montar uma biblioteca;
- Open....***: abre um arquivo já existente;
- Close***: fecha a janela de modelo;
- Save, Save as....***: salva modelo atual;
- Preferences***: permite ajustar diversas preferências;
- Print..., Printer Setup....***: imprime ou define configurações de impressão;
- Exit MATLAB***: sai do Matlab.

Edit:

- Cut, Copy, Paste, Clear***: recorta, copia, cola e apaga, respectivamente, os objetos selecionados;
- Select all***: seleciona todos os objetos da janela ativa;
- Copy Model***: copia modelo para ser colado em outro modelo ou em um editor de textos;
- Find***: acha objetos;
- Create Subsystem***: cria subsistema agrupando todos os blocos selecionados;
- Mask Subsystem***: modifica o ícone criando uma nova interface a partir do subsistema;
- Look Under Mask***: permite ver o subsistema por trás da nova interface.

Na Figura 2.2, é apresentado um exemplo de utilização da opção ***Create Subsystem***. O subsistema possui duas entradas e uma saída, a função matemática implementada é descrita por $y = 2 * x_1 + \sqrt{x_2}$, onde x_1 é uma variável associada à entrada 1 (In1), x_2 é uma variável associada à entrada 2 (In2) e y é uma variável associada à saída (Out1).

Quando o modelo é composto por muitos blocos é comum utilizar-se desta opção para agrupar alguns blocos que tenham uma função específica. Na Figura 2.3, mostra-se o diagrama de blocos que representa o modelo de um sistema de controle de temperatura de uma residência, sendo que foi criado um subsistema denominado ***House*** para englobar o modelo termodinâmico da residência.

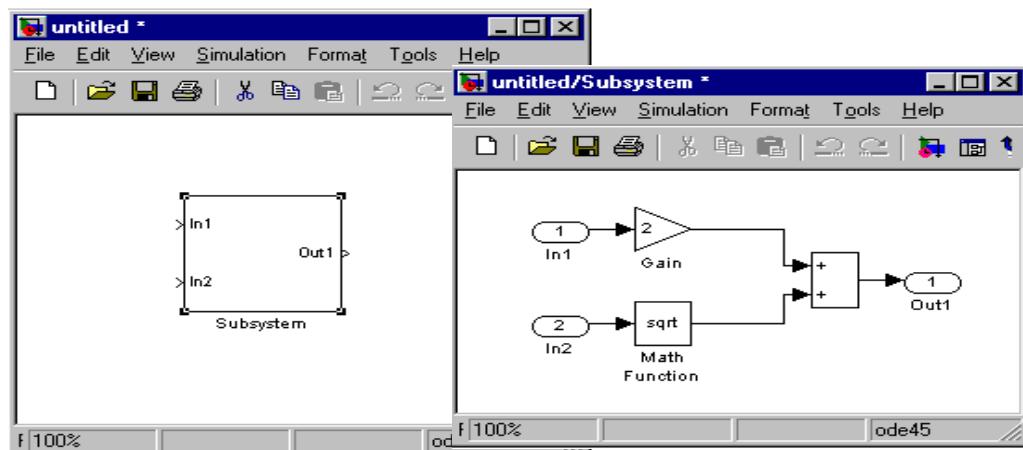
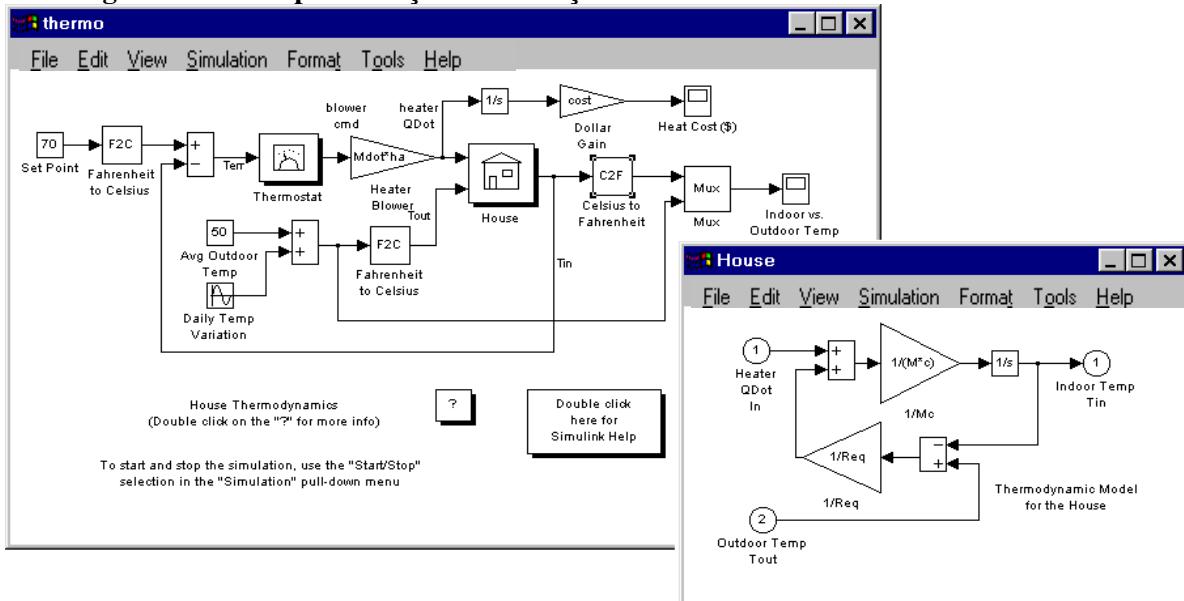


Figura 2.2: Exemplo de Criação de Subsistema utilizando a Opção Create Subsystem

Figura 2.3: Exemplo Avançado de Criação de Subsistema e Mascaramento de Bloco.



Neste último exemplo (Figura 2.3), o recurso ***mask*** foi usado para mudar a apresentação dos ícones *House* e *Thermostat*. Estes ícones podem ser criados utilizando o comando ***iconedit*** do Matlab.

View: ativa ou desativa barras de menus, *browsers*, etc.

O *Browser* do Modelo identifica blocos com a nova interface (*mask*), blocos com parâmetros *OpenFcn* definidos, e sistemas que contêm subsistemas usando símbolos antes do nome do bloco ou do sistema. Estes símbolos são:

- + : Significa que o sistema é expansível o que significa que tem sistemas dentro dele.
- : Quando o sistema já está expandido o símbolo menos (-) aparece.

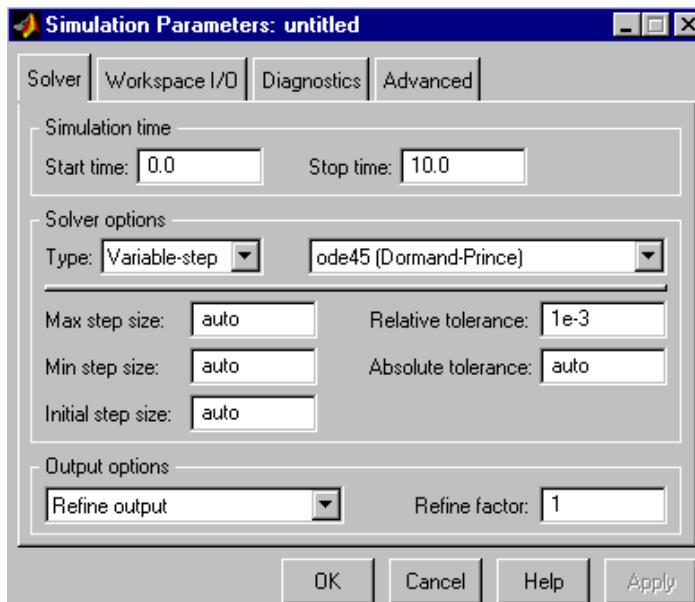
Usando este *browser*, podem ser abertos subsistemas, parâmetros de blocos, etc.

Simulation: permite iniciar, interromper e reiniciar as simulações, além de alterar os parâmetros da simulação (Para ver com detalhes os parâmetros de simulação leia a *subseção 2.2.3*).

Format: altera cores, tamanho e estilo das fontes e de apresentação (orientação, título e sombreamento) do bloco selecionado. Permite ainda girar o bloco selecionado de forma a facilitar as interligações.

2.2.3 Modificando os Parâmetros da Simulação

Antes de iniciar qualquer simulação, é necessário ajustar os seus parâmetros. Primeiramente, escolhe-se a opção **Parameters** no menu **Simulation**; então, a janela abaixo, denominada **Simulation parameters: Untitled** (ou o título do modelo), é aberta, permitindo o ajuste dos parâmetros.



As ações executadas ao pressionar os botões são resumidas a seguir:

Botão	Ação
OK	Aplica os valores de parâmetro e fecha a caixa de dialogo. Durante a simulação, os valores de parâmetro são aplicados imediatamente.
Cancel	Muda os valores dos parâmetros para os valores que a caixa de diálogo tinha quando esta foi recentemente aberta, e aplica estes valores para simulação.
Help	Janela ajuda ara a pagina da caixa de dialogo atual. Observe que temos cinco paginas na caixa de dialogo como visto na figura 2.3. Elas são: <i>Solver</i> , <i>Workspace I/O</i> , <i>Diagnostics</i> , <i>RTW</i> , <i>RTW External</i> .
Apply	Aplica o valor dos parâmetros atuais e mantém a caixa de dialogo (a janela da figura 2.3) aberta. Durante a simulação, o valor dos parâmetros é aplicado imediatamente.

A pasta **Solver** está dividida em três partes: **Simulation time**, **Solver options**, e **Output options**, que são explicadas nos próximos parágrafos.

- **Simulation time:** Os valores *default* dos tempos inicial e final de simulação, que são, respectivamente, 0.0 e 10.0 s, podem ser alterados nos campos **Start time** e **Stop time**. É importante ressaltar que as simulações não são feitas em tempo real, ou seja, o tempo de simulação não é o tempo real de relógio.
- **Solver Options:** A simulação de modelos no Simulink envolve integração numérica de um conjunto de equações diferenciais ordinárias (ODEs). A qualidade e rapidez da solução dependerão do método escolhido em *Solver Options*. Pode-se escolher entre *Variable-step solvers*, que mudam o tamanho do passo durante a simulação e possuem controle de erro, e *Fixed-step solvers*, que mantêm o passo constante e não possuem controle de erro.

Os algoritmos de passo variável (*Variable-step solvers*) são descritos brevemente a seguir:

1. **ode45** é baseado na fórmula explícita de Runge-Kutta (4,5), o par Dormand-Prince. É um excelente método numérico de solução para modelos contínuo e é o *default* do Simulink.
2. **ode23** é baseado em um par explícito Runge-Kutta (2,3) de Bogacki e Shampine.
3. **ode113** resolve na forma variável Adams-Bashforth-Moulton PECE.
4. **ode15s** resolve na forma variável e é baseado em diferenciação numérica de fórmulas (NDFs).

Serve para problemas complicados onde *ode45* não foi eficiente.

5. **ode23s** é baseado em uma fórmula modificada de Rosenbrock de segunda ordem. Pode resolver alguns tipos de problemas complicados onde *ode15s* não é eficiente.
6. **ode23t** é baseado em uma fórmula trapezoidal modificada.
7. **ode23tb** é baseado em uma fórmula trapezoidal mista.
8. **discrete:** O Simulink seleciona quando detecta um modelo que não tem estados contínuos.

Para os algoritmos de passo fixo (*Fixed-step solvers*):

1. **ode5** é uma versão de passo fixo do *ode 45*, a fórmula Dormand-Prince.
2. **ode4** é RK4, a fórmula de quarta ordem de Runge-Kutta.
3. **ode3** é uma versão fixa do *ode23*, a fórmula de Bogacki-Shampine.
4. **ode 2** é o método de Heun, também conhecido como a fórmula melhorada de Euler.
5. **ode 1** é o método de Euler.
6. **discrete** útil para modelos que não utilizam estados e onde o controle de erro não é importante.

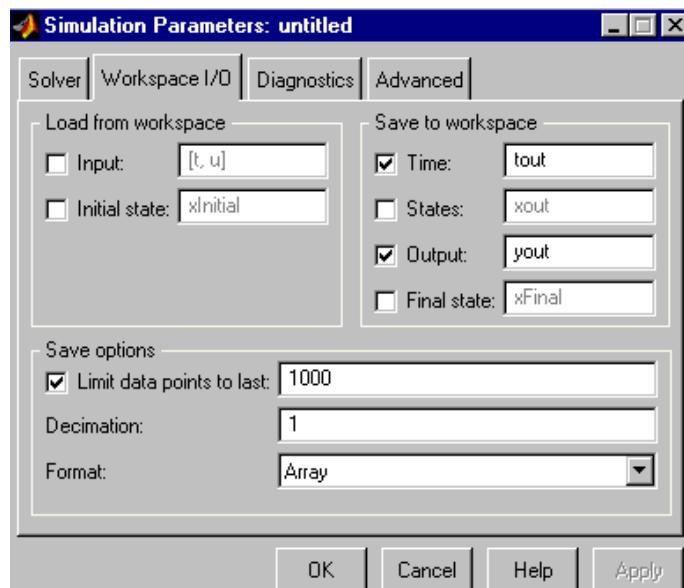
Ainda no item **Solver Options**, têm-se as seguintes opções:

- Step Sizes:** Consistem de passos mínimo e máximo para métodos numéricos de passo variável e no valor do passo para métodos de passo fixo. Em **auto**, o Matlab escolhe estes valores.
- Error Tolerances:** Técnicas padrão de controle de erro local são usadas para monitorar o erro passo a passo.

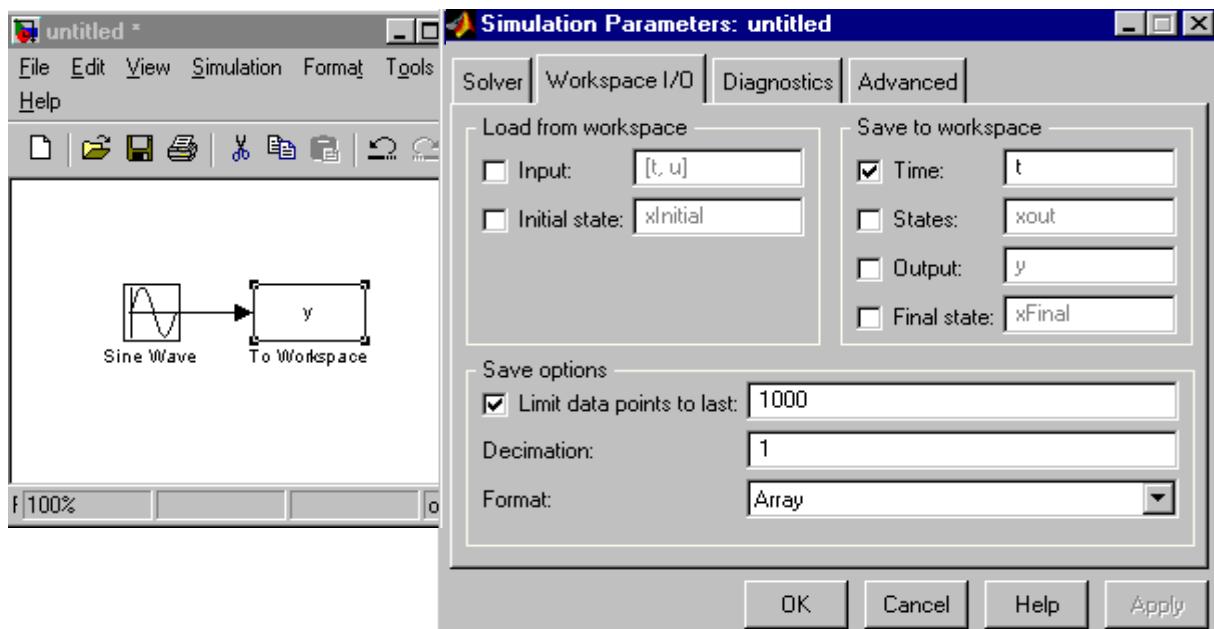
- **Output Options:** Permite controlar a saída, isto é, quantos pontos calculados pelo Simulink a simulação deve gerar.

A pasta **Workspace I/O** aparece ao ser selecionada a aba Workspace I/O. Pode-se direcionar a saída ou o resultado da simulação e armazenar em variáveis do Workspace (na janela de comando do Matlab) para que possam ser manipuladas também no Matlab. Isto é feito através do bloco **To Workspace** existente na biblioteca Sinks do Simulink e definindo uma variável para o **Workspace**, a qual armazenará os resultados da simulação feita no Simulink. É possível, por exemplo, traçar gráficos no Matlab usando o comando **plot** a partir dos resultados obtidos de simulações feitas no Simulink.

A próxima figura apresenta a pasta **Workspace I/O**, que inclui alguns parâmetros modificáveis. No caso de utilização do bloco **To Workspace**, pode-se salvar apenas a variável tempo para a janela de comandos do Matlab, pois a variável de saída é enviada pelo próprio bloco.



Para finalizar, apresenta-se um exemplo utilizando o bloco **To Workspace**. Após realizar a Simulação no Simulink, a curva da resposta é traçada no ambiente do Matlab. Primeiramente, construa o diagrama de blocos e escolha o nome da variável tempo (Time) a ser enviada para a janela de comandos do Matlab conforme ilustrado abaixo.



Na janela de comandos do Matlab, utilize o comando ***plot*** na seguinte forma:

```
>> plot(t,y,'-r')
```

Na pasta **Diagnostics**, pode-se indicar a ação desejada para vários tipos de eventos ou condições encontrados durante a simulação. Pode-se escolher entre erro (**Error**), cuidado (**Warning**) e sem mensagem (**None**). A mensagem “cuidado” não termina a simulação, a mensagem “erro” pára com a simulação.

2.3 Exercícios

Um sistema é linear se atende, simultaneamente, aos princípios da homogeneidade e da aditividade. Caso não atenda a, pelo menos um destes princípios, ele é não linear. Para exercitar o Simulink, proponha experimentos que verifiquem se um sistema é ou não linear, e se é ou não variante no tempo.

Para isso, construa os sistemas dados pelas equações abaixo, onde $x(t)$ representa a entrada e $y(t)$, a saída e verifique se estes sistemas são:

- a) Homogêneos/Não-Homogêneos
- b) Aditivos/Não-Aditivos
- c) Lineares/Não-Lineares
- d) Variantes/Invariantes no tempo

Sistema 1: $\frac{dy}{dt} + 2y = x$

Sistema 2: $2\frac{dy}{dt} + 0,6\sqrt{y} = x$

Sistema 3: $\frac{dy}{dt} + (1 + \cos 2t).y = x$

3 LINEARIZAÇÃO DE SISTEMAS NÃO-LINEARES

3.1 Introdução

Sistemas dinâmicos lineares são aqueles representados por equações diferenciais lineares. O termo linear refere-se à aplicabilidade do *Princípio da Linearidade*, isto é, se o sinal de entrada $u_1(t)$ produz como solução $y_1(t)$ e se o sinal de entrada $u_2(t)$ produz como solução $y_2(t)$, então o sinal de entrada $\alpha u_1(t) + \beta u_2(t)$ produzirá a solução $\alpha y_1(t) + \beta y_2(t)$, quaisquer que sejam os sinais $u_1(t)$ e $u_2(t)$ e os reais α e β .

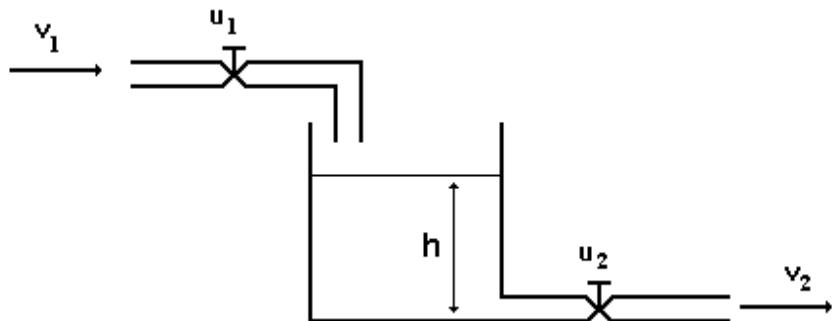
A análise e o projeto de sistemas lineares são extremamente facilitados devido à existência de soluções analíticas para equações diferenciais lineares. Entretanto, sistemas reais eletromecânicos, hidráulicos, pneumáticos, eletrônicos e muitos outros, em geral, envolvem relações não lineares entre suas variáveis. Quase sempre as equações destes sistemas não possuem soluções analíticas e freqüentemente é possível obter apenas estimativas ou soluções aproximadas das verdadeiras soluções.

Devido a estas dificuldades matemáticas inerentes a sistemas não lineares, normalmente é conveniente linearizá-los. Com isso, é possível usar as ferramentas de análise e projeto dos sistemas lineares, desde que seja respeitada a faixa de validade do modelo linearizado.

Nesta experiência, tem-se como objetivo avaliar o comportamento de um sistema não linear comparado a seu modelo linearizado. O sistema usado para este fim é um sistema de nível de líquido, cujo comportamento é descrito brevemente a seguir.

3.2 Sistema de Nível de Líquido

Considere um sistema de nível de líquido (presente em diversos ambientes industriais) como o da figura abaixo:



As variáveis do sistema de nível de líquido são: $h(t)$: o nível do fluido, u_1 : a abertura da válvula de entrada, u_2 : a abertura da válvula de saída, v_1 : a vazão na válvula de entrada e v_2 : a vazão na válvula de saída. As aberturas das válvulas de entrada e de saída variam de zero (completamente fechada) até um (completamente aberta).

A equação não linear que descreve o comportamento do sistema de nível é dada por:

$$A \frac{dh(t)}{dt} = k_1 u_1 - k_2 u_2 \sqrt{h(t)} \quad (1.1)$$

onde A é a área do tanque e k_1 e k_2 são constantes. Para avaliação das características do sistema, consideram-se as seguintes constantes:

$$A = 2 \text{ m}^2 \quad k_1 = 1,5 \quad k_2 = 0,8 \quad \text{e} \quad u_2 = 1 \quad (1.2)$$

3.3 Parte Teórica (Fazer antes da realização da experiência)

Utilizando as constantes definidas em (1.2) e baseando-se na equação (1.1), estude as características do sistema de nível de líquido seguindo o seguinte roteiro:

Exercício 1 - Obtenha uma representação em diagrama de blocos para o sistema representado pelo seu modelo não linear. Considere o nível h como a saída e a abertura de válvula u_1 como a entrada. ($u_2 = \text{cte.}$)

Exercício 2 - Linearize a função em torno de um ponto \bar{u}_1 genérico e obtenha o diagrama de blocos para o sistema linearizado. Considere o nível h como a saída e a abertura de válvula u_1 como a entrada.

Exercício 3 - Considere que a válvula de entrada seja posicionada em $\bar{u}_1 = 0,75$ (ponto de operação) e calcule o nível de estabilização do processo (\bar{h}).

3.4 Parte Experimental

Exercício 4 - Utilizando o Simulink, implemente o modelo não linear do sistema conforme diagrama de blocos obtido no exercício 1. Simule e verifique se o nível de estabilização do sistema coincide com o valor teórico encontrado no exercício 3. Considere $\bar{u}_1 = 0,75$.

Exercício 5 - Ainda usando o Simulink, implemente (juntamente com o modelo do exercício 4) o modelo linearizado do sistema. Varie a abertura da válvula de entrada e compare os resultados obtidos com os dois modelos, verificando o erro percentual resultante da utilização do modelo linearizado. Monte uma tabela de acordo com o modelo abaixo. **Comece verificando o comportamento dos dois modelos para uma variação nula da válvula de entrada. ($\Delta u_1 = 0$).** Obs.: Lembre-se que $u_1 \in [0;1]$.

Δu_1	u_1	$h_{\text{não linear}} (\text{m})$	$h_{\text{linearizado}} (\text{m})$	erro %
-0,25				
-0,15				
-0,10				
-0,05				
-0,02				
0,00				
0,02				
0,05				
0,10				
0,15				
0,25				

Analise e comente detalhadamente esta tabela.

O cálculo do erro% deve ser feito utilizando a seguinte fórmula:

$$\text{erro \%} = \frac{\Delta h_{\text{real}} - \Delta h_{\text{aprox}}}{\Delta h_{\text{real}}} * 100\%$$

onde: $\Delta h_{\text{real}} = h_{\text{real}} - \bar{h}$ e $\Delta h_{\text{aprox}} = h_{\text{aprox}} - \bar{h}$

portanto,

$$\boxed{\text{erro \%} = \frac{h_{\text{real}} - h_{\text{aprox}}}{\Delta h_{\text{real}}} * 100\%}$$

Exercício 6 - Use o Matlab para traçar as seguintes "curvas estáticas". (Não use o Simulink!)

- Num gráfico, trace as curvas $h = f(u_1)$ para os modelos não linear e linearizado em torno do ponto $\bar{u}_1 = 0,75$. Analise e comente este gráfico;
- Em outro gráfico, trace as curvas $h = f(u_1)$ para os modelos não linear e linearizado em torno do ponto $\bar{u}_1 = 0,4$. Analise e comente este gráfico;
- Compare os dois gráficos e observe que o sistema tem regiões onde a não-linearidade é mais acentuada, de forma que, mesmo para pequenas variações em torno do ponto de equilíbrio, o modelo linearizado não representa de forma fiel o comportamento do sistema real.

Dica: Uma equação estática significa que as variáveis estão estáticas, ou estacionárias, num ponto de equilíbrio. Note que ao variarmos a abertura da válvula u_1 , o nível estabilizará num valor $h \neq \bar{h}$. Sendo assim, deseja-se saber o valor de estabilização para os mais diversos valores de $u_1 \in [0,1]$. Note ainda que num ponto de equilíbrio qualquer $dh/dt = 0$.

Exercício 7 - Construa um gráfico $erro\% = f(u_1)$ para cada um dos modelos linearizados obtidos anteriormente. Considere que o modelo linearizado representa de forma satisfatória o modelo real se o erro% relativo ao valor de estabilização for menor que 10%. Determine então a faixa de variação da abertura da válvula de entrada para a qual podemos utilizar tais modelos em substituição ao não linear.

Note que o cálculo do erro% deve ser feito utilizando a fórmula apresentada anteriormente, ou seja:

$$erro\% = \frac{h_{real} - h_{aprox}}{\Delta h_{real}} * 100\%$$

Dica: A divisão de vetores no Matlab deve ser feita de forma escalar, conforme mostrado abaixo.

<code>>> u1=[0:0.01:1];</code>	% definir intervalo de variação de u_1
<code>>> hreal=.....;</code>	% fornecer a equação $hreal=f(u_1)$
<code>>> hb=.....;</code>	% hb significa h
<code>>> haprox=.....;</code>	% fornecer a equação $haprox=f(u_1)$
<code>>> erro=((hreal-haprox)./(hreal-hb))*100</code>	% fornecer a equação para erro%
<code>>> plot(u1,erro)</code>	% traçar a equação $erro\% = f(u_1)$

Note que foi usado um ponto antes do sinal de divisão, conforme explicado no roteiro da experiência 1.

Determine, também, a faixa de valores para a qual pode ser variado o nível real do tanque de forma que o modelo linearizado obtido anteriormente represente de forma satisfatória o modelo real.

4 ANÁLISE DA RESPOSTA TRANSITÓRIA DE SISTEMAS DE 1^A ORDEM

4.1 Introdução

Vários sistemas físicos e econômicos podem ser modelados matematicamente através de equações diferenciais lineares ou equações a diferenças lineares. Para análise e projeto de sistemas de controle deve-se ter uma base para comparar o desempenho de diferentes sistemas de controle. Esta base pode ser obtida especificando-se sinais de teste de entrada particulares e comparando-se as respostas de diversos sistemas a estes sinais de entrada. A utilização de sinais de teste pode ser justificada pela correlação que existe entre as características de um sistema para um sinal de entrada de teste típico e a capacidade do sistema para responder aos sinais de entrada reais. Assim, a partir das representações matemáticas e de sinais de teste, é possível avaliar o comportamento dinâmico das diferentes plantas.

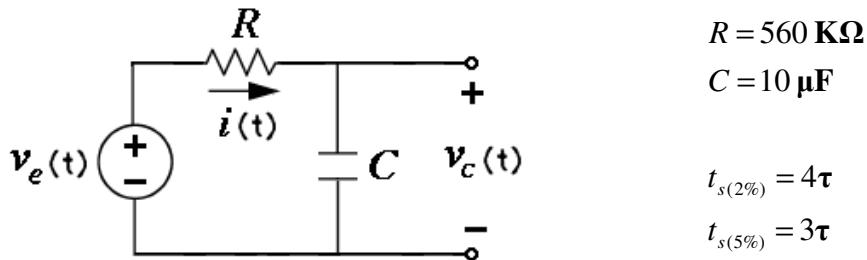
Os sinais de entrada de teste comumente utilizados são: degrau, rampa, impulso e senóide. A determinação de qual ou quais destes sinais de entrada típicos devem ser aplicados para analisar as características do sistema depende da forma de entrada a qual o sistema está sujeito mais freqüentemente durante a operação normal. Se as entradas para um sistema de controle variam gradativamente com o tempo, então uma rampa pode ser um bom sinal de teste. Para um sistema sujeito a entradas do tipo “choque”, um impulso pode ser a entrada adequada. Uma vez projetado um sistema de controle baseado em sinais de teste, normalmente o desempenho do sistema para entradas reais é satisfatório.

O objetivo desta experiência é analisar algumas propriedades dos sistemas de primeira e segunda ordem, tais como, resposta impulsiva, resposta à rampa e resposta ao degrau. Para isso, é importante distinguir entre:

- **Resposta Transitória :** parte da resposta de um sistema que desaparece quando o tempo tende ao infinito.
- **Resposta Estacionária :** parte da resposta de um sistema que permanece quando o tempo tende ao infinito.

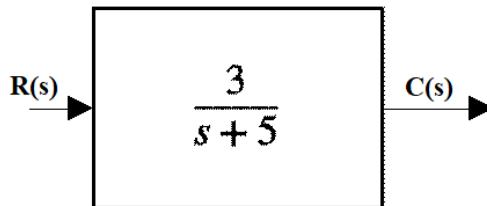
4.2 Exercícios

Exercício 1 - Encontre a função de transferência $V_c(s)/V_e(s)$, implemente-a no Simulink e observe o comportamento da resposta quando é aplicada uma tensão de 5 Volts na forma de degrau. Obtenha o instante em que a resposta alcança o regime permanente, considerando uma tolerância de 2% e de 5% e compare com os valores teóricos.



Mude a entrada do sistema para uma rampa de inclinação 3. Mostre as curvas de entrada e de saída no mesmo gráfico e observe que, em regime permanente, a saída do sistema não acompanha a entrada. *Obtenha o valor do erro em regime estacionário e justifique-o matematicamente.*

Exercício 2 – Considere o sistema de primeira ordem mostrado abaixo:

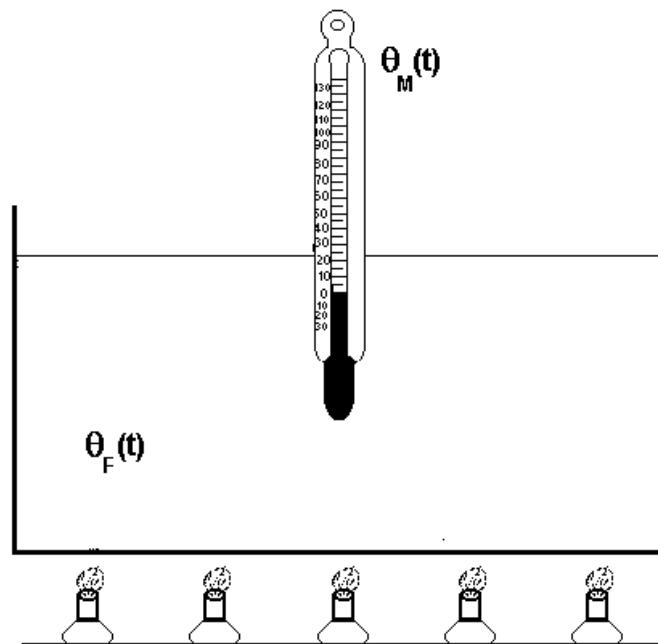


- Encontre a resposta analítica $c(t)$ para este sistema quando submetido a uma entrada em degrau de amplitude 2, ou seja, $r(t)=2u(t)$;
- Usando o Matlab (não o Simulink), plote num só gráfico a resposta obtida da simulação do sistema e a resposta calculada a partir da função analítica $c(t)$, determinada no item anterior. Compare os resultados. **Nota:** Para obter a resposta a partir da simulação do sistema, use o comando `step(num,den,t)` a ser explicado pelo professor.
- Usando o teorema do valor final, calcule o valor de estabilização da saída em regime permanente e compare com o valor obtido na alínea anterior.

Exercício 3 - Um termômetro, caracterizado pelos parâmetros capacitância térmica (**C**) e resistência térmica (**R**), pode ser modelado como um sistema linear de 1^a ordem. Este termômetro, cuja temperatura inicial é $\theta_M(0) = \theta_0$, é imerso num recipiente com água à temperatura $\theta_F(t)$.

- Deduza a equação diferencial que descreve o comportamento dinâmico deste sistema térmico (termômetro sujeito a uma fonte externa de calor);
- Aplicando a Transformada de Laplace a este modelo diferencial, obtenha sua correspondente função de transferência;
- A temperatura inicial deste termômetro é 25 °C, quando ele é subitamente imerso num recipiente com água à temperatura constante de 55 °C. Sabendo que sua temperatura leva 3 minutos para alcançar 50 °C, determine sua constante de tempo.
- Crie um modelo do termômetro usando o Simulink, simule e analise o resultado.
- Considere, agora, que a temperatura da água no recipiente seja inicialmente 25 °C e cresça linearmente a uma taxa de 6 °C/min. Calcule o erro que este termômetro apresentará? Implemente

esta situação usando o Simulink, simule e analise o resultado.



5 ANÁLISE DA RESPOSTA TRANSITÓRIA DE SISTEMAS DE 2^A ORDEM

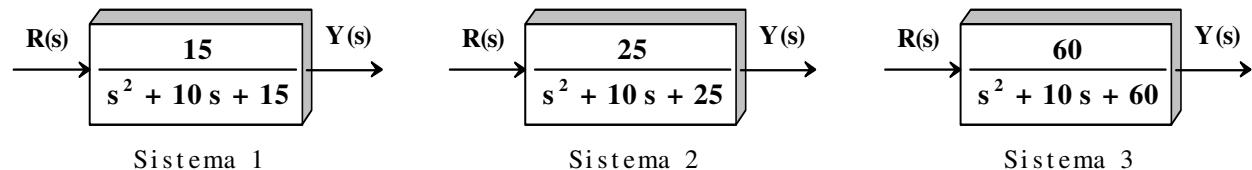
5.1 Introdução

Em muitos casos práticos, as características de desempenho desejadas para sistemas de controle são especificadas em termos de grandezas no domínio do tempo. Frequentemente, elas são especificadas em termos da resposta transitória para uma entrada em degrau, pois esta entrada é muito fácil de gerar fisicamente e é suficientemente severa. Além disso, o conhecimento da resposta ao degrau permite computar a resposta para qualquer entrada.

O estudo da resposta transitória de sistemas de segunda ordem é muito importante, uma vez que uma grande parcela dos sistemas físicos ou são deste tipo ou têm um comportamento tal que podem ser aproximados por um sistema de segunda ordem equivalente. Por isso, esta experiência tem como objetivo fixar os principais conceitos sobre esta importante classe de sistemas.

5.2 Exercícios

Exercício 1 - Sejam os sistemas abaixo. Implemente-os no MATLAB e obtenha, num mesmo gráfico, suas respostas para uma entrada tipo degrau unitário.



DICA: Para traçar as curvas de resposta ao degrau no Matlab, utiliza-se o comando *step*. No entanto, este comando não permite diferenciar as curvas da mesma forma que o comando *plot*. Sendo assim, para que possamos escolher a cor e o tipo de linha, o seguinte procedimento é aplicado:

```

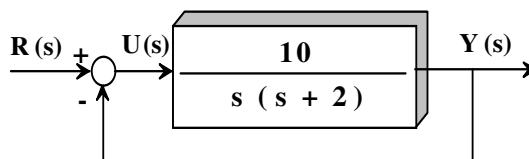
>> t = 0:0.001:4; % definir o tempo e o passo de simulação
>> y = step(num, den, t); % armazenar numa variável o resultado obtido na aplicação do degrau
>> plot(t, y, '-r') % tracar a curva usando o comando plot já conhecido

```

- a) Usando o comando ***roots***, ensinado na aula de introdução ao MatLab, verifique a posição dos pólos para cada um dos sistemas e classifique-os quanto aos casos de amortecimento.
 - b) Comparando as FTs dos sistemas apresentados aqui com a FT de um sistema típico de 2^a ordem, determine o ξ de cada um deles e verifique se a classificação, feita no item anterior, está de acordo com a teoria. Observe e comente as características das respostas, comparando-as entre si.

NOTA: Observe que as respostas ao degrau poderiam ter sido obtidas mais facilmente com o uso do Simulink. Entretanto, o cálculo das raízes de um polinômio só pode ser feito no Matlab, o que demanda a inserção das FTs de cada sistema no ambiente Matlab.

Exercício 2 – Implemente o sistema abaixo no Matlab e obtenha sua resposta ao degrau unitário.



DICA: Da mesma forma que no exercício anterior, a implementação deste sistema deve ser feita no Matlab e não no Simulink. Um novo comando é introduzido para auxiliar na tarefa de obter a FT de malha fechada de um sistema realimentado, usando a FT do caminho direto $G(s) = N_1(s)/D_1(s)$ e a FT do caminho de retroação $G(s) = N_2(s)/D_2(s)$. Para isso, siga o procedimento abaixo:

```
>> N1 = 10; D1 = [1 2 0]; % Definição de G(s)
>> N2 = 1; D2 = 1; % Definição de H(s)
>> [Nmf, Dmf] = feedback(N1, D1, N2, D2) % Obtenção da FT de malha fechada
>> step(Nmf, Dmf) % Aplicação do comando step à FTMF
```

- Usando o gráfico, determine a freqüência (em rad/s) com que esta resposta oscila.
- Calcule ω_n e ω_d e compare com o resultado do item a.

Exercício 3 – Implemente os sistemas abaixo no Matlab e obtenha suas respostas (em malha aberta) para entrada degrau unitário.

$$G_1(s) = \frac{4.8}{s^2 + 2s + 4.8} \quad G_2(s) = \frac{2.8}{s^2 + 2s + 2.8} \quad G_3(s) = \frac{2}{s^2 + 2s + 2}$$

- Usando o comando **roots** do Matlab, obtenha os pólos para cada um dos sistemas e mostre, num único gráfico, a localização desses pólos. Observe que todos têm partes reais - σ iguais.
- Obtenha o $t_s(5\%)$ de forma **experimental** e **teórica** para cada um dos sistemas e compare os resultados. Observe que o resultado teórico é o mesmo para os três sistemas, pois o produto $\xi\omega_n$ (parte real dos pólos) não muda, mas que os resultados experimentais não são iguais.
Justifique!!!

Exercício 4 – Implemente os sistemas abaixo no Matlab e obtenha suas respostas (em malha aberta) para entrada degrau unitário.

$$G_1(s) = \frac{4.8}{s^2 + 2s + 4.8} \quad G_2(s) = \frac{19.2}{s^2 + 4s + 19.2} \quad G_3(s) = \frac{173}{s^2 + 12s + 173}$$

- Usando o comando **roots** do Matlab, verifique a posição dos pólos para cada um dos sistemas. Mostre, num mesmo gráfico, a localização dos pólos para os três sistemas e observe que todos os pólos estão sobre a reta que faz um ângulo $\beta \approx 62.87^\circ$ com a origem.
- Determine os valores de M_p experimental para os três sistemas e comente os resultados com base na localização dos pólos.
- Determine t_p , t_r e $t_s(2\%)$ para os três sistemas e comente os resultados em relação aos pólos.

Exercício 5 - Sejam os sistemas abaixo. Implemente-os no Matlab (não implementar no Simulink) e obtenha, **ISOLADAMENTE**, suas respostas para uma entrada tipo degrau unitário. Utilize a função “**step**” e assegure-se de obter uma curva suave.

$$G_1(s) = \frac{9}{s^2 + 9} \quad G_2(s) = \frac{100,25}{s^2 - s + 100,25} \quad G_3(s) = \frac{2}{s^2 + s - 2}$$

- Usando o comando **roots** do Matlab, verifique a posição dos pólos para cada um dos sistemas.
- A partir da informação de localização dos pólos obtida no item anterior, classifique os sistemas quanto aos casos de amortecimento.

6 ANÁLISE DA RESPOSTA TRANSITÓRIA DE SISTEMAS DE 2^A ORDEM E SUPERIOR

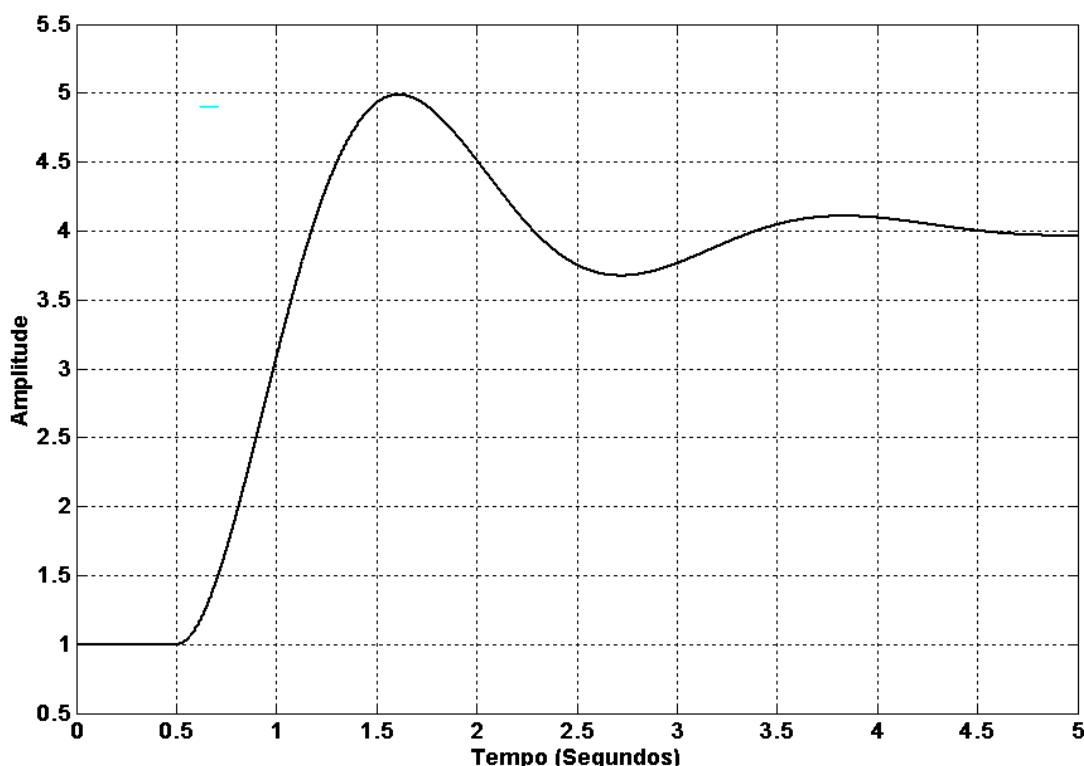
6.1 Introdução

Conforme explicado na experiência anterior, o estudo da resposta transitória de sistemas de segunda ordem é muito importante uma vez que grande parte dos sistemas físicos é deste tipo, ou ao menos pode ser aproximada, com boa precisão, por um modelo desta ordem. Esta experiência continua abordando o comportamento transitório de sistemas de segunda ordem sob a influência de pólos e zeros adicionais.

Primeiramente, porém, solicita-se ao aluno exercitar os conhecimentos de sistemas de segunda para estimar o sistema a partir da sua resposta temporal.

6.2 Exercícios

Exercício 1 – A seguir é mostrada a resposta de um sistema real à entrada degrau de amplitude 2.

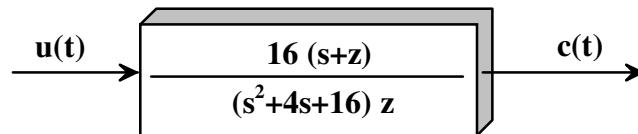


- Determine uma função de transferência que represente o modelo matemático deste sistema. Utilize o Simulink para simular o sistema encontrado e verifique se a resposta produzida é igual a do gráfico acima. Não esqueça que o degrau aplicado foi de amplitude dois.
- Obtenha a resposta deste mesmo sistema para uma entrada degrau de amplitude 3. Determine M_p , t_p e t_r e compare com os valores obtidos através do gráfico acima.

→ Observe que há uma parte da resposta que é constante.

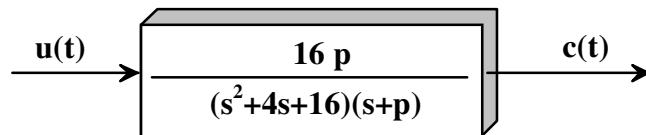
Exercício 2 - Seja o sistema em malha aberta mostrado abaixo. Faça a variável “z” assumir os seguintes

valores: $z = \infty^1$; $z = 12$; $z = 6$; $z = 4$ e $z = 2$ e implemente os cinco sistemas no Simulink, observando, num único gráfico, suas respostas para uma entrada tipo degrau unitário.



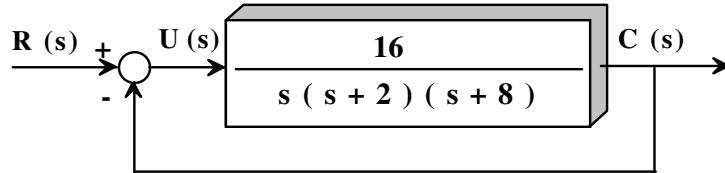
- Obtenha os valores experimentais de M_p , t_p , t_r e t_s (critério 5%);
- Compare os valores obtidos com os fornecidos pelos ábacos anexos;
- Analise qualitativamente o que ocorre com a resposta do sistema na presença do zero adicional.

Exercício 3 - Seja o sistema em malha aberta mostrado abaixo. Faça a variável “p” assumir os seguintes valores: $p = \infty^2$; $p = 12$; $p = 6$; $p = 4$ e $p = 2$ e implemente os cinco sistemas no Simulink, observando num único gráfico suas respostas para uma entrada tipo degrau unitário.



Analise qualitativamente o que ocorre com a resposta do sistema na presença do pólo adicional.

Exercício 4 – Considere que o sistema mostrado abaixo é submetido a uma entrada em degrau unitário.

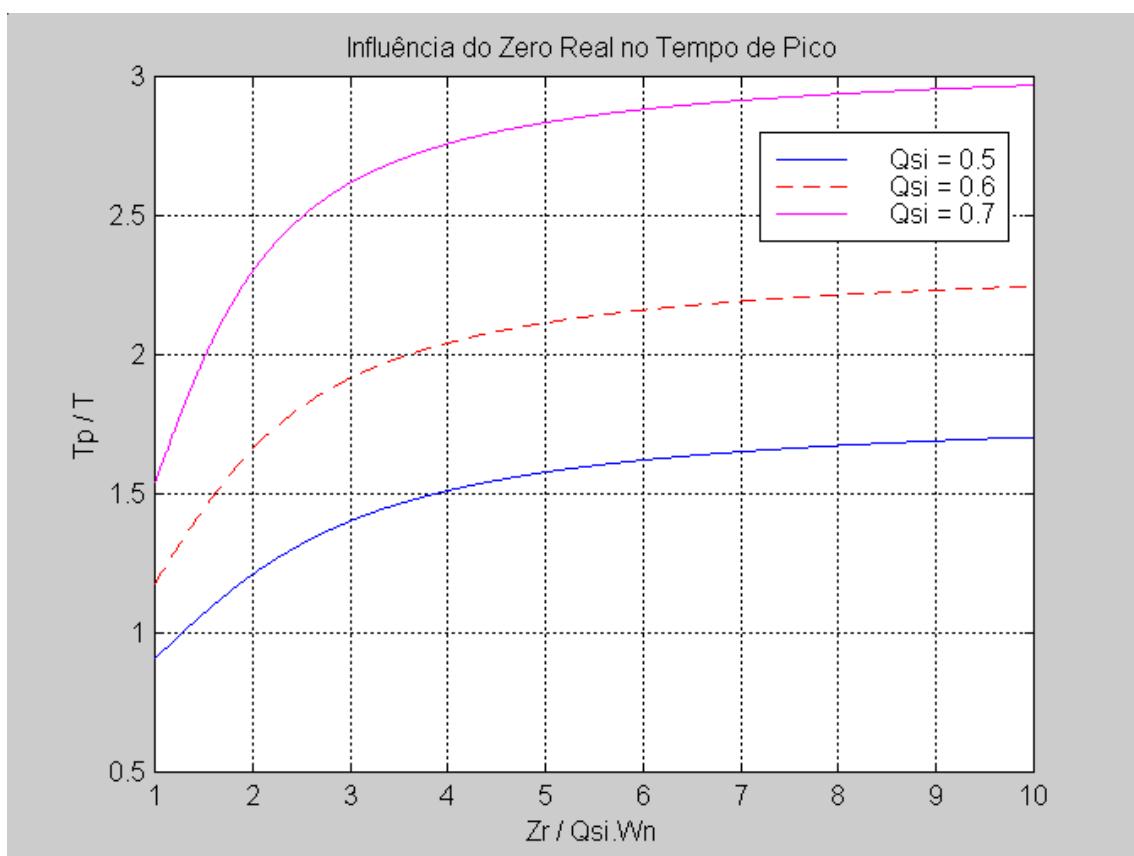
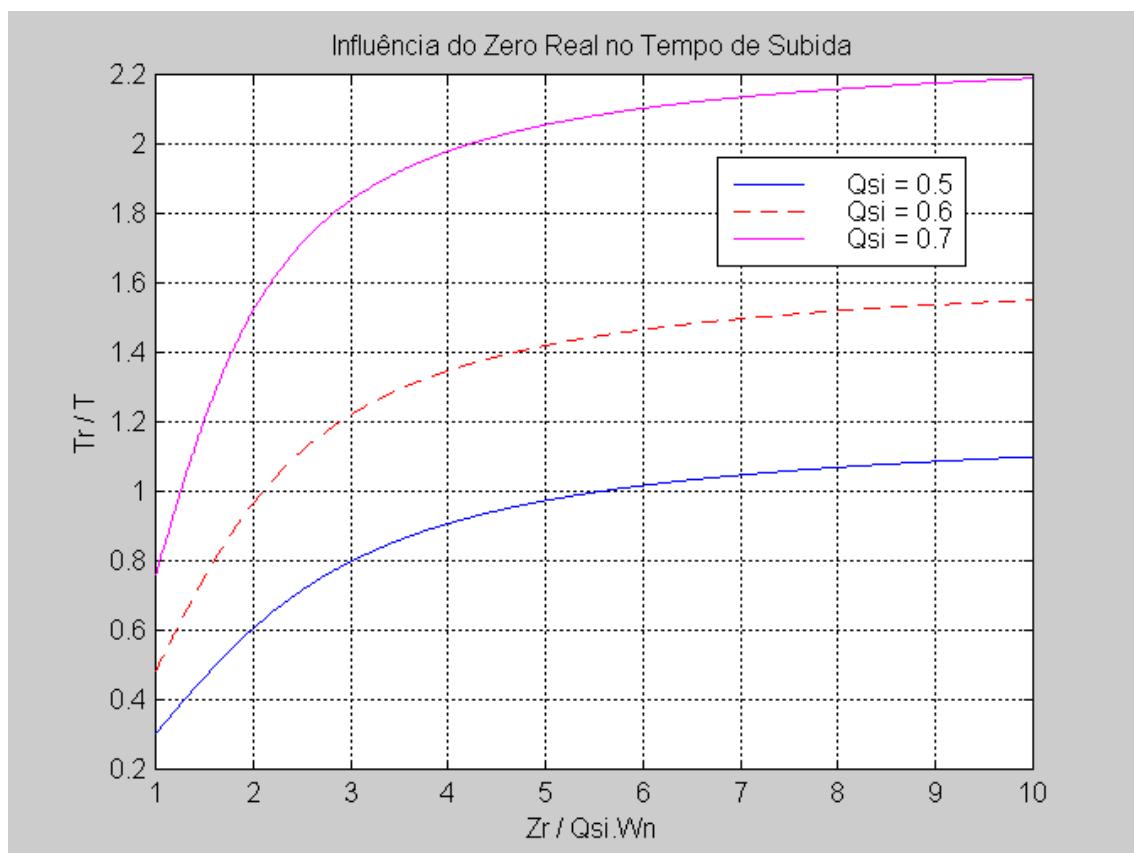


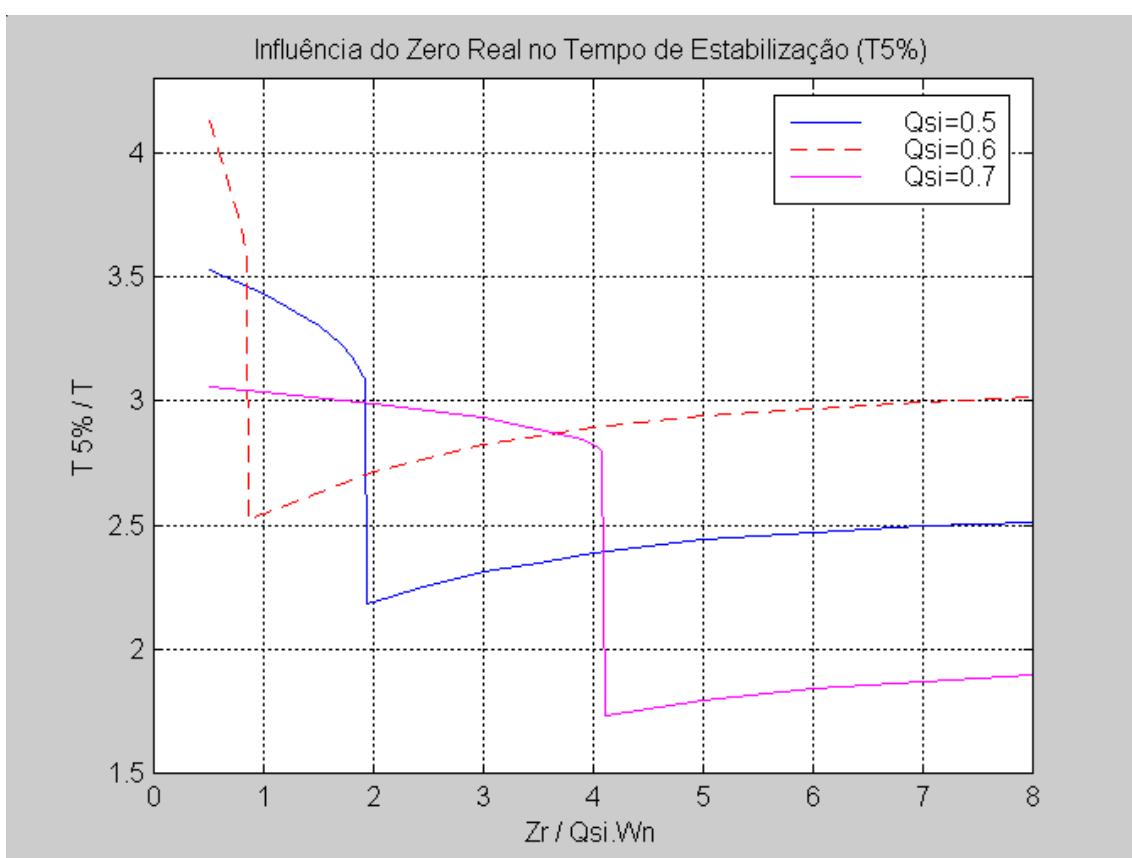
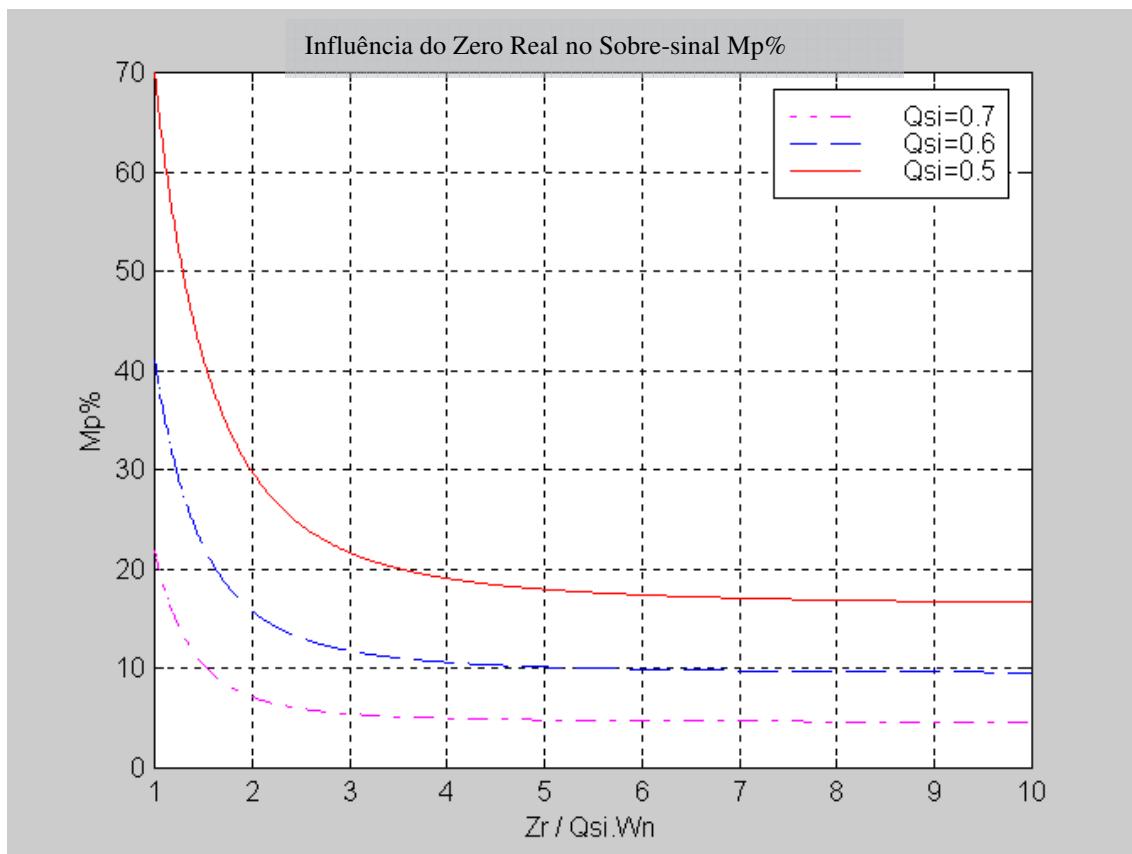
- Obtenha os valores experimentais de M_p , t_p , t_r e t_s (critério 5%) para este sistema.
- Obtenha uma função de transferência de segunda ordem $C(s)/R(s)$ que possa ser utilizada em substituição ao sistema mostrado acima.
- Compare a resposta ao degrau do sistema obtido no item anterior com o sistema original, comparando e justificando os resultados (obtenha as duas curvas num mesmo gráfico).

¹ Para fazer isto, basta desprezar a existência deste zero, o que resulta no sistema de segunda ordem original.

² Para fazer isto, basta desprezar a existência deste pólo, o que resulta no sistema de segunda ordem original.

6.3 Ábacos: Influência do Zero Adicional





7 ANÁLISE DA RESPOSTA EM FREQÜÊNCIA ATRAVÉS DOS DIAGRAMAS DE BODE

7.1 Introdução

Nesta experiência, serão abordadas as rotinas do MatLab que geram os Diagramas de Bode e cooperam na análise das características de sistemas lineares a partir de sua resposta em freqüência.

7.2 Traçando Diagramas de Bode no Matlab

O traçado de Diagramas de Bode pode ser feito de diversas formas. A mais simples e rápida consiste em utilizar uma função da *Toolbox Control* que acompanha o Matlab, cujo uso é ilustrado abaixo:

```
>>num=18;
>>den=[1 10];
>>bode(num,den)
```

Observe que o comando *bode(num,den)* traça os diagramas de módulo e fase numa mesma figura e que a determinação das escalas é feita pelo próprio Matlab.

O mesmo diagrama, em uma faixa de freqüência definida pelo usuário, é mostrado a seguir:

```
>>w=0.1:0.1:1000;
>>bode (num, den, w)
```

Note que a escala das freqüências foi alterada para acompanhar a modificação realizada.

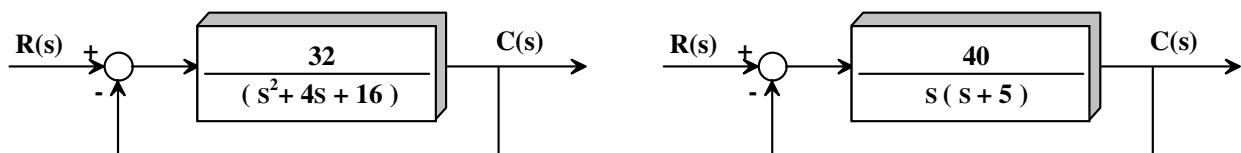
Caso haja interesse nas magnitudes e fases da função de transferência, o comando é usado na forma abaixo, com ou sem a especificação de w:

```
>>[mod, fase] = bode(num, den, w);
```

Na variável **mod**, são armazenadas as magnitudes da FT (não os valores em dB) e na variável **fase**, são armazenados os ângulos em graus. Nenhum gráfico é plotado.

7.3 Exercícios

Exercício 1 - Determine o coeficiente de erro apropriado para cada um dos sistemas abaixo através da

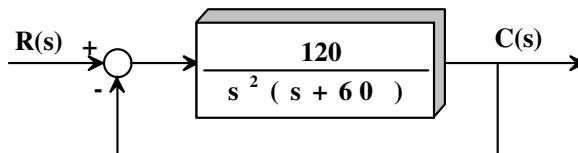


análise de seus diagramas de Bode e compare com os resultados teóricos.



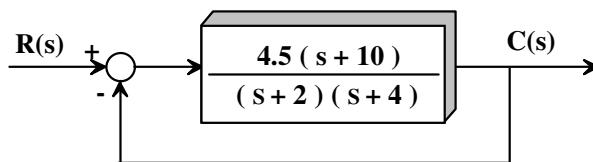
SISTEMA 3





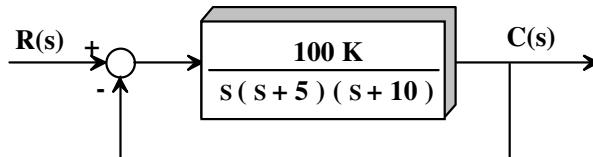
Exercício 2 - Seja o

sistema mostrado abaixo.



- Determine seu coeficiente de erro através do diagrama de Bode e diretamente a partir da FT;
- Considere a aplicação de uma entrada degrau unitário e calcule os erros absolutos e percentuais resultantes;
- Utilizando o Matlab ou Simulink, obtenha a resposta deste sistema para entrada degrau unitário e verifique se os erros absolutos e percentuais conferem com os calculados no item anterior.

Exercício 3 - Seja o sistema mostrado abaixo.



- Para $K = 1$, obtenha o seu diagrama de Bode em malha aberta e determine a freqüência de cruzamento de ganho - ω_g , a freqüência de cruzamento de fase - ω_f , a margem de ganho - K_g e K_g em dB e a margem de fase - γ .
- Utilize o comando mostrado abaixo para obter K_g e γ diretamente via Matlab. Compare esses valores com os obtidos anteriormente.

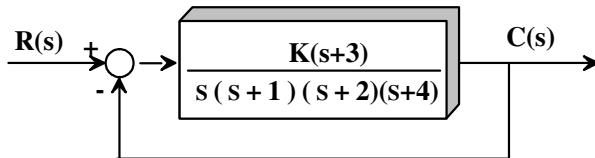
```
>> [Kg, Gama] = margin(num, den)
```

- Experimente, também, o comando

```
>> margin(num, den)
```

- Use o critério de estabilidade de Routh-Hurwitz para determinar a faixa de valores de K tal que o sistema seja estável. Compare com o valor obtido para K_g e interprete os resultados.
- Implemente o sistema acima no Matlab/Simulink para um ganho $K = K_g$, obtenha sua resposta em malha fechada para entrada degrau unitário, analise e comente o resultado.

Exercício 4 - Seja o sistema mostrado abaixo.



```
>> clf % Clear figure
>> num=[1 3]; den=conv([1 0], conv([1 2], [1 4]));
>>r=rlocus(num, den);
```

```
>>Plot(r,'-');v=[-6 6 -6 6];axis(v); axis('square')
>>grid on
>>title('Lugar geometrico das raizes de G(s)=\frac{K(s+3)...
{s(s+1)(s+2)(s+4)}');
>>xlabel('Eixo Real');ylabel('Eixo Imaginário');
>>gtext('o');
>>gtext('x');
>>gtext('x');
>>gtext('x');
>>gtext('x');
```