# SISTEMAS DINÂMICOS I

Usando a ferramenta MATLAB

EDUARDO L.L. CABRAL, LARISSA DRIEMEIER

Universidade de São Paulo

Escola Politécnica

# CONTEÚDO

# Parte I

# INTRODUÇÃO AO MATLAB

O MATLAB, do inglês *Matrix Laboratory*, é um ambiente de programação de alto desempenho voltado para a resolução de problemas que podem ser expressos em notação matemática.

# 1 INTRODUÇÃO

## 1.1 FERRAMENTA MATLAB

MATLAB é uma abreviação para MATrix LABoratory. O MATLAB, portanto, é um sistema interativo cujo elemento básico da informação é uma matriz que não requer dimensionamento. Trata-se de um ambiente de alto nível que possui ferramentas avançadas de análise numérica, cálculo com matrizes, processamento de sinais e visualização de dados. O MATLAB também possui características de linguagem de programação. O software possui funções matemáticas já existentes, programadas em linguagem própria e agrupadas de acordo com a área de interesse em *toolboxes*.

Assim, o MATLAB pode ser usado para:

- Cálculos matemáticos;
- Desenvolvimento de algoritmos;
- Modelagem, simulação e visualização de sistemas;
- Análise, exploração e visualização de dados;
- Gráficos científicos e de engenharia;
- Desenvolvimento de aplicações, incluindo a elaboração de interfaces gráficas com o usuário.

## 1.2 OBJETIVO

O nosso objetivo é aprender a utilizar o software MATLAB® para resolver alguns problemas comuns da área de dinâmica e controle dos sistemas.

Não é objetivo esgotar todos os temas e opções de uma ferramenta tão poderosa. Aliás, quando estiver com dificuldades no uso do software, tenha sempre em mente que existe uma grande chance deste seu problema já ter sido discutido nos inúmeros sites de ajuda e tutoriais disponíveis na internet, ou em livros didáticos - por exemplo, [4].

# 2 INICIANDO O MATLAB

Execute o MATLAB, clicando no ícone. A tela principal do programa é mostrada na Figura 1. A tela contém, em sua visualização padrão, uma janela de comandos, *Command Window*; uma janela para exibição da área de trabalho, *Workspace*, onde ficam armazenadas as variáveis definidas pelo usuário; e o *Command History*, ou histórico de comandos. A janela de comando fornece a principal forma de comunicação entre o usuário e o interpretador MATLAB, que exibe um sinal de prontidão, *prompt*, para indicar que está pronto para receber instruções.

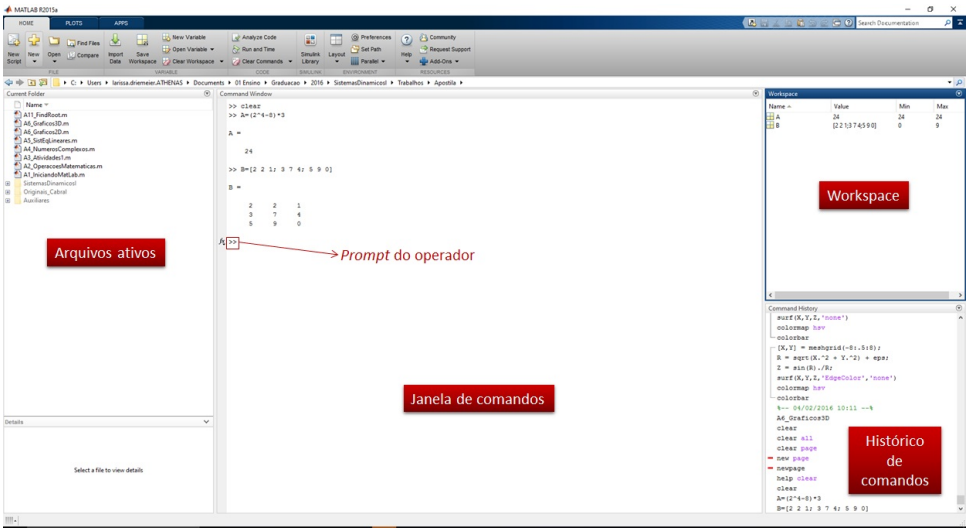



Figura 1: Tela principal do MATLAB2015a.

Antes de iniciar a sessão de trabalho é conveniente aumentar a fonte da letra usada na janela de comando. Clique em *Preferences* → *Fonts*, em seguida em *Desktop Code Font*, conforme Figura 2 e ajuste o tamanho da fonte para (no mínimo) 12 pontos. Acredite: muitos erros de digitação pode ser evitados com esta simples providência!

## 2.1 CONCEITOS BÁSICOS DA JANELA DE COMANDOS

Há diversas maneiras de se obter mais informações sobre uma função ou tópico do MATLAB. Se o nome da função for conhecido pode-se digitar, na janela de comando:

```
» help <nome da função>
```

Também é possível fazer uma busca por palavra-chave com o comando lookfor. Por exemplo, o comando:

```
» lookfor identity
```

retorna uma descrição curta de funções relativas a matrizes identidade. Além dessas formas, pode-se consultar a documentação do MATLAB clicando no ícone de ajuda da janela de comandos.

As funções abaixo também ajudam no controle da janela de comandos,

**clc,home:** limpa a janela de comandos;

**clear :** limpa da memória variáveis e funções;

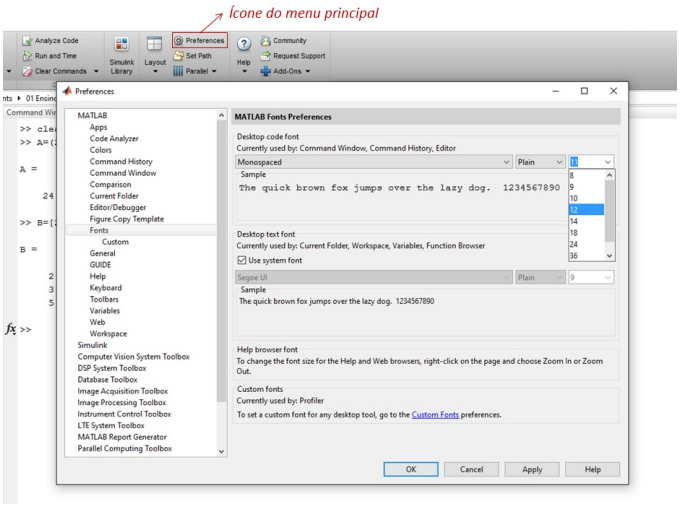



Figura 2: Ajuste da fonte usada na Janela de Comandos.

**computer:** retorna string contendo o computador que está executando MATLAB;

**delete :** apaga um arquivo ou um objeto gráfico;

**demo :** abre a janela *MATLAB examples*;

**ver :** mostra o número da versão do MATLAB e dos toolboxes instalados;

**version :** retorna a versão em uso do MATLAB;

**who :** lista o nome das variáveis armazenadas;

**whos :** lista as propriedades das variáveis atuais (nomes, dimensão, número de bytes e classe).

**load :** carrega variáveis armazenadas em arquivos .m;

## 2.2 COMANDOS E EXPRESSÕES

Para inserir algum comando, simplesmente digite na janela de comandos. Se você cometer algum erro, digite ENTER até aparecer o sinal *prompt* novamente e redigite o comando. O MATLAB guarda na memória seus comandos, portanto, para rever comandos anteriores use as teclas ↑ ou ↓. Para facilitar a repetição de comandos é possível também dar um duplo-clique nos itens da janela de histórico. Não se esqueça de que todo comando deve ser finalizado teclando ENTER.

Comece com a seguinte instrução,

```
» a=5/10
a =
      0.5000
```

Para você aprender uma operação diferente do MATLAB, digite a seguinte expressão e deduza a função do comando $< \backslash >$,

```
» a=5\10
```

A Tabela 1 mostra as principais operações aritméticas escalares do MATLAB.

O resultado de qualquer comando que não seja atribuído a uma variável específica é armazenado em uma variável padrão chamada *ans*. Exemplo:

| Símbolo | Operação | MATLAB |
|---|---|---|
| ^ | exponenciação $a^b$ | a^b |
| * | multiplicação ab | a * b |
| / | divisão à direita $a/b = \frac{a}{b}$ | a/b |
| \ | divisão à esquerda $a\backslash b = \frac{b}{a}$ | a\b |
| + | adição $a + b$ | a + b |
| − | subtração $a - b$ | a − b |

Tabela 1: Operações escalares. Tabela extraída de Palm III [4].

```
» 5/10
ans=
      0.5000
```

Quando for interessante omitir a exibição do resultado de qualquer comando basta encerrá-lo com ponto-e-vírgula. Exemplo:

```
1 >> a=5/10; % armazena a variavel mas nao exibe na tela
```

O símbolo de porcentagem serve para criar comentários de uma linha, tanto na janela de comandos quanto no ambiente de programação do MATLAB.

Quando se deseja continuar o comando na próxima linha, o sinal utilizado pelo MATLAB é representado por 3 pontos < ... >. Ou seja:

```
» a=10; b=20;
» c=a+...
» b
c =
     30
```

Vale informar que este comando não funciona para comentários.

O usuário pode interromper a execução do MATLAB, a qualquer momento, pressionando o `Crtl-c`.

## 2.3 STRINGS

O MATLAB define como *string* o conjunto de caracteres (vetor de caracteres) colocados entre aspas simples. Para acessar a variável é necessário definir a localização dos caracteres. Isto é,

```
» a='exemplo com string'
a =
       exemplo com string
» a(13:18)
ans =
       string
```

## 2.4 VARIÁVEIS

No MATLAB os nomes das variáveis devem ser palavras únicas, sem a inclusão de espaços e não devem conter acentos. O MATLAB é sensível à caixa, ou seja, diferencia letras maiúsculas de minúsculas e aloca automaticamente o espaço de memória necessário para as variáveis usadas. As três regras básicas para nomenclatura de variáveis são apresentadas na Tabela 2.

| | |
|---|---|
| As variáveis são sensíveis a letras maiúsculas e minúsculas | `VAR, Var, var` são três variáveis distintas. |
| As variáveis podem possuir até 31 caracteres - o excesso de caracteres será ignorado. | `EstouEntendendoTudoAteAgora` pode ser uma variável. |
| O nome da variável deve começar com uma letra, seguida de qualquer número, letra ou caracter de sublinhado. | `X51, X_51` podem ser uma variáveis. |

Tabela 2: Nomenclatura de variáveis.

Existem algumas variáveis especiais utilizadas pelo MATLAB, tais como `pi` (constante Pi),`i, j` (número imaginário $\sqrt{-1}$), `ans` (variável padrão para o último resultado), etc... Você pode redefinir estas variáveis, mas não convém...

## 2.5 FORMATOS NUMÉRICOS

O MATLAB mostra um resultado numérico em um determinado formato, conforme Tabela 3. O formato default de um número real é aquele definido como *format short*, com aproximação de quatro casas decimais. Se os dígitos significativos estiverem fora desta faixa, o MATLAB mostra o resultado em notação científica. Você pode definir um formato diferente. A forma de representação dos números internamente não muda, somente a exibição dos resultados.

| Formato | Resultado | Exemplo |
|---|---|---|
| format short | Ponto fixo; 4 casas decimais | 1.3333 |
| format short e | Not. científica; 4 casas decimais | 1.3333e+000 |
| format long | Ponto fixo; 14 casas decimais | 1.33333333333333 |
| format long e | Not. científica; 4 casas decimais | 1.333333333333333e+000 |
| format hex | Hexadecimal | 3ff5555555555555 |
| format rat | formato racional (aprox.), i.é, razão de inteiros | 4/3 |
| format bank | valor monetário (dólar e centavos); 2 casas decimais | 1.33 |
| format + | símbolos +,- e espaços em branco | + |

Tabela 3: Formatos numéricos.

## 2.6 FUNÇÕES COMUMENTE UTILIZADAS

A Tabela 4 e a Tabela 5 mostram, respectivamente, algumas funções trigonométricas e elementares comumente utilizadas.

| | | | |
|---|---|---|---|
| `sin(x)` | seno de x | `sinh(x)` | seno hiperbólico de x |
| `cos(x)` | coseno de x | `cosh(x)` | coseno hiperbólico de x |
| `tan(x)` | tangente de x | `tanh(x)` | tangente hiperbólica de x |
| `cot(x)` | cotangente de x | `coth(x)` | cotangente hiperbólica de x |
| `sec(x)` | secante de x | `sech(x)` | secante hiperbólica de x |
| `csc(x)` | cosecante de x | `csch(x)` | cosecante hiperbólica de x |
| `asin(x)` | arco cujo seno é x | `asinh(x)` | arco cujo seno hiperbólico é x |
| `acos(x)` | arco cujo cosseno é x | `acosh(x)` | arco cujo coseno hiperbólico é x |
| `atan(x)` | arco cuja tangente x | `atanh(x)` | arco cuja tangente hiperbólica é x |
| `acot(x)` | arco cuja cotangente x | `acoth(x)` | arco cujo cotangente hiperbólica é x |
| `acsc(x)` | arco cuja cosecante x | `acsch(x)` | arco cujo cosecante hiperbólica é x |
| `asec(x)` | arco cuja secante x | `asech(x)` | arco cujo secante hiperbólica é x |

Tabela 4: Funções trigonométricas.

| | |
|---|---|
| `abs(x)` | valor absoluto, ou seja, módulo de x |
| `exp(x)` | exponencial (base e) |
| `fix(x)` | arredonda para inteiro, em direção ao zero - p. ex., `fix(4.89)` $= 4$ |
| `floor(x)` | similar ao comando `fix` |
| `round(x)` | arredonda para o inteiro mais próximo - p. ex., `round(4.89)` $= 5$, `round(4.27)` $= 4$ |
| `ceil(x)` | arredonda para o próximo inteiro acima - p. ex., `ceil(4.27)` $= 5$ |
| `gcd(x,y)` | máximo divisor comum entre x e y |
| `lcm(x,y)` | mínimo múltiplo comum entre x e y |
| `log(x)` | logaritmo natural (base e) |
| `log10(x)` | logaritmo decimal (base 10) |
| `log2(x)` | logaritmo base 2 e desmembra número em ponto-flutuante |
| `rem(x,y)` | resto da divisão de x por y - p. ex., `rem(8,3)` $= 2$ |
| `sign(x)` | função sinal de x |
| `sqrt(x)` | raiz quadrada de x |

Tabela 5: Funções elementares.

## 2.7 EXERCÍCIOS

1. Calcule as seguintes expressões usando MATLAB,
   - `2/2*3`
   - `6-2/5+7^2-1`
   - `10/2\15-3+2*4`
   - `3^2/4`

- 3^2^2

2. Tente entender o significado dos comandos `round, floor e ceil,`
   - $\mathtt{round}(6/9+3*2)$
   - $\mathtt{floor}(6/9+3*2)$
   - $\mathtt{ceil}(6/9+3*2)$
   - $\mathtt{round}(1/9+3*2)$
   - $\mathtt{floor}(1/9+3*2)$
   - $\mathtt{ceil}(1/9+3*2)$

# 3 MANIPULAÇÃO DE MATRIZES

O tipo numérico padrão usado pelo MATLAB é a matriz de valores em ponto flutuante: números reais ou complexos são armazenados em matrizes 1x1. A maneira mais simples de se armazenar uma matriz na memória é com uma atribuição da seguinte forma:

```
» A = [2 1 3 4 5]
```

O resultado do comando anterior é mostrado na Figura 3. Note que o comando passou a fazer parte do histórico do programa e que a matriz foi armazenada na área de trabalho.

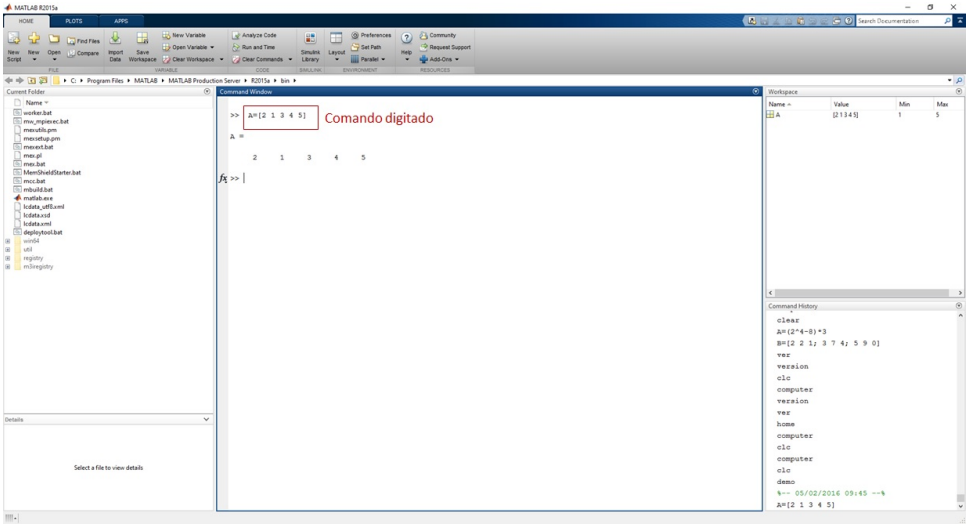



Figura 3: Atribuição de valores.

A alocação da matriz também pode ser confirmada pelos comandos `who` (mostra o nome das variáveis armazenadas) ou `whos` (mostra os nomes e espaços ocupados pelas variáveis), já mencionados,

```
» whos
Name      Size   Bytes   Class    Attributes
A         1x5    40      double
» who
Your variables are:
A
```

A matriz deste exemplo é chamada de vetor linha, já que se trata de uma matriz com apenas 1 linha. Na digitação, os valores do vetor podem ser separados por espaços, como no exemplo, ou por vírgulas. Para criar um vetor coluna deve-se separar cada linha das demais usando ponto-e-vírgula. Exemplo:

```
» B = [5; -4; 6.5]
B=
     5.0000
    -4.0000
     6.5000
```

Para criar uma matriz bidimensional basta combinar as sintaxes anteriores:

```
» M = [2 1 3; 4 6 7; 3 4 5]
M=
     2   1   3
     4   6   7
     3   4   5
```

Os elementos de uma matriz podem ser acessados pelo nome da variável, seguido de índices entre parênteses, sendo que o primeiro elemento é *sempre o de índice 1*. Exemplo de acesso:

```
» x=B(2)
x=
    -4
```

Se uma nova informação for atribuída a um vetor ou matriz os redimensionamentos necessários serão feitos automaticamente. Exemplo:

```
» A=[4 5 9];
» A(6)=8
A=
     4   5   9   0   0   8
```

Para acessar os elementos de uma matriz escreve-se o conjunto de índices entre parênteses, separados por vírgula. Exemplo:

```
» x=M(2,3)
x=
     7
```

Todos os elementos de uma certa dimensão de uma matriz podem ser indexados em conjunto, como no comando abaixo que cria um vetor linha com todos os elementos da segunda linha da matriz M.

```
» v1 = M(2,:)
v1 =
       4   6   7
```

Esse comando pode ser traduzido como *armazene em* v1 *os elementos de* M *que estão na linha 2 e em todas as colunas*. Da mesma forma, o comando a seguir cria um vetor coluna com os elementos da primeira coluna da matriz M:

```
» v2 = M(:,1)
v1 =
       2
       4
       3
```

O uso do sinal <:> também permite a atribuição de valores a uma dimensão completa de uma matriz. Por exemplo, a seguinte instrução sobre a matriz M atribui o valor 5 a todos elementos da primeira linha da matriz:

```
» M(1,:) = 5
M =
     5  5  5
     4  6  7
     3  4  5
```

O arranjo vazio, expresso por [] não contém nenhum elemento. É possível eliminar uma linha ou coluna de uma matriz igualando-se a linha ou coluna selecionada ao arranjo vazio. Por exemplo, a instrução a seguir remove a segunda linha da matriz M.

```
» M(2,:) = []
M =
     5  5  5
     3  4  5
```

Finalmente, matrizes podem ser concatenadas por meio de atribuições diretas. Exemplo:

```
M = [[5; 5; 5] v2 v1']
M =
     5  2  4
     5  4  6
     5  3  7
```

O termo $v1'$ significa *utilizar a transposta de v1*.

## 3.1 SEQUÊNCIAS

A definição de uma sequência de números no MATLAB é muito simples. O símbolo <:> serve para criar uma sequência igualmente espaçada de valores, entre dois limites especificados, inteiros ou não. Por exemplo, a instrução abaixo cria um vetor linha com os valores 3, 4, 5, 6, 7 e 8 (o incremento padrão é unitário).

```
» v3 = 3:8
v3 =
      3  4  5  6  7  8
```

O incremento pode ser definido pelo usuário se a seqüência for criada sob a forma `[Valor inicial: Incremento: Valor final]`. Exemplo:

```
» v4 = 2:0.5:4
v4 =
      2.0000  2.5000  3.0000  3.5000  4.0000
```

Outra forma de se obter um vetor com valores igualmente espaçados é pelo uso da função `linspace`. Por exemplo, a instrução abaixo gera um vetor linha com 25 valores igualmente espaçados entre 10 e 200. Se o parâmetro que controla o número de pontos for omitido, a seqüência terá 100 valores.

```
» y = linspace(10,200,25);
```

A diferença entre usar o operador <:> e a função `linspace` é que a primeira forma exige **o espaçamento entre os valores** enquanto a segunda requer **a quantidade de valores**.

Sequências com valores linearmente espaçados são usados, normalmente, para fornecer valores de variáveis independentes para funções. Por exemplo, as instruções a seguir criam um vetor com 50 valores da função $y = \sin(x)$, para $x \in [0, 2\pi]$:

```
» x = linspace(0,2*pi,50);
» y = sin(x);
```

'pi' é uma função interna do MATLAB que retorna o valor de $\pi$. Neste tipo de operação, chamada de vetorizada, o MATLAB cria o vetor $y$ com a mesma dimensão do vetor $x$.

Em situações que exijam grandes variações de valores pode ser interessante que a variação de valores seja logarítmica, o que pode ser obtido com o uso da função `logspace`, de sintaxe semelhante à de `linspace`. Por exemplo, a instrução abaixo cria um vetor com 50 valores espaçados logaritmicamente entre$10^0 = 1$ e $10^4 = 1000$.

```
» f = logspace(0,4,50);
```

## 3.2 OPERAÇÕES MATEMÁTICAS

O MATLAB reconhece os operadores matemáticos comuns à maioria das linguagens de programação nas operações com escalares (números reais e complexos). Nas operações com matrizes é preciso respeitar as regras da Matemática em relação às dimensões envolvidas. Por exemplo, considere:

```
» A= [6,3];
» B=[4,8];
» b=2;
```

A Tabela 6 resume algumas das principais operações elemento a elemento do MATLAB.

| Símb. | Operação | Exemplo | |
|---|---|---|---|
| + | Adição escalar-arranjo | A + b | $[6\ 3] + 2 = [8\ 5]$ |
| − | Subtração escalar-arranjo | A − b | $[6\ 3] - 2 = [4\ 1]$ |
| + | Adição de arranjos | A + B | $[6\ 3] + [4\ 8] = [10\ 11]$ |
| − | Subtração de arranjos | A − B | $[6\ 3] - [4\ 8] = [2\ -5]$ |
| .* | Multiplicação de arranjos | A.*B | $[6\ 3].*[4\ 8] = [6*4\ 3*8] = [24\ 24]$ |
| ./ | Divisão de arranjos à direita | A./B | $[6\ 3]./[4\ 8] = [6/4\ 3/8] = [1.5\ 0.375]$ |
| .\ | Divisão de arranjos à esquerda | A.\B | $[6\ 3].\backslash[4\ 8] = [4/6\ 8/5] = [0.667\ 2.667]$ |
| .^ | Exponenciação de arranjos | A.^b | $[6\ 3].\hat{}2 = [6^2\ 3^2] = [36\ 9]$ |
| | | b.^A | $2.\hat{}[6\ 3] = [2^6\ 2^3] = [64\ 8]$ |
| | | A.^B | $[6\ 3].\hat{}[4\ 8] == [6^4\ 3^8] = [1296\ 6561]$ |

Tabela 6: Operações elemento a elemento. Extraída de Palm III [4].

Considere as seguintes matrizes,

$$A = \begin{bmatrix} 1 & 4 & 5 \\ 2 & 3 & 9 \\ 6 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 5 & 5 \\ 4 & 6 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

Para as operações:

1. $C = A + B$ (Soma);
2. $C = A - B$ (Subtração)
3. $C = A * B$ (Multiplicação matricial)
4. $C = A. * B$ (Multiplicação elemento-a-elemento)
5. $C = A./B$ (Divisão elemento-a-elemento)
6. C = A.^2 (Potenciação elemento-a-elemento)

Verifique que, de acordo com as definições da matemática e as convenções do MATLAB, obtém-se os seguintes resultados:

$$1.C = \begin{bmatrix} 6 & 9 & 10 \\ 6 & 9 & 11 \\ 9 & 4 & 6 \end{bmatrix} \quad 2.C = \begin{bmatrix} -4 & -1 & 0 \\ -2 & -3 & 7 \\ 3 & -4 & -4 \end{bmatrix} \quad 3.C = \begin{bmatrix} 36 & 49 & 38 \\ 49 & 64 & 61 \\ 33 & 34 & 35 \end{bmatrix}$$

$$4.C = \begin{bmatrix} 5 & 20 & 25 \\ 8 & 18 & 18 \\ 18 & 0 & 5 \end{bmatrix} \quad 5.C = \begin{bmatrix} 0.2 & 0.8 & 1.0 \\ 0.5 & 0.5 & 4.5 \\ 2.0 & 0 & 0.2 \end{bmatrix} \quad 6.C = \begin{bmatrix} 1 & 16 & 25 \\ 4 & 9 & 81 \\ 36 & 0 & 1 \end{bmatrix}$$

Importante observar que a multiplicação de uma matriz $A(n \times k)$ por uma matriz $B(k \times m)$ produz uma matriz $n \times m$. Para as outras operações mostradas, as matrizes $A$ e $B$ devem ter as mesmas dimensões.

## 3.3 OUTRAS OPERAÇÕES COM MATRIZES

Para o vetor linha $v = \begin{bmatrix} 16 & 5 & 9 & 4 & 2 & 11 & 7 & 14 \end{bmatrix}$, verifique os seguintes comandos,

```
v(3)              % Extrai o terceiro elemento de v
v([1 5 6])        % Extrai o primeiro, quinto e sexto elemento de v
v2 = v([5:8 1:4]) % Extrai e inverte as duas metades de v
v(5:end)          % Extrai do quinto ao ultimo elemento de v
v([2 3 4]) = [10 15 20]   % Substitui alguns elementos de v
```

Ou ainda,

```
A = magic(4) %Quadrado magico 4x4: mesmo valor de soma dos ...
    elementos ao longo de qualquer linha, coluna ou da diagonal ...
    principal.
A(2,4)   % Extrai o elemento da linha 2 e quarta coluna 4
A(2:4,1:2) % Forma uma nova matriz com os elementos das linhas 2, 3 ...
    e 4 e colunas 1-2
A(14)   % usando um indice, MATLAB trata a matriz como se seus ...
    elementos fossem alinhados em um vetor coluna
```

```
idx = sub2ind(size(A), [2 3 4], [1 2 4]) %sub2ind converte os ...
    indices (linha,coluna) em um indice linear
A(idx) %valores de A correspondentes aos indices lineares ...
    encontrados anteriormente
```

Há uma série de funções disponíveis no MATLAB para geração ou alteração de matrizes e vetores, exemplificadas na Tabela 7. Além disso, a Tabela 8 fornece algumas matrizes especiais existentes no MATLAB.

| Função | Operação |
|---|---|
| `x=max(A);` | Retorna o maior componente de $A$. Se $A$ for uma matriz, o resultado é um vetor linha contendo o maior elemento de cada coluna. Para vetores, o resultado é o maior valor (ou o número complexo com maior módulo) entre seus componentes. |
| `x=min(A);` | Semelhante ao `max(A)`, mas retorna os valores mínimos. |
| `[vmax imax] = max(v);` | Para o vetor $v$, retorna o maior elemento em $vmax$ e o índice correspondente em $imax$. |
| `x = size(A);` | Retorna as dimensões da matriz $A$ em um vetor linha, $x = [m, n]$, contendo o número de linhas ($m$) e colunas ($n$) da matriz. |
| `[m n] = size(A);` | Retorna o número de linhas e colunas em variáveis separadas. |
| `x = length(A);` | Retorna o comprimento do vetor A ou o comprimento da maior dimensão da matriz $A$. Nesse último caso, `length(A) = max(size(A))` |
| `x = det(A);` | Retorna o determinante da matriz quadrada $A$. |
| `d=eig(A);` | Determina autovalores de $A$. |
| `[V,D]=eig(A);` | Determina autovalores e autovetores de $A$. |
| `inv(A);` | Calcula a inversa de $A$. |
| `poly(A);` | Calcula a equação característica de $A$. |
| `norm(x);` | Retorna o comprimento geométrico do vetor $\sqrt{x_1^2 + x_2^2 + ... + x_n^2}$. |
| `sort(A);` | Rearranja em ordem crescente cada coluna de $A$ e retorna uma matriz com as mesmas dimensões de $A$. |
| `sum(A);` | Soma os elementos de cada coluna de $A$ e retorna um vetor linha que contém o resultado das somas. |

Tabela 7: Outras operações.

Vale a pena ainda mencionar o comando `find` para obtenção de elementos de uma matriz ou vetor que obedecem uma determinada condição,

```
» x = find(expressão);
```

| Função | Operação |
|---|---|
| `x = eye(n);` | Retorna uma matriz identidade com dimensão $n \times n$, isto é, com valores unitários na diagonal principal e nulos nas demais posições. |
| `x = zeros(n);` | Cria uma matriz quadrada com dimensão $n \times n$ de elementos nulos. |
| `x = zeros(m,n);` | Cria uma matriz retangular com dimensão $m \times n$ de elementos nulos. |
| `x = ones(n);` | Semelhante ao comando zeros, gera matrizes com valores unitários (preenchidas com 1's). |
| `rand(n);` | Cria uma matriz quadrada $n \times n$ de elementos aleatórios distribuídos entre 0 e 1. |
| `randn(n);` | Cria uma matriz quadrada $n \times n$ de elementos aleatórios que seguem uma distribuição normal, com média 0 e variância 1. |

Tabela 8: Matrizes especiais do MATLAB.

Encontra e retorna todos os elementos de um vetor ou matriz que satisfazem a uma certa expressão lógica. Normalmente, usam-se argumentos à esquerda da instrução de busca para armazenar os índices dos elementos de interesse. Exemplo:

```
» A = [3 -2 1; 0 2 -1; 4 1 2];
» [L C] = find(A>2 & A<=4)
L =
     1
     3
C=
     1
     1
```

Nesse exemplo apenas os elementos $A(1,1)$ e $A(3,1)$ atendem ao critério desejado. Percebe-se que L é o vetor contendo a localização na linha e C na coluna. As expressões válidas podem incluir operadores relacionais e lógicos, resumidos na Tabela 9.

Esses operadores se aplicam a escalares e matrizes, de acordo com regras que podem ser consultadas na documentação do MATLAB. Se o argumento da função for apenas o nome de uma matriz ou vetor, serão retornados os índices de todos os elementos da matriz ou vetor que forem diferentes de zero.

Finalmente, os comandos `all` e `any` analisa o número de valores nulos de cada coluna de uma matriz. Isto é, o comando `all` retorna 1 para cada coluna da matriz A que contenha somente valores não nulos e 0 em caso contrário, gerando um vetor linha. Por exemplo,

```
» A = [3 -2 1; 0 2 -1; 4 1 2];
» x = all(A)
x=
    0   1   1
```

| Operadores relacionais | |
|---|---|
| < | Menor que |
| <= | Menor ou igual |
| > | Maior que |
| >= | Maior ou igual |
| == | Igual a |
| ~ = | Diferente de |
| **Operadores lógicos** | |
| & | operação *E* |
| \| | operação *OU* |
| ~ | negação lógica |

Tabela 9: Operadores relacionais e lógicos.

Para vetores, a função retorna 1 se todos os elementos forem não nulos e 0 em caso contrário. A função `any` retorna 1 para cada coluna da matriz A que contenha algum valor não nulo e 0 em caso contrário, gerando um vetor linha. A função também trabalha com vetores. Exemplo:

```
» x = any(A)
x=
     1   1   1
```

## 3.4 EXERCÍCIOS

Antes de iniciar qualquer nova atividade no MATLAB é recomendável limpar a janela de comando digitando `clc`. Em seguida, deve-se remover todas as variáveis da memória, usando o comando `clear`. Se quiser eliminar apenas uma variável, deve-se usar a sintaxe `clear <nome da variável>`.

1. Treine (Extraído de Braun [1]):
   - $g = \begin{bmatrix} 1; & 2; & 3; & 4 \end{bmatrix}$
   - $g = \begin{bmatrix} 1 & 2 & 3 & 4; & 5 & 6 & 7 & 8 \end{bmatrix}$
   - $g - 2$
   - $2 * g - 1$
   - $g = \begin{bmatrix} 1 & 2 & 3 & 4; & 5 & 6 & 7 & 8; & 9 & 10 & 11 & 12 \end{bmatrix}$
   - $h = \begin{bmatrix} 1 & 1 & 1 & 1; & 2 & 2 & 2 & 2; & 3 & 3 & 3 & 3 \end{bmatrix}$
   - $g - h$
   - $g * h$
   - $h * g$
   - $g. * h$
   - $g./h$
   - $h.\backslash g$
   - $g * h'$

- $g' * h$
- $e = \begin{bmatrix} 1 & 2 & 3; & 4 & 5 & 6; & 7 & 8 & 0 \end{bmatrix}$
- $f = \begin{bmatrix} 9 & 8 & 7; & 6 & 5 & 4; & 3 & 2 & 1 \end{bmatrix}$
- $e * f$
- $f * e$
- $q = rand(3)$
- A^2
- A^2 − A ∗ A
- A.^2
- A.^2 − A ∗ A

2. Armazene as seguintes matrizes

$$M1 = \begin{bmatrix} 2 & -1 & 5 \\ 3 & 1 & 1 \\ 2 & 0 & 2 \end{bmatrix} \qquad M2 = \begin{bmatrix} 1 & -2 & 3 \\ 0 & 5 & 2 \\ -1 & 0 & 4 \end{bmatrix}$$

3. Calcule as seguintes operações:
   - M1 + M2;
   - M1–M2;
   - M1 ∗ M2;
   - M1. ∗ M2;
   - M1.\M2;
   - M1.^2;
4. Obtenha a matriz transposta de M1
5. Obtenha a matriz inversa de M2
6. Com uma linha de comando, calcule a multiplicação entre M1 e a inversa de M2
7. Com uma linha de comando, calcule a multiplicação entre M1 e a transposta de M2
8. Gere uma matriz de dimensão $5 \times 5$ com a diagonal igual a 3 e todos os outros elementos iguais a zero.
9. Obtenha os índices e os valores dos elementos da matriz M1 que contém valores menores do que zero.
10. Obtenha os índices e os valores dos elementos da matriz M2 que sejam maiores ou iguais a -2 e menores ou iguais a 2.
11. (Extraído de Palm III [4]) Escreva **uma** sentença de atribuição no MATLAB para cada uma das seguintes funções, considerando que $w, x, y, z$ são vetores linha de mesma dimensão e $c, d$ são escalares:
    - $f = \frac{1}{\sqrt{2\pi c/x}}$
    - $E = \frac{x+w/(y+z)}{x+w/(y-z)}$;

- $A = \frac{e^{-c/(2x)}}{(\ln y)\sqrt{dz}}$;
- $S = \frac{x(2.15+0.35y)^{1.8}}{z(1-x)^{y}}$;

12. Declare a matriz $A = \begin{bmatrix} 2 & 10 & 7 & 6 \\ 3 & 12 & 25 & 9] \end{bmatrix}$ e:
    - Altere o elemento $A(2,1)$ para 18.
    - Acrescente uma terceira linha a matriz com os elementos 30 21 19 1.
    - Defina o elemento $A(2,8)$ como $-16$
    - Defina uma matriz B que contenha as três primeira linhas da matriz A e as colunas de 2 a 5.
13. Dada a matriz:

$$A = \begin{bmatrix} 3 & 7 & -4 & 12 \\ -5 & 9 & 10 & 2 \\ 6 & 13 & 8 & 11 \\ 15 & 5 & 4 & 1 \end{bmatrix}$$

    - Ordene cada coluna e armazene o resultado em A1.
    - Ordene cada linha e armazene o resultado em A2.
    - Some cada coluna e armazene o resultado em b1.
    - Some cada linha e armazene o resultado em b2.
    - Avaliar o valor máximo no vetor resultante da multiplicação elemento a elemento da segunda coluna de A2 pela primeira coluna de A.
    - Utilizar a divisão elemento a elemento para dividir a primeira linha de A pela soma dos três primeiros elementos da terceira coluna de A1.

# Parte II

# GRÁFICOS E PROGRAMAÇÃO

Aqui são apresentados os comandos básicos empregados na plotagem de gráficos bi- e tri- dimensionais e noções sobre programação. Ressalta-se que esta parte da apostila tem apenas o suficiente para você iniciar o uso das poderosas ferramentas gráficas e de programação disponíveis no MATLAB.

# 4 GRÁFICOS

O MATLAB possui diversas ferramentas para traçados de gráficos bidimensionais ou tridimensionais.

## 4.1 GRÁFICOS BIDIMENSIONAIS

A maneira mais simples de traçar um gráfico xy é pelo uso da função `plot`. A forma `plot(x,y)` desenha um gráfico bidimensional dos pontos do vetor $y$ em relação aos pontos do vetor $x$, sendo que ambos devem ter o mesmo número de elementos. O gráfico resultante é desenhado em uma janela de figura com as escalas automáticas nos eixos $x$ e $y$ e segmentos de reta unindo os pontos. Por exemplo, para desenhar o gráfico da função:

$$y = 1 - 1.1547e^{-1.5x}\,\sin(2.5981x + 1.0472)$$

no intervalo $x \in [0, 10]$, pode-se utilizar a seguinte seqüência de comandos:

```
x = 0:0.1:10;
y = 1-1.1547*exp(-1.5*x).*sin(2.5981*x+1.0472);
plot(x,y)
```

O resultado, apresentado na Figura 4, é exibido em uma janela de figura identificada por um número. O mesmo resultado é obtido com a função `fplot`, basta identificar a string a ser representada no domínio do gráfico,

```
y='1-1.1547*exp(-1.5*x).*sin(2.5981*x+1.0472)';
fplot(y,[0 10]
```

Outra função utilizada para confeccionar gráficos bidimensionais é a função `ezplot`. Esta função também tem como argumentos de entrada uma função string e um intervalo de variação. Se $f = f(x, y)$, o comando `ezplot` representa a função considerando $f(x, y) = 0$,

```
ezplot('x^2-3*y^2+2*x*y',[0 100])
```

Em algumas ocasiões é interessante que as escalas dos eixos sejam representadas em escala logarítmica (ao invés da escala linear padrão). Nestes casos, é possível usar as funções `semilogx`, `semilogy` ou `loglog`, que alteram, respectivamente, a escala do eixo $x$, do eixo $y$ e de ambos. Normalmente os valores que compõem tais gráficos também são gerados com espaçamentos logarítmicos, via função `logspace`.

A função `plot` pode trabalhar com várias duplas de vetores, sobrepondo mais de um gráfico em uma mesma janela. O resultado da seqüência de comandos a seguir está representado na Figura 5.

```
x=-1:0.1:1;  % Cria vetor 'x': valores entre 1 e -1 espacados de 0.1
y=x.^2;      % Calcula y
z=x.^3;      % Calcula Z
plot(x,y,'r*',x,z,'b:') % Traca os dois graficos - x vs y e x vs z
xlabel('Valor de x')        % Nomeia o eixo x
ylabel('y e z')     % Nomeia o eixo y
```

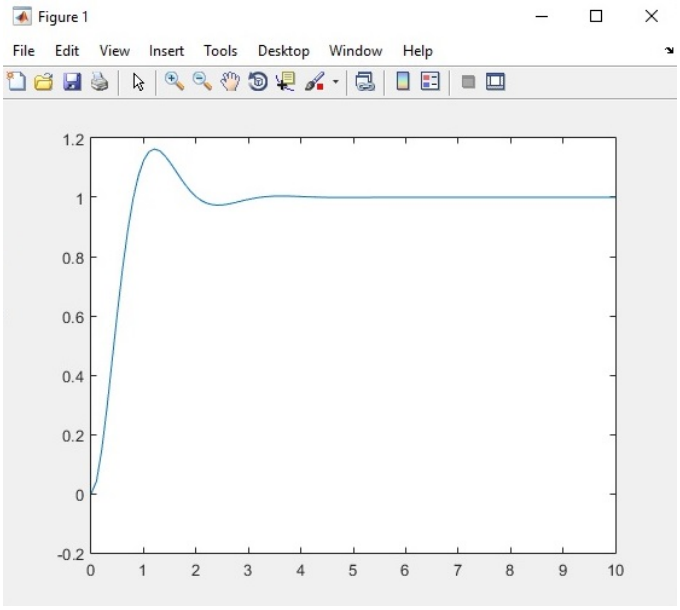



Figura 4: Exemplo de resultado gráfico da função `plot`.

```
title('Graficos sobrepostos')   % Atribui um titulo ao grafico
grid    % Ativa as linhas de grade da janela
```

Note que foram usadas funções para nomear os eixos (`xlabel` e `ylabel`) e o título do gráfico (`title`), além de exibição de linhas de grade (`grid`). O estilo e cor da linha estão definidos no comando `plot`. A Tabela 10 mostra as configurações de cores, marcadores e linhas, opções essas válidas para plotar gráficos no MATLAB, em 2D e 3D.

| Cores de linhas | | Marcadores de ponto | | Tipos de linhas | |
|---|---|---|---|---|---|
| Símb. | Cor | Símb. | Marcador | Símb. | Tipo |
| y | amarelo | . | ponto | − | sólida |
| m | magenta | o | círculo | : | pontilhada |
| c | azul-claro | x | x | −. | traço-ponto |
| r | vermelho | + | + | −− | tracejada |
| g | verde | * | asterisco | | |
| b | azul escuro | s | quadrado | | |
| w | branco | d | losango | | |
| k | preto | v | triângulo | | |
| | | ^ | triângulo | | |
| | | < | triângulo para esquerda | | |
| | | > | triângulo para direita | | |
| | | p | pentagrama | | |
| | | h | hexagrama | | |

Tabela 10: Códigos para cores, marcadores e tipos de linha em gráficos no MATLAB.

Outra forma de se obter gráficos sobrepostos é com o uso da função `hold`, que faz com que todos os resultados gráficos subsequentes ao seu uso sejam dese-

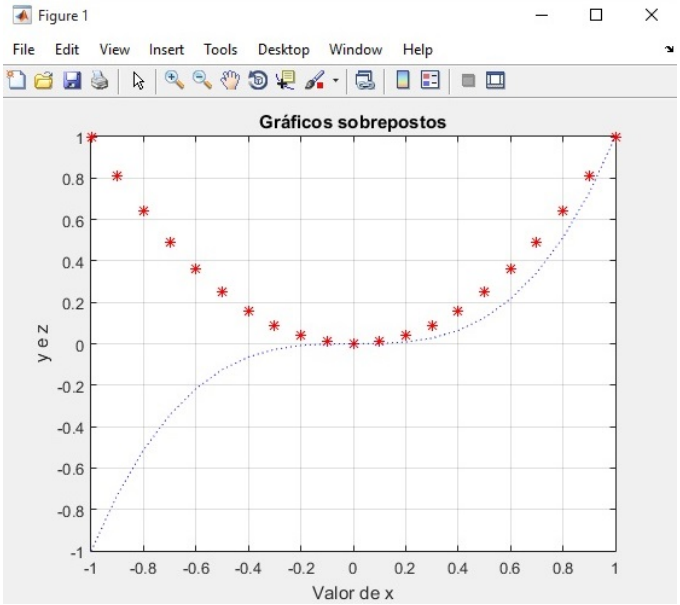



Figura 5: Gráfico com duas funções superpostas.

nhados em uma mesma janela de figura. Exemplo (considerando as variáveis do exemplo anterior):

```
plot(x,y);  % Desenha o grafico de uma funcao
hold on     % Ativa a 'trava' de exibicao grafica
plot(x,z);  % Desenha outro grafico na mesma janela de figura
hold off        % Desativa a 'trava' de exibicao grafica
```

Para que os gráficos sejam plotados no mesmo gráfico, mas um por um, usa-se a sequência de comandos `hold on` e `pause`,

```
x=linspace(0,2*pi,100);
y=sin(x);
z=0.5*sin(3*x);
plot(x,y,'r*')
pause % pausa ate ser pressionada uma tecla
hold on % Mantem o grafico atual
plot(x,z,'b:')
pause
close
```

O comando `subplot(nl,nc,ng)` pode ser usado para plotar mais de um gráfico na mesma janela; `nl`, `nc` são, respectivamente, o número de linhas e o número de colunas de gráficos; e `ng` é o número do gráfico em questão. Por exemplo, a sequência de comandos abaixo gera a Figura 6.

```
K = [1:100].^2;
Y = K.^(-0.4);
subplot(3,1,1);
plot(K, Y);
grid on
subplot(3,1,2);
semilogx(K, Y);
grid on
subplot(3,1,3);
loglog(K, Y);
grid on
```

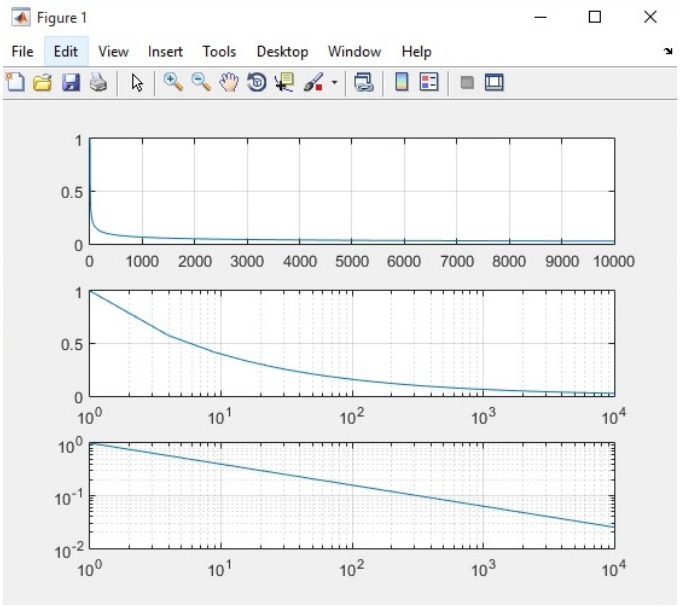



Figura 6: Função `subplot`.

Todos os resultados gráficos aparecem na janela de figura ativa. Uma nova janela pode ser criada ou ativada pelo comando `figure`. Quando usada sem argumentos, esta função cria uma janela de título *Figure No. xx*, sendo xx um número sequencial, considerado disponível pelo MATLAB. O uso do comando `figure(n)` cria a janela de figura n, se necessário, e a torna ativa. Outros recursos da função `plot` podem ser consultados na documentação do MATLAB.

A escala dos eixos é feita automaticamente. No entanto, esta pode ser alterada através do comando `axis([xmin,xmax,ymin,ymax])`, que gera os eixos nos limites especificados.

Quando os argumentos para plotar são complexos, a parte imaginária é ignorada, exceto quando é dado simplesmente um argumento complexo. Para este caso especial é plotada a parte real versos a parte imaginária. Então, `plot(Z)`, quando Z é um vetor complexo, é equivalente a `plot(real(Z),imag(Z))`.

## 4.2 GRÁFICOS TRIDIMENSIONAIS

Gráficos em três dimensões podem ser traçados pelo MATLAB com a mesma facilidade que os bidimensionais. A função `plot3` funciona de forma semelhante à função `plot` para o traçado de gráficos de linha em 3D. Por exemplo, a sequência de comandos a seguir produz um gráfico de uma hélice tridimensional. Note o uso da função `zlabel` para nomear o eixo z do gráfico. O resultado está representado na Figura 7.

```
t = linspace(0,6*pi,100);
plot3(sin(t),cos(t),t);
xlabel('seno(t)');
ylabel('cosseno(t)');
zlabel('z = t');
title('Grafico de helice');
grid on;
```

O MATLAB também pode construir gráficos de superfícies a partir de um conjunto de coordenadas tridimensionais xyz. Inicialmente, é preciso gerar matrizes X e Y com, respectivamente, linhas e colunas preenchidas com os valores das va-

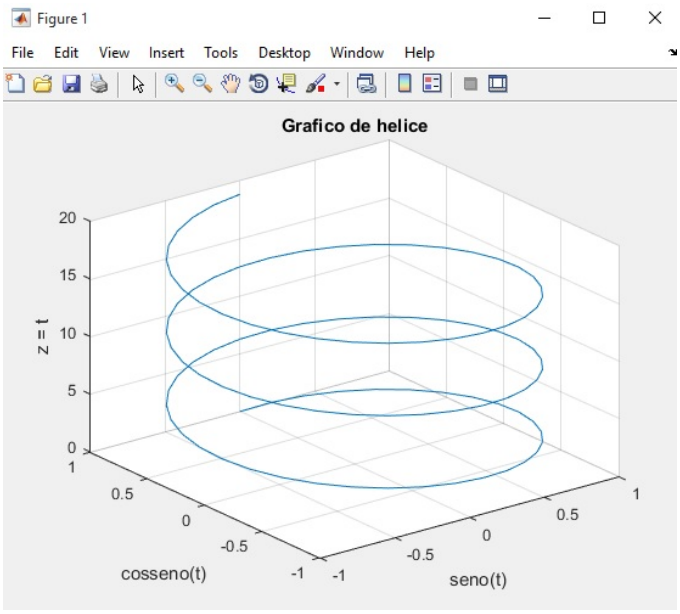



Figura 7: Gráfico de linha tridimensional.

riáveis x e y. Isto pode ser feito diretamente pela função `meshgrid` (omita o <;> das duas últimas linhas de comando para entender a lógica da função),

```
x = linspace(0,2,3);  % Geracao de valores para 'x' e 'y',
y = linspace(3,5,2);  % ambos com a mesma dimensao!
[X,Y] = meshgrid(x,y) % Criacao da matriz da malha 'xy'
z=X.*Y
```

A partir dessas matrizes, que representam uma grade retangular de pontos no plano xy, qualquer função de duas variáveis pode ser calculada em uma matriz Z e desenhada pelo comando `mesh`. Exemplo para o gráfico de um parabolóide elíptico (Figura 8):

```
x = -5:0.5:5;        % Definicao da malha de pontos no eixo 'x'
y = x;       % Repeticao da malha do eixo x para o eixo 'y'
[X,Y] = meshgrid(x,y);  % Criacao da matriz da malha 'xy'
Z = X.^ 2 + Y.^ 2;      % Calculo da funcao z = f(x,y)
mesh(X,Y,Z)     % Tracado do grafico da funcao 'z'
```

Verifica-se que a malha gerada é colorida. O usuário pode escolher um dos mapas de cores existente no MATLAB (veja o próximo exemplo). Porém, se essa matriz for omitida, como no exemplo anterior, as cores das linhas serão relacionadas com a altura da malha sobre o plano xy.

A função `mesh` cria uma malha tridimensional em que cada ponto é unido por segmentos de reta aos vizinhos na malha. Usando a função `surf` é possível gerar um gráfico de superfície em que os espaços entre os segmentos são coloridos, conforme o mapa de cores definido pela função `colormap`. O código `hsv` da listagem abaixo refere-se a um dos mapas de cores disponíveis no MATLAB apresentados na Tabela 11.

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
surf(X,Y,Z)
colormap hsv  % define o mapa de cores
```

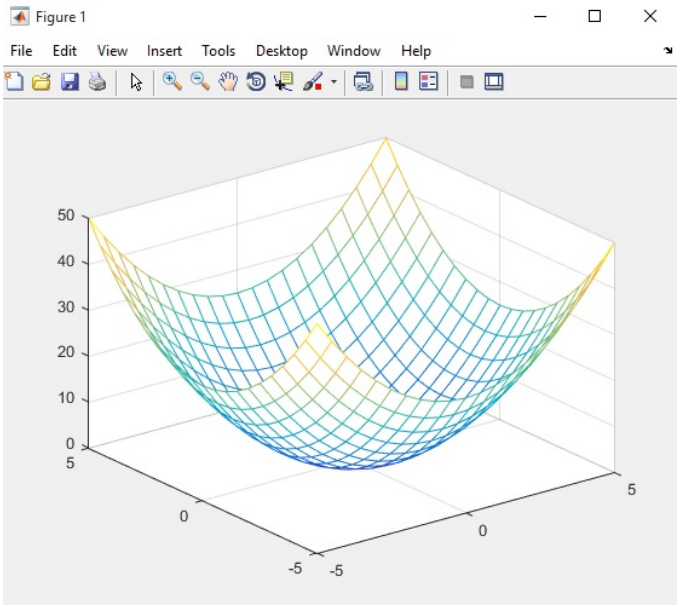



Figura 8: Parabolóide elíptico gerado com a função `mesh`.

```
colorbar     % para colocar a barra de cores
```

| Função | Mapa de cores |
|---|---|
| Hsv | Escalar com cores saturadas |
| Hot | Preto-vermelho-amarelo-branco |
| Gray | Escalar linear de tons de cinza |
| Bone | Escala de tons de cinza levemente azulados |
| Copper | Escala linear de tons acobreados |
| Pink | Tons pastéis de rosa |
| White | Mapa de cores totalmente branco |
| Flag | Vermelho, branco, azul e preto alternados |
| Jet | Uma variante do mapa hsv |
| Prism | Mapa de cores denominado *prisma* |
| Cool | Tons de ciano e magenta. |
| lines | Mapa de cores que usa as mesmas cores do comando `plot` |
| colorcube | Mapa de cores denominado *cubo colorido* |
| summer | Tons de amarelo e verde |
| autumn | Tons de vermelho e amarelo |
| winter | Tons de azul e verde |
| spring | Tons de magenta e amarelo |

Tabela 11: Mapas de cores utilizados pelo MATLAB.

A Figura 10 mostra o uso de diferentes mapas de cores e, aproveitando, mostra também a opção de como eliminar as linhas de grade da superfície. A Figura foi gerada através dos comandos,

```
x=-3:0.1:3; y=x; [X,Y]=meshgrid(x,y); Z=X.^ 3 + Y.^3 - 5*X.*Y + 1/5;
surf(X,Y,Z), colormap([summer])
pause
surf(X,Y,Z,'EdgeColor','none'), colormap([winter])
```

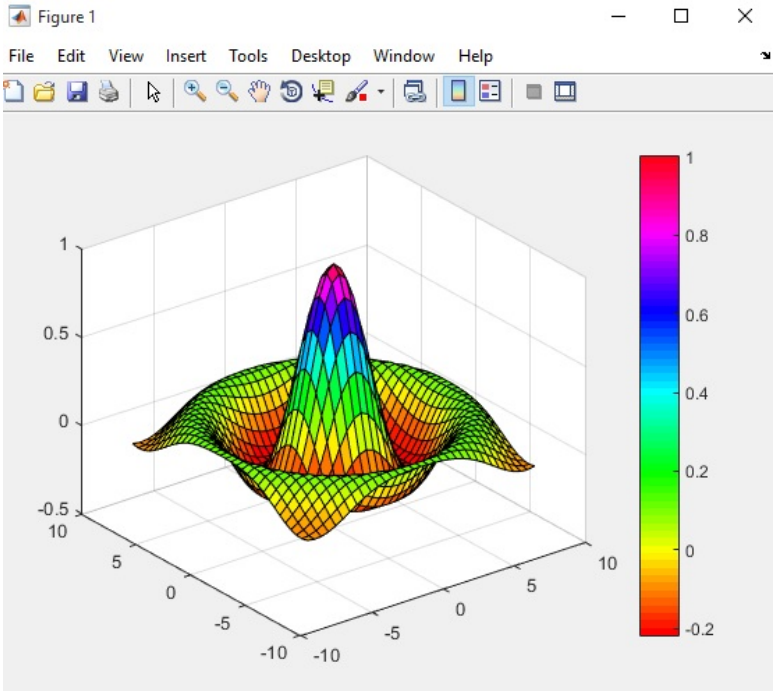
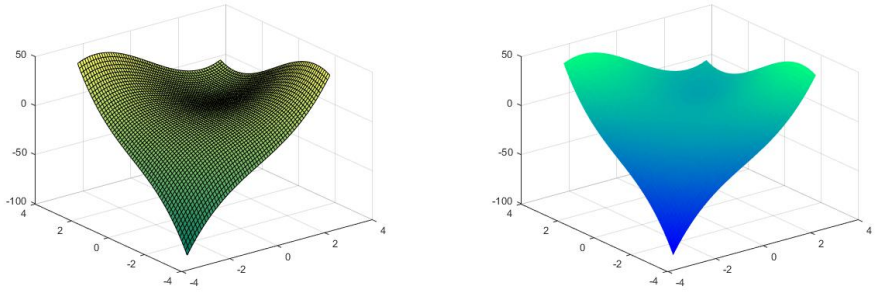



Figura 9: Superfície gerada pela função `surf`.



(a) *Summer.* (b) *Winter, sem linhas.*

Figura 10: Exemplo de uso do mapa de cores.

## 4.3 EXERCÍCIOS

1. Plote a função 2D $\sin(\pi/x)$, para $x = -1..1$
2. Veja o exemplo abaixo:

```
» A = [1,2,3,4,5; 1,0,1,2,1; 2,1,0,1,2].'
A =
     1   1   2
     2   0   1
     3   1   0
     4   2   1
     5   1   2
» plot(A(:,1), A(:,2:end))
```

A Tabela 12 mostra a variáveis temperatura e vento, em função da hora. Crie uma variável similar à A do exemplo e faça o que é pedido abaixo.

| Horas | Temperatura | Vento |
|---|---|---|
| 0 | 9 | 12 |
| 2 | 8 | 13 |
| 4 | 6 | 14 |
| 6 | 6 | 15 |
| 8 | 8 | 17 |
| 10 | 10 | 13 |
| 12 | 14 | 19 |
| 14 | 17 | 11 |
| 16 | 15 | 7 |
| 18 | 13 | 8 |
| 20 | 11 | 7 |
| 22 | 10 | 14 |

Tabela 12: Valores da temperatura do ar e velocidade do vento.

Plote,

- somente a variável temperatura;
- utilize o comando `area` para preencher a área sob a curva do itm anterior;
- plote as duas variáveis no mesmo gráfico, mas em cores diferentes;
- plote as duas variáveis no mesmo gráfico, mas em espessura de linha diferente;
- plote as duas variáveis no mesmo gráfico, somente com marcadores;

3. Plote, para uma malha quadrada adequada $[x, y]$, a função $z = xe^{(-x^2-y^2)}$. Coloque legenda.
   - Faça duas malhas: uma pouco e outra bastante refinada.
   - Use o comando `surf(x,y,z,gradient(z))` para que a opção `gradient(z)` determine a distribuição de cores.
   - Com ajuda da função `view(azimute, elevacao)`, gere uma figura com 4 gráficos (comando `subplot`), com as seguintes combinações de vista: `view(0,0), view(0,90), view(90,0), view(45,30)` . Em caso de dúvidas, consulte:

http://www.mathworks.com/help/matlab/visualize/setting-the-viewpoint-with-azimuth-and-elevation.html.

4. A Figura 11 ilustra as forças atuantes no avião em linha reta, em nível e com velocidade constante. A força de sustentação é da ordem de 10-20 vezes a força de arrasto; o CG está sempre a frente do ponto de aplicação da sustentação, para que a aeronave possa ser controlada (a força estabilizadora é pequena, mas controla o momento resultante do deslocamento do CG); o arrasto é distrubuído por toda a superfície do avião.

   As partes do avião destinadas à sustentação (asa, leme, estabilizadores, hélice, ...) são chamadas genericamente de aerofólio. Aerofólio tem seção transversal típica, achatada e alongada, conforme Figura 11b.

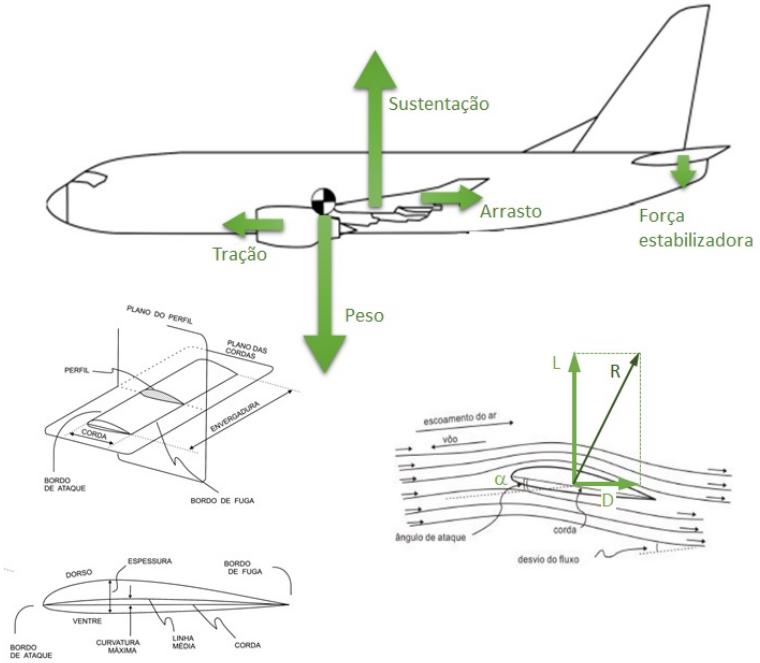



Figura 11: Forças atuantes em um avião e detalhamento do aerofólio.

Para facilitar o estudo das forças de um aerofólio, a resultante aerodinâmica é dividida em duas componentes:

- Sustentação (L de *Lift*): é a componente da resultante aerodinâmica perpendicular à direção do vento relativo. Esta é a força útil do aerofólio.
- Arrasto (D de *Drag*): é a componente da resultante aerodinâmica paralela à direção do vento. É geralmente nociva e deve ser reduzida ao mínimo possível. Essa força depende de alguns fatores como a forma do corpo, a sua rugosidade e o efeito induzido resultante da diferença de pressão entre a parte inferior e superior da asa.

As seguintes fórmulas são comumente utilizadas para estimar as forças de sustentação, L, e arrasto, D, de um aerofólio:

$$L = \frac{1}{2}\rho C_L S V^2$$
$$D = \frac{1}{2}\rho C_D S V^2$$

em que V é a velocidade do ar, S é a área de referência, $\rho$ é a densidade do ar, e $C_L$ e $C_D$ são, respectivamente, os coeficientes de sustentação e arrasto.

Cada perfil de aerofólio apresenta uma curva característica de $C_L$ quanto $C_D$ contra o ângulo de ataque $\alpha$.

[Extraído de Palm III [4]] Experimentos em túnel de vento para um aerofólio em particular resultaram nas seguintes fórmulas:

$$C_L = 4.47 \times 10^{-5}\alpha^3 + 1.15 \times 10^{-3}\alpha^2 + 6.66 \times 10^{-2}\alpha + 1.02 \times 10^{-1}$$
$$C_D = 5.75 \times 10^{-6}\alpha^3 + 5.09 \times 10^{-4}\alpha^2 + 1.8 \times 10^{-4}\alpha + 1.25 \times 10^{-2}$$

para $\alpha$ expresso em graus.

Dessa forma,

- Plote as forças de sustentação e arrasto desse aerofólio versus V para $0 \leqslant V \leqslant 300\text{Km/h}$, para uma área de $30\text{m}^2$ com um ângulo de ataque de $4^o$, massa específica do ar nas condições de pressão e temperatura de vôo de aproximadamente $1\ \text{kg/m}^3$.
- A razão sustentação-arrasto (L/D = CL/CD) é uma indicação da eficácia de um aerofólio. Plote L/D versus $\alpha$ para $-2^o \leqslant \alpha \leqslant 22^o$. Determine o ângulo de ataque que maximiza a eficiência.

5. Dado o amortecedor a êmbulo mostrado na Figura 12, a seguinte expressão para a constante c de amortecimento pode ser deduzida [6],

$$c = \mu\left[\frac{3\pi D^3 l}{4d^3}\left(1 + \frac{2d}{D}\right)\right]$$

onde $D, l$ são o diâmetro e comprimento do pistão, respectivamente, que se move a uma velocidade $v_o$ dentro de um cilindro cheio de um líquido de viscosidade $\mu$; d é a folga entre o pistão e a parede do cilindro.

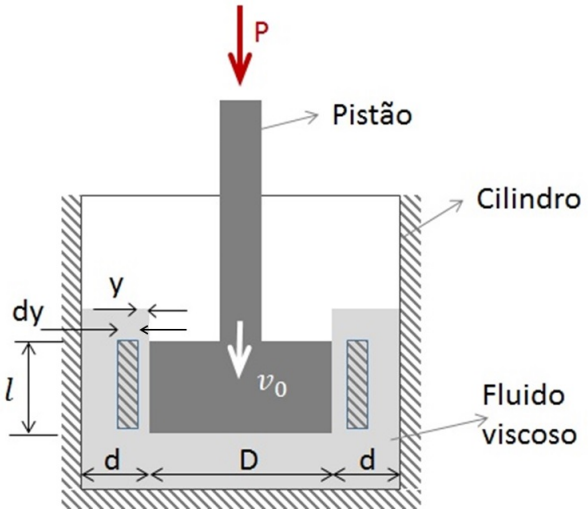



Figura 12: Amordetecedor de êmbulo. Figura adaptada de Rao [6].

Plote o gráfico do amortecimento em função da relação d/D.

# 5 PROGRAMAÇÃO

Um dos aspectos mais poderosos do MATLAB é a possibilidade de se criar programas em uma linguagem de programação interpretada[1] usando a mesma notação aceita na janela de comando.

Arquivos contendo código MATLAB são arquivos de texto com a extensão .m chamados de arquivos-M (M-files). Estes arquivos podem conter o código de *scripts* ou *funções*, cujas principais características estão relacionadas na Tabela 13.

| Arquivos de script | Arquivos de funções |
|---|---|
| Não aceitam argumentos nem retornam valores ao *workspace*. | Aceitam argumentos e retornam valores ao *workspace*. |
| Trabalham com as variáveis definidas no *workspace*. | Trabalham com variáveis definidas localmente ao arquivo. |
| Principal aplicação: automatização de comandos que precisam ser executados em uma certa sequência. | Principal aplicação: adaptação da linguagem MATLAB a qualquer situação de programação necessária. |

Tabela 13: Características das formas de programação MATLAB.

Sendo assim, uma vez escolhido o tipo de código, você deve abrir um arquivo *.m*. É possível usar qualquer editor de textos para sua criação, mas o uso do editor embutido no MATLAB é preferido por fornecer recursos úteis ao programador como auto-indentação, destaque de palavras reservadas, ferramentas de depuração, etc. Portanto, **siga o seguinte caminho**: File → New → Script ou File → New → Function. Neste arquivo deve ser escrito o programa. Uma vez escrita a rotina a ser executada, deve-se salvar o arquivo.

## 5.1 CRIANDO UM SCRIPT

Um *script* é meramente um conjunto de comandos MATLAB que são salvos em um arquivo. Eles são úteis para automatizar uma série de comandos que se pretende executar em mais de uma ocasião. **O script pode ser executado digitando o nome do arquivo na janela de comando ou chamando as opções de menu na janela de edição: *Debug* ou *Run*.**

Por exemplo, digite o código listado a seguir e salve-o com o nome *freefall.m*. Os comentários podem ser omitidos.

```
% Script 1 - script file to compute the
%velocity of the free-falling bungee jumper for
%the case where the initial velocity is zero.
clear all %limpa toda a memoria
clc %limpa a janela de comando
g = 9.81; m = 68.1; t = 12; cd = 0.25;
v = sqrt(g * m / cd) * tanh(sqrt(g * cd / m) * t)
```

1 Como a linguagem de programação do MATLAB é interpretada, todos os códigos precisam ser executados a partir do MATLAB. É possível criar executáveis independentes, assunto que não será discutido neste material.

Os arquivos do tipo *script* são úteis quando se deseja efetuar uma sequência de comandos com muita frequência. Como mostra o exemplo acima, os *scripts* se utilizam dos dados presentes na memória (*workspace*) para efetuar os comandos.

As primeiras linhas comentadas do arquivo *.m* são exibidas caso o usuário utilize o comando `help + nome do arquivo`. Ou seja:

```
» help freefall
Script 1 - script file to compute the
velocity of the free-falling bungee jumper for
the case where the initial velocity is zero.
```

A primeira linha de comentário, chamada de linha H1 é usada nas buscas por palavra-chave do comando `lookfor`. Assim, por exemplo:

```
» lookfor script
freefall   Script 1 - script file to compute the
```

Explicite a variável g,

```
» g
g =
     9.8100
```

## 5.2 CRIANDO UMA FUNÇÃO

Arquivos de função são arquivos-M que começam com a palavra `function`. Em contraste com arquivos de `script`, eles aceitam argumentos de entrada e saída. Portanto, eles são análogos às funções definidas pelo usuário em linguagens de programação como Fortran, Visual Basic ou C.

A sintaxe para o arquivo `function` pode ser representada generalizada como

```
function outvar = funcname(arglist)
% helpcomments
statements
outvar = value;
```

onde *outvar = nome da variável de saída*, *funcname = nome da função*, *arglist = lista de argumentos da função* (i.é, valores delimitados por vírgula que irão passar pela função), *helpcomments = texto usado para informar o usuário sobre a função* (esta parte pode ser chamada digitando `Help funcname` na janela de comandos), e *statements = funções do MATLAB para computar o valor de saída outvar*.

A primeira linha de *helpcomments* (H1) é usada nas buscas por palavra-chave do comando `lookfor`. Então, inclua palavras chave nessa linha. O arquivo deve ser salvo como *funcname.m*. Pode-se rodar a função digitando *funcname* na janela de comando. Importante ressaltar que, apesar do MATLAB diferenciar maiúsculas e minúsculas, o sistema operacional de seu computador pode não diferenciar. Enquanto o MATLAB trata funções como *freefall* e FreeFall como variáveis diferentes, para o sistema operacional elas podem ser a mesma coisa.

Cada função trabalha com variáveis locais, isoladas do espaço de memória do *workspace*. As funções podem ser executadas mais rapidamente que os scripts, pois quando um arquivo de função é chamado pela primeira vez, os comandos são compilados e colocados em memória, de modo que estão sempre prontos para executar. E permanecem assim enquanto durar a sessão MATLAB.

O programa script descrito anteriormente, por exemplo, se adequa ao formato `function` da seguinte forma:

```
function v = freefall(t, m, cd)
% freefall: bungee velocity with second-order drag
% v=freefall(t,m,cd) computes the free-fall velocity
% of an object with second-order drag
% input:
% t = time (s)
% m = mass (kg)
% cd = second-order drag coefficient (kg/m)
% output:
% v = downward velocity (m/s)
g = 9.81; % acceleration of gravity
v = sqrt(g * m / cd)*tanh(sqrt(g * cd / m) * t);
```

Exemplo de execução da função `freefall`,

```
» freefall(12,68.1,0.25)
ans =
        50.6175
```

Tente explicitar a variável g,

```
» g
Undefined function or variable 'g'.
```

## 5.3 CONTROLE DE FLUXO

Como a maioria das linguagens de programação, o MATLAB permite o uso de estruturas condicionais e repetitivas para controle de fluxo. Nesse trabalho discutiremos algumas formas de uso dos comandos `if`, `while`, `for`, `switch` [2].

### 5.3.1 *Estrutura condicional if*

O comando `if` avalia uma expressão e, dependendo de seu valor, executa um determinado conjunto de instruções. Em sua forma mais simples, usa-se:

```
if expressao
  <COMANDOS>
end
```

Se a expressão lógica for verdadeira todos os comandos até o finalizador `end` serão executados. Em caso contrário a execução do código continuará na instrução após o finalizador `end`. A forma completa da estrutura inclui a declaração `else` para a indicação de comandos a serem executados se a expressão for falsa:

```
if expressao
  <COMANDOS V>
else
  <COMANDOS F>
end
```

As expressões podem incluir operadores relacionais e lógicos, como mostrado na Tabela 9, definida no Capítulo 3. Exemplo:

```
if x == 0
   y = sin(3*t+a);
```

2 O MATLAB reconhece 6 estruturas de controle de fluxo: `if`, `switch`, `while`, `for`, `try-catch`, `return`.

```
  else
     y = cos(3*t-a);
5 end
```

### 5.3.2 *Estrutura repetitiva while*

O laço `while` executa um grupo de comandos enquanto uma expressão de controle `for` avaliada como verdadeira. A sintaxe para este tipo de laço é:

```
  while expressao
    <COMANDOS>
  end
```

Exemplo de um trecho de código que simula a operação da função interna `sum`:

```
  S = 0;
2 i = 1;
  while i < = length(V)
    S = S+V(i);
    i = i+1;
  end   % Neste ponto, S = sum(V)
```

### 5.3.3 *Estrutura repetitiva for*

O laço `for` executa repetidamente um conjunto de comandos por um número especificado de vezes. Sua forma geral é:

```
  for variavel de controle = valor inicial: incremento: valor final
    <COMANDOS>
  end
```

O incremento pode ser negativo ou omitido (caso em que será adotado um incremento unitário). Exemplo de um trecho de código que simula a operação da função interna `max`:

```
  M = V(1);
2 for i = 2:length(V)   % O incremento foi omitido!
    if V(i) > M
      M = V(i);
    end
  end   % Neste ponto, M = max(V)
```

### 5.3.4 *A estrutura switch*

A estrutura `switch` é uma alternativa à utilização de `if, elseif, else`. A sintaxe geral é,

```
  switch expressao_de_entrada    %escalar ou string
    case valor1
        grupo de sentencas 1
4   case valor2
        grupo de sentencas 2
  ...
    otherwise
        grupo de sentencas n
```

```
9 end
```

A estrutura `switch` é capaz de lidar com múltiplas condições em uma única sentença estrutura `case`. Veja o exemplo abaixo.

```
switch angulo
  case {0,360}
      disp('Norte')
  case {-180,180}
      disp('Sul')
  case {-270,90}
      disp('Leste')
  case {-90,270}
      disp('Oeste')
  otherwise
      disp('Direcao desconhecida')
end
```

## 5.4 VETORIZAÇÃO

O laço `for` é fácil de implementar e entender. No entanto, para MATLAB, não é, necessariamente, o meio mais eficiente para repetir ações um número específico de vezes. Devido à capacidade MATLAB para operar diretamente sobre matrizes, vetorização fornece uma opção muito mais eficiente. Por exemplo, o seguinte para a estrutura `loop`:

```
i = 0;
for t = 0:0.02:50
i = i + 1;
y(i) = cos(t);
end
```

pode ser representada na forma vetorizada como,

```
t = 0:0.02:50;
y = cos(t);
```

Deve-se notar que, para um código mais complexo, a vetorização pode não ser tão evidente. Dito isto, a vetorização é recomendada sempre que possível.

## 5.5 ENTRADA DE DADOS

A função `input` permite a entrada de dados ou expressões durante a execução de *script* ou *function*, exibindo (opcionalmente) um texto ao usuário. Exemplo:

```
N = input('Digite o tamanho do vetor: ');
```

Se o valor de entrada for uma expressão, seu valor será avaliado antes da atribuição à variável usada no comando. Se o valor de entrada for um texto ou caractere $'s'$ (*string*) deve ser incluído na lista de argumentos da função. Exemplo:

```
Titulo_Grafico = input('Titulo do grafico: ','s');
```

Outra forma disponível de interação via teclado é dada pela função `pause`. Quando usada sem argumentos, a instrução interrompe a execução de um script

ou função até que o usuário pressione alguma tecla[3]. A função `pause` é especialmente útil para permitir ao usuário a leitura de várias informações impressas em tela ou durante a fase de depuração do programa.

O comando `load` também pode ser utilizado para importar dados, salvos em formato ASCII ou de texto. No entanto, as variáveis devem estar no formato MATLAB,

```
load nome_arquivo
```

ou

```
nome_variavel= load nome_arquivo
```

## 5.6 COMANDO FPRINTF

O objetivo não é esgotar o assunto, e, portanto, o exemplo abaixo mostra como fazer uma função interativa, e organizar a saída de dados. Pesquise mais sobre a função, se precisar.

```
function freefalli
% freefalli: interactive bungee velocity
% freefalli interactive computation of the
% free-fall velocity of an object
% with second-order drag.
g = 9.81; % acceleration of gravity
m = input('Mass (kg): ');
cd = input('Drag coefficient (kg/m): ');
t = input('Time (s): ');
v = sqrt(g * m / cd)*tanh(sqrt(g * cd / m) * t);
disp(' ')
fprintf('The velocity is %8.4f m/s\n', v)
```

Rode a função e perceba que os dados serão inseridos dentro da função, e não serão armazenados na *workspace*,

```
» freefalli
Mass (kg): 68.1
Drag coefficient (kg/m): 0.25
Time (s): 12

The velocity is 50.62 m/s
```

## 5.7 EDIÇÃO DE FUNÇÕES EXISTENTES

O código da maioria das funções discutidas nesse texto pode ser visualizado ou editado, digitando-se:

```
» edit <nome da funcao>
```

Apesar de possível, não é recomendável alterar diretamente o código das funções internas do MATLAB, como `inv`, `max`, etc... Se desejar, crie uma nova versão com outro nome.

3 A função `pause` também pode ser usada com argumentos. Quando usada sob a forma `pause(N)`, a função interrompe a execução do código atual por N segundos.

## 5.8 SUBFUNÇÕES

Os arquivos-M podem conter mais de uma função. A primeira delas, cujo nome deve coincidir com o nome do arquivo, é a *função primária*, enquanto as demais são *subfunções*. As subfunções podem ser definidas em qualquer ordem após a função primária e suas variáveis sempre tem escopo local. Não é preciso usar qualquer indicação especial de fim de função porque a presença de um novo cabeçalho indica o fim da função (ou subfunção) anterior. Como exemplo, analise o código da função Baskara, listado a seguir.

```
% BASKARA.M - Exemplo de uso de uma subfuncao para
% calculo de raizes de equacao do 2o grau

% Funcao primaria: mesmo nome que o do arquivo .M
function x = Baskara(v)

a = v(1); b = v(2); c = v(3);   % Obtem coeficientes
D = Delta(a,b,c);               % Calcula "Δ"

% Calcula raizes reais, se existirem
if isreal(D)
   r1 = (-b+D)/(2*a);   % Calcula raiz
   r2 = (-b-D)/(2*a);   % Calcula raiz
   if r1 == r2
      x = r1;       % Retorna apenas uma raiz
   else
      x = [r1; r2]; % Retorna raizes distintas
   end
else
   disp('A equacao nao possui raizes reais');
   x = [];   % Retorno nulo
end

% Subfuncao para calculo de "Δ"
function d = Delta(a,b,c)
d = sqrt(b^2-4*a*c);
```

## 5.9 EXERCÍCIOS

1. Para os vetores: $x = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$ $y = \begin{bmatrix} 20.4 & 12.6 & 17.8 & 88.7 & 120.4 \end{bmatrix}$

   Crie uma função que organize os dados da seguinte maneira,

   | x | y |
   |---|---|
   | 1 | 20.40 |
   | 2 | 12.60 |
   | 3 | 17.80 |
   | 4 | 88.70 |
   | 5 | 120.40 |

   Dica: Crie uma variável $z = \begin{bmatrix} x; y \end{bmatrix}$ e use `fprintf`.

2. Escreva um programa utilizando a função `switch` para calcular o total de dias decorridos em um ano, dados: número do mês, dia e a indicação se o ano é bissexto ou não.

3. Escreva um programa *.m* para calcular a exponencial de uma matriz. O cálculo da exponencial de uma matriz é muito importante na área de sistemas dinâmicos. Uma aplicação de exponencial de matriz é a solução homogênea

(devido à condição inicial) de uma equação diferencial ordinária. Crie uma `function` que realiza o cálculo da exponencial de uma matriz partir da sua expansão em séries. A expansão em série de uma função exponencial é dada por:

$$e^A = I + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + + \frac{1}{4!}A^4 + ...$$

onde $A$ é uma matriz de dimensão $nxn$ e $I$ é a matriz identidade de dimensão $nxn$. Essa função deve receber uma matriz quadrada de dimensão qualquer, o número N de termos da série e retornar a exponencial $e^A$ calculada com N termos.

Para testar a sua função crie uma matriz com dimensão 4x4 e calcule a sua exponencial com diferentes números de termos na série (diferentes valores de N). Utilize, por exemplo, N variando de 3 a 10. Compare o resultado da sua função com a função `exp` do MATLAB.

A função `exp` do MATLAB calcula exponencial de matrizes. Note que a palavra `exp` é utilizada pelo MATLAB e, portanto, ela não pode ser utilizada como nome de uma função criada pelo usuário.

4. Crie um *script* para calcular a rigidez equivalente $k$ de um sistema de suspensão similar ao ilustrado na Figura 13. Inicialmente defina a equação de rigidez de mola helicoidal e calcule a rigidez $k$ de cada mola; calcule a rigidez equivalente para as 3 molas idênticas e paralelas da figura.

$$k = \frac{d^4 G}{8D^3 n}$$

onde $d$ é o diâmetro do arame que compõe a mola; $G$ é o módulo cisalhante do material; $D$ é o diâmetro médio da mola; $n$ é o número de espiras ativas. Para testar seu programa, considere que cada uma das três molas helicoidais é fabricada em aço, com $G = 80GPa$, 5 espiras efetivas, $D = 2000mm$ e $d = 200mm$. A rigidez equivalente desse conjunto de molas deve ser: $k_{eq} = 120N/mm$.

Figura 13: Figura extraída de `http://www.sctco.com/pdf/sect_1.pdf`.

5. Crie um `script` que plote a constante elástica torcional equivalente de um eixo de um propulsor a hélice, em função da espessura $50mm < t < 220mm$ das paredes (constante para as duas seções), conforme Figura 14.

6. Crie uma `function` que calcule a constante elástica equivalente de um tambor de içamento equipado com cabo de aço e montado conforme a Figura 15.

7. **DESAFIO:** O filme Contatos Imediatos do 3º grau (em inglês Close Encounters of the Third Kind, algumas vezes abreviado como CE3K ou simplesmente Close Encounters), de 1977, foi escrito e dirigido por Steven Spielberg. O título é tirado da *classificação de contatos imediatos com alienígenas* criada pelo ufologista J. Allen Hynek - veja Figura 16.

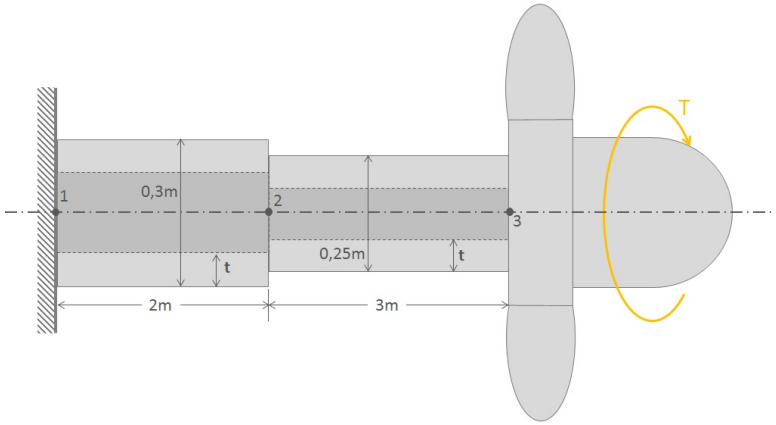



Figura 14: Figura adaptada de Rao [6].

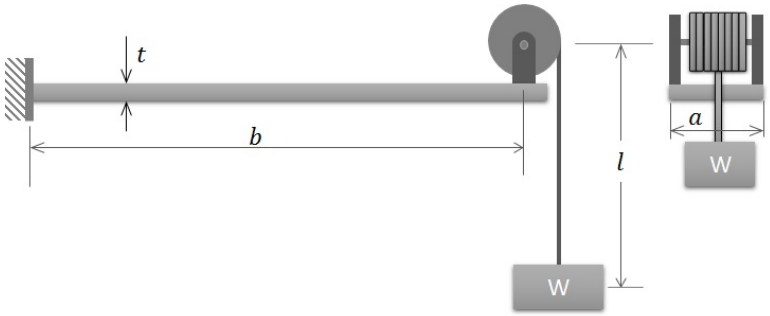



Figura 15: Tambor de içamento.

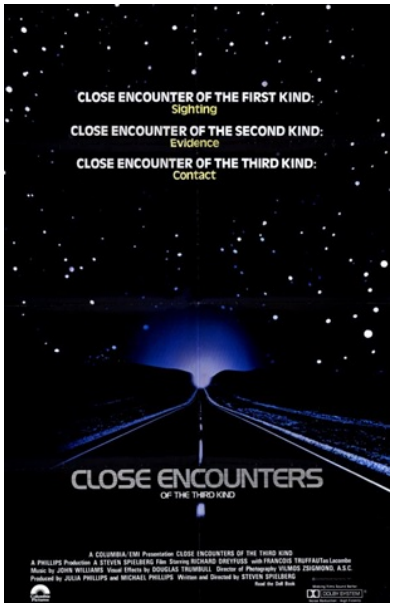



Figura 16: Close Encounters

A comunicação entre os humanos e a raça alienígena era feita através de uma sequência de tons que os cientistas acreditavam ser reconhecida pelos alienígenas. Esta sequência era composta por 5 tons nas frequências 493,9Hz, 554,4Hz, 440Hz, 220Hz e 329,6Hz. Sua tarefa consiste em criar um programa MATLAB que gere esta sequência de tons, considerando todos com a mesma duração. Mais precisamente você deve criar uma função `contatos(T)` em que `T` é a duração de cada tom da sequência. Por exemplo, ao digitar:

```
» contatos(5)
```

deverá ser gerada a sequência de tons nos alto-falantes do PC com duração total de 5s.

Parte III

# POLINÔMIOS E SISTEMAS DE EQUAÇÕES LINEARES

Soluções de polinômios e de equações lineares e são bastante simples no MATLAB. Um polinômio é representado por um vetor linha contendo seus coeficientes em ordem crescente, enquanto um sistema de equações lineares pode ser escrito sob a forma matricial.

# 6 POLINÔMIOS

No MATLAB um polinômio é representado por um vetor linha contendo seus coeficientes em ordem crescente. Isto é, uma função polinomial da forma:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$$

pode ser representada no MATLAB por um vetor de coeficientes, em ordem decrescente de potência:

$$p = [a_n \quad a_{n-1} \quad ... \quad a_2 \quad a_1 \quad a_0]$$

Por exemplo, o polinômio $g(x) = x^3 - 2x - 5$ é representado pelo seguinte vetor:

```
» g = [1 0 -2 -5];
```

O MATLAB possui funções específicas para operações com polinômios, como a determinação de raízes, avaliação, diferenciação, etc.

As raízes (reais ou complexas) de um polinômio são calculadas pela função `roots`. Por exemplo,

```
» r = roots(g)
r =
     2.0946 + 0.0000i
    -1.0473 + 1.1359i
    -1.0473 - 1.1359i
```

Note a forma de representação de números complexos no MATLAB (parte real + parte imaginária * i), já estudada anteriormente.

De forma inversa, se forem conhecidas as raízes de um polinômio, a função `poly` reconstrói o polinômio original. Por exemplo, os coeficientes do vetor g do exemplo anterior, podem ser recuperados pela instrução:

```
» p1 = poly(r)
p1 =
      1.0000   -0.0000   -2.0000   -5.0000
```

»

Avaliar um polinômio significa determinar o valor de $p(x)$ para um dado valor de x. Para calcular, por exemplo, $g(2.4)$ usa-se a função `polyval`,

```
» y = polyval(g,2.4)
y =
     4.0240
```

As operações de multiplicação e divisão entre polinômios correspondem, respectivamente, às operações de convolução e deconvolução, implementadas pelas funções `conv` e `deconv`. Por exemplo, considere os seguintes polinômios:

$$f(x) = 9x^3 - 5x^2 + 3x + 7 \qquad g(x) = 6x^2 - x + 2$$

O produto $f(x)$ e $g(x)$ pode ser calculado com a seguinte sequência de comandos:

```
» f=[9,-5,3,7];
» g=[6,-1,2];
» produto=conv(f,g)
» [quociente,resto]=deconv(f,g)
produto =
             54      -39       41       29       -1   14
quociente =
             1.5000  -0.5833
resto =
             0       0        -0.5833  8.1667
```

Note que o grau do polinômio resultante é dado pela: soma dos graus dos polinômios envolvidos na multiplicação; e a subtração dos graus dos polinômios envolvidos na divisão.

A derivada de uma função polinomial pode ser obtida diretamente a partir do vetor que representa a função com o uso da função `polyder`. Por exemplo, a derivada de $f(x) = 2x^3 + x^2 - 3x$ pode ser calculada com:

```
» f = [2 1 -3 0];
» f1 = polyder(f)
f1 =
      6  2  -3
```

## 6.1 EXERCÍCIOS

Os exercícios sobre Polinômios foram extraídos de [4].

1. Obtenha a raíz do polinômio,

$$x^3 + 13x^2 + 52x + 6 = 0$$

2. Utilize o MATLAB para confirmar que,

$$\left(20x^3 - 7x^2 + 5x + 10\right)\left(4x^2 + 12x - 3\right) = 80x^5 + 212x^4 - 124x^3 + 121x^2 + 105x - 30$$

3. Utilize o MATLAB para confirmar que,

$$\frac{12x^3 + 5x^2 - 2x + 3}{3x^2 - 7x + 4} = 4x + 11, \qquad \text{resto: } 59x - 41$$

4. Utilize o MATLAB para confirmar que,

$$\frac{6x^3 + 4x^2 - 5}{12x^3 - 7x^2 + 3x + 9} = 0,7108, \qquad \text{quando } x = 2.$$

# 7 SISTEMAS DE EQUAÇÕES LINEARES

Todo sistema de equações lineares pode ser escrito sob a forma matricial $Ax = b$. Por exemplo, para o sistema,

$$\begin{aligned} 3x_1 - 2x_2 + x_3 &= -4 \\ + 2x_2 - x_3 &= 7 \\ 4x_1 + x_2 + 2x_3 &= 0 \end{aligned}$$

tem-se

$$A = \begin{pmatrix} 3 & -2 & 1 \\ 0 & 2 & -1 \\ 4 & 1 & 2 \end{pmatrix}; x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}; b = \begin{pmatrix} -4 \\ 7 \\ 0 \end{pmatrix}$$

Se a matriz dos coeficientes $(A)$ for quadrada e não-singular, ou seja, sem linhas ou colunas linearmente dependentes, a solução (única) do sistema é dada por:

$$x = A^{-1}b$$

Essa solução pode ser calculada de forma direta pelo MATLAB por qualquer uma das instruções:

```
» x = inv(A)*b ;
» x = A\b
```

As duas formas fornecem as mesmas respostas, mas os cálculos envolvidos no uso do operador $<$ exigem menos memória e são mais rápidos do que os envolvidos no cálculo de uma matriz inversa. O MATLAB também resolve sistemas sob a forma $xA = b$ ou sistemas com mais de uma solução, mas essas soluções não serão discutidas neste material.

## 7.1 EXERCÍCIOS

1. Resolva, se possível, os seguintes sistemas lineares.

$$\begin{aligned} 3x_1 - 2x_2 + x_3 &= -4 \\ + 2x_2 - x_3 &= 7 \\ 4x_1 + x_2 + 2x_3 &= 0 \end{aligned}$$

$$\begin{aligned} x_1 + 4x_2 + 7x_3 &= 5 \\ -3x_1 + 0x_2 - 9x_3 &= 1 \\ 2x_1 + 5x_2 + 11x_3 &= -2 \end{aligned}$$

$$\begin{aligned} x_1 + 2x_2 &= -4 \\ 3x_1 + 6x_2 &= 5 \end{aligned}$$

$$\begin{aligned} x_1 + 2x_2 &= 4 \\ 3x_1 + 4x_2 &= 5 \end{aligned}$$

2. Considere a Figura 17 representando um sistema de 4 molas ligadas em série sujeito a uma força F de 2700N. Determine as relações de equilíbrio, e os deslocamentos $x_i$ no MATLAB, dadas as constantes das molas (em MPa): $k_1 = 150$, $k_2 = 50$, $k_3 = 75$ e $k_4 = 225$.

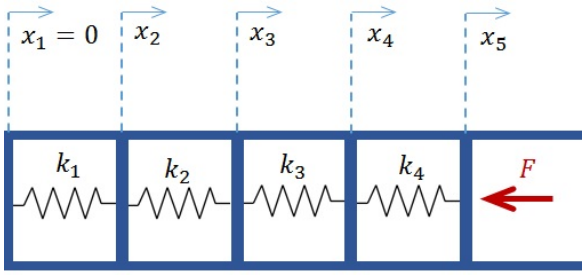



Figura 17: Molas em série.

3. Uma transportadora tem três tipos de caminhões, $C_1$, $C_2$ e $C_3$, que estão equipados para levar três tipos diferentes de máquinas, de acordo com a seguinte tabela:

| Caminhão | máquina A | máquina B | máquina C |
|---|---|---|---|
| 1 | 1 | 0 | 2 |
| 2 | 1 | 1 | 1 |
| 3 | 1 | 2 | 1 |

Por exemplo, o caminhão 1 pode levar uma máquina A, nenhuma máquina B e duas máquinas C.

Supondo que cada caminhão vai com sua carga máxima, quantos caminhões de cada tipo devemos enviar para transportar exatamente 12 máquinas A, 10 máquinas B e 16 máquinas C?

# Parte IV

# SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS

A simulação de um sistema dinâmico consiste na solução de equações diferenciais para considções iniciais de contorno. Sistemas de equações diferenciais lineares e não lineares podem ser escritos na forma matricial e resolvidos com ajuda do MATLAB.

# 8

# SISTEMAS DINÂMICOS LINEARES INVARIANTES NO TEMPO

## 8.1 REPRESENTAÇÃO DE EQUAÇÕES DIFERENCIAIS NO ESPAÇO DE ESTADOS

Vamos, inicialmente, adicionar algumas definições importantes na análise de um sistema dinâmico. Define-se como *estado* o menor conjunto de variáveis que determinam completamente o comportamento do sistema para qualquer instante $t$. Para tal, é necessário o conhecimento dessas variáveis no instante $t = t_0$ e das *variáveis de entrada* no instante $t \geqslant t_0$.

Qualquer sistema dinâmico linear de $m$ entradas: $u_1(t), u_2(t), ..., u_m(t)$; $p$ saídas: $y_1(t), y_2(t), ..., y_p(t)$ e $n$ variáveis de estado: $x_1(t), x_2(t), ..., x_n(t)$, pode ser escrito na seguinte forma:

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \qquad \text{equação dos estados} \\ y(t) &= C(t)x(t) + D(t)u(t) \qquad \text{equação de saída} \end{aligned} \tag{1}$$

onde

- $x(t)$- vetor de variáveis de estados (dimensão $nx1$);
- $u(t)$ - vetor de variáveis de entrada (dimensão $mx1$);
- $y(t)$ - vetor de variáveis de saída (dimensão $px1$);
- $A(t)$- matriz de transmissão dos estados ($nxn$);
- $B(t)$ - matriz de coeficientes de entrada ($nxm$);
- $C(t)$ - matriz de coeficientes de saída ou matriz dos sensores ($pxn$);
- $D(t)$ - matriz de coeficientes de alimentação direta ($pxm$).

As variáveis de estado $x(t)$ representam a condição instantânea do sistema. Importante ressaltar que **a primeira derivada das variáveis de estado sempre estão presentes nas equações dinâmicas**. Quando o modelo matemático é obtido usando as leis da fícisa, então as variáveis de estado são aquelas associadas às diversas formas de energia armazenadas no sistema. Por exemplo, em um sistema mecânico, geralmente posição e velocidade, associadas à energia potencial e cinética, respectivamente, são as variáveis de estado.

As variáveis de entrada $u(t)$ aqui consideradas são geradas por agentes externos (fontes) que alteram as condições de energia do sistema. Existe diferença entre variáveis de entrada e de perturbação. As variáveis de entrada são utilizadas para controlar o sistema, enquanto que as variáveis de perturbação são desconhecidas e, geralmente, *dificultam* o controle.

As variáveis de saída $y(t)$ são medidas por sensores instalados no sistema, são as variáveis controladas.

A forma da Equação 1 de representar o modelo matemático de um sistema dinâmico é conhecida como *forma do espaço dos estados*. Nessa forma, um sistema dinâmico de ordem $n$ é representado por um conjunto de $n$ equações diferenciais de primeira ordem. A forma do espaço de estados é usada em Controle Moderno e será utilizada aqui para resolver sistemas lineares e não lineares. Para maiores

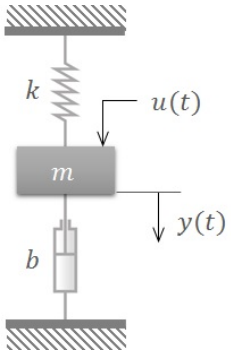



Figura 18: Sistema mecânico massa-mola-amortecedor. Figura adaptada de Ogata [3].

detalhes, estude o capítulo 2 do livro texto da disciplina [3]. Existe ainda a representação pela função de transferência, usada em Controle Clássico, assunto a ser tratado futuramente.

Como exemplo considere o sistema massa-mola-amortecedor da Figura 18, cujo modelo é representado pela seguinte equação diferencial de $2^a$ ordem:

$$m\ddot{y}(t) + b\dot{y}(t) + ky(t) = u(t) \tag{2}$$

Como o sistema é de segunda ordem, contém duas variáveis de estado. Define-se os *estados do sistema* como sendo a posição e a velocidade da massa, respectivamente:

$$x_1(t) = y(t)$$
$$x_2(t) = \dot{y}(t)$$

e a entrada,

$$u_1(t) = u(t)$$

No caso, a entrada é um escalar e não um vetor, ($m = 1$). Colocando na forma de espaço dos estados (Equação 1), por substituição na Equação 2, tem-se

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{1}{m}\left(-kx_1 - bx_2\right) + \frac{1}{m}u_1(t)$$

Define-se o vetor de estados, de dimensão 2x1 (i.é, $n = 2$), como

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

Portanto, define-se a equação de estado sob a forma matricial como,

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-k}{m} & \frac{-b}{m} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) \tag{3}$$

A equação de saída é,

$$y(t) = x_1(t) \tag{4}$$

ou, na forma matricial,

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

Comparando-se Equação 3 e Equação 4 com Equação 1, tem-se,

$$A = \begin{bmatrix} 0 & 1 \\ \frac{-k}{m} & \frac{-b}{m} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = 0$$

Como as matrizes não envolvem a função tempo $t$ explicitamente, tem-se um *sistema invariante no tempo*.

Existe uma classe própria no MATLAB para sistema lineares descritos na forma do espaço dos estados criada pela função `ss` (que representa, em inglês, *state space*) e definida pelas matrizes $A, B, C$ e $D$.

Por exemplo, o sistema:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

pode ser armazenado em uma variável tipo `sys` no MATLAB pela seguinte seqüência de comandos:

Essa solução pode ser calculada de forma direta pelo MATLAB por qualquer uma das instruções:

```
» A = [0 1; -3 -2]
» B = [0; 3]
» C = [1 0]
» D = [0]
» sys = ss(A,B,C,D)
A=
       0   1
       -3  -2

B =
       0
       3

C =
       0   1

D =
       0

sys =
a =
       x1  x2
x1     0   1
x2     -3  -2

b =
       u1
x1     0
x2     3

c =
       x1  x2
y1     1   0

d =
       u1
y1     0

Continuous-time state-space model.
```

## 8.2 SIMULAÇÃO DE SISTEMAS DINÂMICOS LINEARES

Existem funções específicas para simular o comportamento de sistemas lineares e invariantes no tempo (LIT) a entradas do tipo impulso, degrau ou de formas genéricas.

### 8.2.1 *Função impulso*

Para simular a resposta a um impulso unitário (em $t = 0s$) de um sistema utiliza-se a função `impulse`, fornecendo as matrizes representativas do sistema pela variável tipo `sys`. Considerando o sistema `sys` do exemplo anterior, a resposta ao impulso é obtida com o comando:

```
» impulse(sys);
```

O resultado da simulação é apresentado em uma janela gráfica, como mostra a Figura 19. Opcionalmente, pode-se fornecer um valor em segundos para o tempo final de simulação. Por exemplo, para simulação de 10s deve-se digitar o comando (Figura 20),

```
» impulse(sys,10);
```

É possível, ainda, armazenar os vetores do tempo de simulação (criado automaticamente pelo MATLAB) e da resposta do sistema, sem desenhar o gráfico correspondente. Por exemplo, o comando,

```
» [y t] = impulse(sys,10);
```

retorna vetores de tempo e saída.

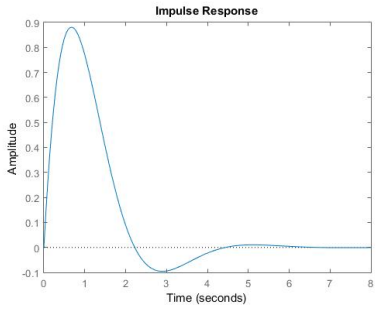



Figura 19: Função impulso.

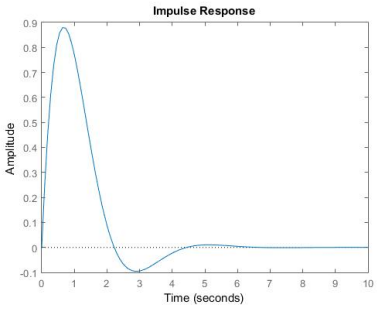



Figura 20: Função impulso 10s.

### 8.2.2 *Função degrau unitário*

A simulação da resposta a uma entrada em degrau unitário é feita pela função step, como em:

```
» step(sys);
```

O resultado desta simulação está representado na Figura 21.

Não é possível alterar a amplitude do degrau usado na simulação. Porém, como se trata da simulação de um sistema linear, a saída para uma entrada em degrau de amplitude $A$ pode ser calculada como $y_A(t) = Ay(t)$.

É possível controlar o tempo de simulação e armazenar os vetores de resposta (saída e tempo). No exemplo,

```
» [y t] = step(sys,10);
```

são armazenadas a saída $y$ e tempo $t$ para o intervalo de $0 - 10s$.

As funções `impulse` e `step` permitem que o usuário forneça um vetor de tempo a ser usado na simulação. Exemplo:

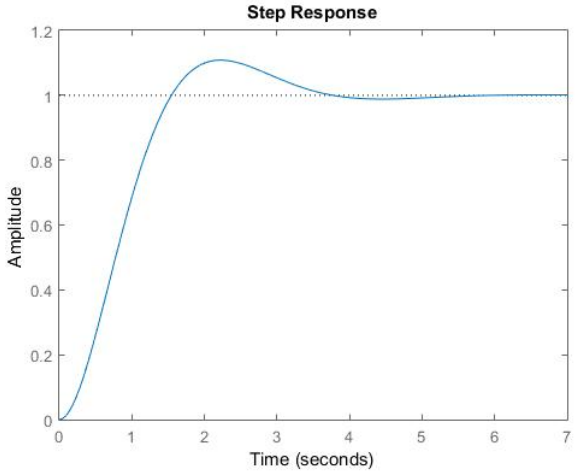



Figura 21: Função step do MATLAB

```
t = 0:0.01:15;
step(sys,t);
```

### 8.2.3 *Entrada genérica*

Para simular a resposta de um sistema linear a uma entrada genérica é preciso usar a função `lsim`, fornecendo a especificação do sistema e os vetores de entrada e de tempo de simulação. Para o sistema `sys` definido anteriormente,

```
t = 0:0.1:10; % Vetor de tempo de simulacao
u = zeros(length(t),1); % Vetor de entrada, com mesma dimensao de t
u(21:30) = 0.5; % Atribuicao de valores nao nulos
lsim(sys,u,t); % Simulacao
```

O resultado da simulação é apresentado em uma janela gráfica, como mostra a Figura 22.

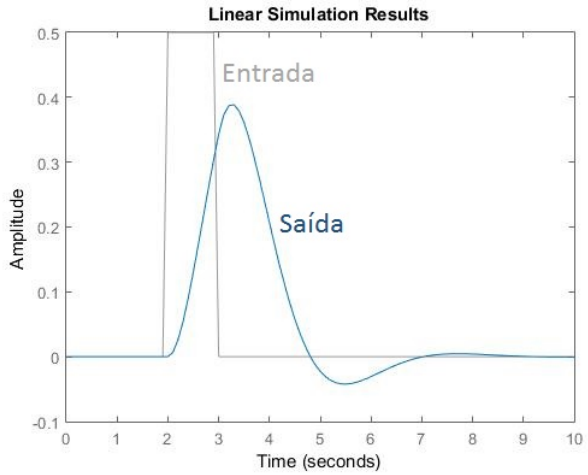



Figura 22: Resposta a um sinal genérico.

O comando `lsim` também pode ser utilizado na seguinte forma,

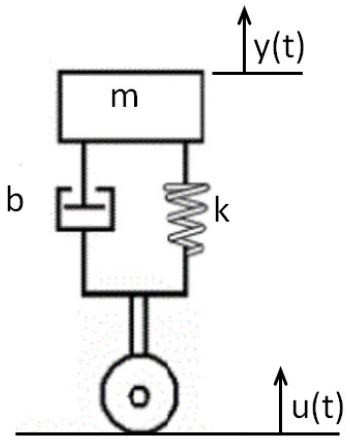



Figura 23: Sistema massa-mola-amortecedor representando 1/4 de veículo.

» `lsim(b,a,u,t);`
quando a equação diferencial tem a forma geral,

$$\frac{d^n}{dt^n}y(t)+a_{n-1}\frac{d^{n-1}}{dt^{n-1}}y(t)+...+a_1\frac{d}{dt}y(t)+a_0y(t)=b_m\frac{d^m}{dt^m}u(t)+...+b_1\frac{d}{dt}u(t)+b_0u(t) \tag{5}$$

onde

$b$ = $[b_m, b_{m-1}, b_{m-2}, ..., b_1, b_0]$ é o vetor de coeficientes especificados no lado direito da Equação 5, que é a equação de interesse;

$a$ = $[1, a_{n-1}, a_{n-2}, ..., a_1, a_0]$ é o vetor de coeficientes do lado esquerdo da Equação 5;

$u$ = é o vetor de instantes conhecidos do sinal $u(t)$ especificados na Equação 5;

$t$ = vetor da mesma dimensão de $u$, o k-ésimo elemento $t(k)$ de $t$ é o tempo, em segundos, no qual ocorre a entrada $u(k)$;

$y$ = vetor da mesma dimensão de $u$ e $t$ que representa instantes do sinal $y(t)$ que satisfazem a Equação 5.

Por exemplo, pode-se comprovar que a relação entre a altura $u(t)$ de uma via e a altura $y(t)$ do carro contendo um sistema de absorção de energia massa-mola entre as rodas e o chassis é (Figura 23),

$$\ddot{y}(t)+\frac{b}{m}\dot{y}(t)+\frac{k}{m}y(t)=g+\frac{b}{m}\dot{u}(t)+\frac{k}{m}u(t) \tag{6}$$

Veja que, nesse caso, $u(t)$ é um deslocamento (perturbação) e não uma força!

Por fim, pode-se converter a equação no formato Equação 6 em equações no espaço de estados, conforme Equação 1 através do comando `tf2ss`, do inglês *transfer function to state-space*:
« `[A,B,C,D]=tf2ss(b,a)`

## 8.3 EXERCÍCIOS

1. Dada a equação diferencial abaixo que representa a dinâmica de um sistema:

   $$2\ddot{y}(t)+\dot{y}(t)+3y(t)=5u(t)$$

   Pede-se:

- Dado que a saída do sistema é a variável $y(t)$, coloque o sistema na forma do espaço dos estados e defina as matrizes $A, B, C$ e $D$ do sistema.
- Defina o sistema no MATLAB usando uma variável tipo `sys`.
- Simule a resposta do sistema para uma força externa de entrada $u(t)$ na forma de degrau unitário no intervalo de tempo de 0 a 30 segundos. Apresente os gráficos da entrada degrau e da variável $y(t)$.
- Defina uma força externa de entrada $u(t)$ senoidal com frequência de $2rad/s$ no intervalo de tempo 0 a 10 segundos. Note que a função senoidal é dada por $\sin(2t)$. Use um vetor de tempo com incremento de 0.01 segundos.
- Simule a resposta do sistema para a entrada senoidal definida no item anterior. Apresente os gráficos da entrada degrau e da variável $y(t)$.

2. Suponha que um automóvel em movimento passe por diferentes obstáculos (elevações) na pista. Analise a resposta do $1/4$ de modelo de veículo aos dois obstáculos mostrados na Figura 24. Dados: $m = 100Kg$, $b = 500Ns/m$ e $k = 200N/m$. Duplique e quadruplique o amortecimento, e compare as respostas.

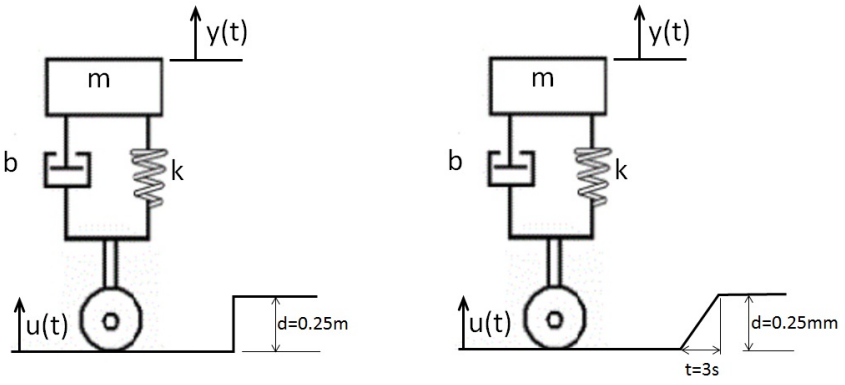



Figura 24: Dois modelos de obstáculo.

# 9 SISTEMAS DINÂMICOS NÃO LINEARES

A simulação de um sistema consiste na solução de suas equações diferenciais para condições iniciais e condições de contorno diferentes de zero. Condições de contorno são as entradas do sistema. Neste Capítulo será visto como se pode utilizar o MATLAB para resolver equações diferenciais não lineares. No MATLAB, há diversas funções, chamadas *solucionadores*, do inglês *solvers*, que utilizam o método Runge-Kutta em passo variável para resolver equações diferenciais numericamente. Os dois solucionadores mais utilizados são a função `ode45` e a função `ode15s`. A função básica, e que deve ser sempre testada primeiro, é a `ode45`, que utiliza combinação dos métodos de Runge-Kutta de quarta e quinta ordem. Se a solução da equação com esse solucionador apresentar problema de convergência ou erro, então utlize a ou, a função `ode15s`.

A sintaxe para para equações diferenciais de primeira ou segunda ordem é basicamente a mesma. No entanto, os arquivos .m são bastante diferentes.

## 9.1 EDO DE PRIMEIRA ORDEM

Para equações diferenciais de primeira ordem, do tipo,

$$\dot{y} = f(t, y) \qquad y(t_0) = y_0 \tag{7}$$

a sintaxe básica para `ode45`,

```
» [tout,yout]=ode45(@ydot,tspan,y0,options);
```

onde `@xdot` é uma `function` cujas entradas são $t, y$ e a saída é um vetor coluna (número de linhas igual à ordem da equação) que representa $dy/dt$, isto é, $f(t, y)$. O vetor `tspan=[t0, tf]` define o intervalo de tempo da simulação[1]; e `y0` é a condição inicial. O argumento `options` refere-se aos recursos avançados dos solucionadores, e não serão tratados aqui. Procure na Internet informações sobre o argumento, que é criado com a função `odeset`.

Enfim, essa função integra o sistema de equações diferenciais definido por $\dot{y} = f(t, y)$ do tempo inicial `t0` ao tempo final `tf` com condições iniciais `y0`. Melhor maneira de entender essa confusão toda é com um exemplo...

Dado o sistema descrito pela Equação 7:

$$\begin{cases} t^2\dot{y} & = y + 3t \\ y(1) & = -2 \end{cases} \qquad 1 \leqslant t \leqslant 4;$$

Inicialmente, cria-se a função `ydot`,

```
1 function  dydt= ydot(y,t);
   dydt=(y+3*t)/t^2;
  end
```

A condição inicial dada é que $y = -2$ para $t = 1$ e queremos integrar $1 \leqslant t \leqslant 4$. O seguinte conjunto de comandos mostra explicitamente a solução,

```
   tspan = [1 4];                    %vetor intervalo de integracao
2  y0 = -2;                          %condicao inicial
```

1 Quaisquer valores intermediários específicos entre `t0` e `tf` em que se deseja saber a solução podem ser adicionados em `tspan`, utilizando `tspan = [t0, t1, t2, ..., tf]`

```
[tout,yout] = ode45(@ydot,tspan,y0); %resolve o problema
plot(yout,tout)
```

## 9.2 EDO DE SEGUNDA ORDEM

Para resolver uma equação de segunda ordem (ou superior) com os solucionadores de EDO do MATLAB você deve, inicialmente, escrever as equações na forma de variáveis de estado. Considere o exemplo,

$$5\ddot{y} + 7\dot{y} + 4y = u(t) \qquad 0 \leqslant t \leqslant 6$$

para $u(t) = \sin(t)$ e condições iniciais $y(0) = 0$ e $\dot{y}(0) = 9$.

Define-se duas novas variáveis, $x_1$ e $x_2$ de modo que,

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \ddot{y} = \frac{1}{5}u(t) - \frac{4}{5}x_1 - \frac{7}{5}x_2$$

Próximo passo é criar uma função que calcule os valores de $\dot{x}_1$ e $\dot{x}_2$ e armazene-os em um vetor coluna,

```
1  function xdot=estado_1(t,x)
   xdot=[x(2); (1/5)*(sin(t)-4*x(1)-7*x(2))];
```

E, para utilizar a função `ode45`,

» `[t,x]=ode45(@estado_1,[0,6],[3,9]);`

Para plotar as duas funções $x_1$ e $x_2$ versus t, utilize a função `plot(t,x)`; para plotar apenas $y = x_1$ digite `plot(t,x(:,1))`.

## 9.3 MODELO DE UM PÊNDULO NÃO LINEAR

Esta seção é um resumo do exemplo apresentado em Palm III [4], capítulo 9, páginas 389-392. O exemplo refere-se a um pêndulo de massa $m$ concentrada na extremidade de uma haste de massa desprezível, mostrado na Figura 25. A equação de movimento do sistema é,

$$\ddot{\theta} + \frac{g}{L}\sin\theta = 0$$

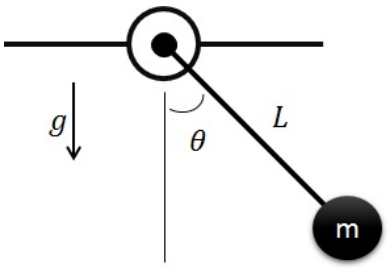



Figura 25: Pêndulo.

Suponha $L = 1m$ e $g = 9,81m/s^2$. Utiza-se o MATLAB para resolver a equação para $\theta(t)$ em dois casos: $\theta(0) = 0,5rad$ e $\theta(0) = 0,8\pi rad$, sempre com $\dot{\theta}(0) = 0$.

### 9.3.1 *Linearização do problema*

Para ângulos pequenos, $\sin\theta \approx \theta$, tornando a equação linear,

$$\ddot{\theta} + \frac{g}{L}\theta = 0$$

cuja solução é trivial,

$$\theta(t) = \theta(0)\cos\sqrt{\frac{g}{L}}t$$

para $\dot{\theta}(0) = 0$. Assim, a amplitude de oscilação é $\theta(0)$ e o período é $T = 2\pi\sqrt{L/g} = 2,006s$

### 9.3.2 *Equações de estado*

Sejam $x_1 = \theta$ e $x_2 = \dot{\theta}$,

$$\dot{x}_1 = \dot{\theta} = x_2$$
$$\dot{x}_2 = \ddot{\theta} = -\frac{g}{L}\sin x_1$$

Dessa forma, soluciona-se os dois casos propostos gerando a `function`,

```
function xdot=pendulum(t,x)
 g=9.81;  L=1;
 xdot=[x(2); -(g/L)*sin(x(1))];
end
```

e os comandos (cuidado com o comando `gtext`, aprenda a usá-lo),

```
[ta,xa]=ode45(@pendulum,[0,5],[0.5,0]);
[tb,xb]=ode45(@pendulum,[0,5],[0.8*pi,0]);
plot(ta,xa(:,1),tb,xb(:,1));
xlabel('Tempo [s]');
ylabel('Angulo [rad]');
gtext('Caso i'),gtext('Caso 2');
```

A solução está ilustrada na Figura 26.

## 9.4 SISTEMA DE VÁRIAS EQUAÇÕES NÃO LINEARES ACOPLADAS

Para o seguinte sistema acoplado de equações diferenciais,

$$\begin{aligned}\ddot{y} &= \dot{x} + \dot{y} + \cos y \\ \ddot{x} &= \dot{y}^2 + \tan y\end{aligned} \tag{8}$$

A solução, via MATLAB, é a seguinte (Figura 27),

```
couplode = @(t,y) [y(2); y(4)^2 + tan(y(3)); y(4); cos(y(3)) + y(2) ...
    + y(4)];
[t,y] = ode45(couplode, [0 0.4999*pi], [0;0;0;0]);
figure(1)
plot(t, y)
grid
str = {'$$ \dot{y} $$', '$$ y $$', '$$ \dot{x} $$', '$$ x $$'};
legend(str, 'Interpreter','latex', 'Location','NW')
```

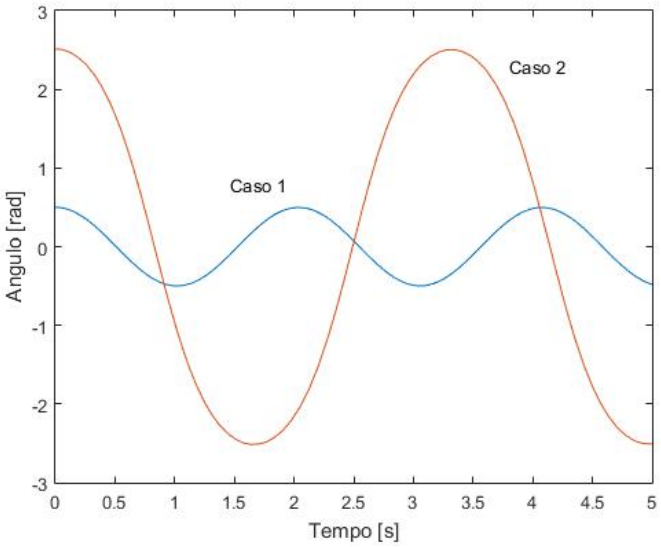



Figura 26: Solução do pêndulo para duas condições iniciais.

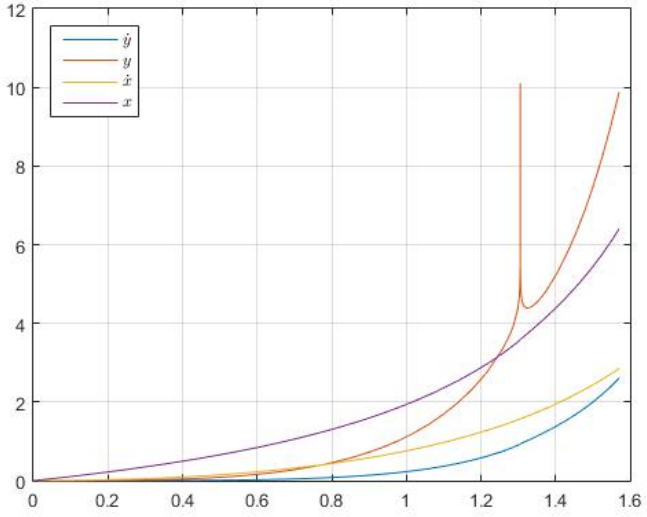



Figura 27: SOlução do sistema acoplado de equações diferenciais dado pela Equação 8.

### 9.4.1 *Exemplo: Sistema de transmissão*

O sistema de transmissão do *Porshe Panamera S E-Hybrid 2014* é mostrado na Figura 28. Destaca-se o caminho da unidade de propulsão, com os motores elétrico e à combustão até as rodas traseiras. O binário gerado pela combustão e forças de eletro-magnéticas da unidade de propulsão é aplicado ao volante de inércia do motor de combustão e ao rotor do motor elétrico. Ambas as partes têm inércia significativa e constituem a *primeira massa* do sistema MMAM (Massa-Mola-Amortecedor-Mola). O eixo de transmissão que liga a unidade de propulsão com o diferencial tem rigidez limitada e atua como uma *mola*. A elasticidade do eixo resulta no *amortecimento*. Finalmente, a inércia do diferencial e as rodas traseiras podem ser considerados como a *segunda massa do sistema*.

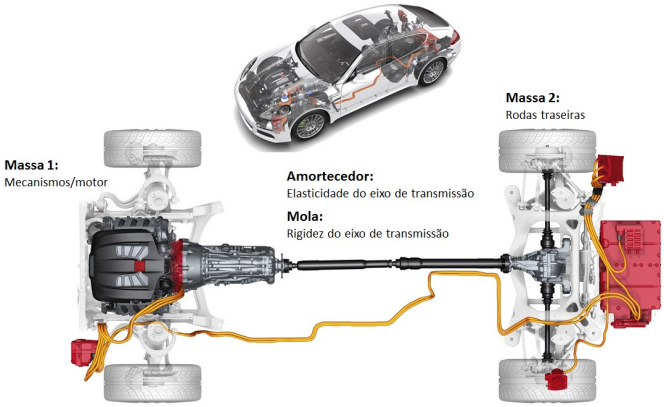



Figura 28: Sistema de transmissão do Porshe Panerama, modelo 2014.

O modelo dinâmico do sistema está ilustrado na Figura 29 e pode ser definido através das seguintes equações diferenciais,

$$\begin{aligned}
\dot{\theta}_1 &= \omega_1 \\
\dot{\theta}_2 &= \omega_2 \\
\dot{\omega}_1 &= \frac{1}{J_1}\left[k\left(\theta_2 - \theta 1\right) + d\left(\omega_2 - \omega_1\right) + \tau_m\right] \\
\dot{\omega}_2 &= \frac{1}{J_2}\left[k\left(\theta_1 - \theta 2\right) + d\omega_1 - \left(d + b\right)\omega_2\right]
\end{aligned} \tag{9}$$

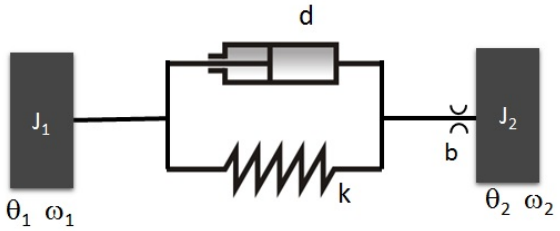



Figura 29: Esquematização do sistema MMAM.

Estude o modelo dinâmico para as variáveis definidas na Tabela 14.

| Nome | Descrição | Valor |
|---|---|---|
| $J_1$ | Momento de inércia da primeira massa | $3.75\ 10^{-6} kgm^2$ |
| $J_2$ | Momento de inércia da segunda massa | $3.75\ 10^{-6} kgm^2$ |
| $k$ | Rigidez torsional do eixo | $0.2656 Nm/rad$ |
| $d$ | Amortecimento torcional do eixo | $3.215\ 10^{-5} Nms/rad$ |
| $\tau_m$ | Torque do motor | $10\ 10^{-2} Nm$ |
| $b$ | Atrito viscoso | $1\ 10^{-5} Nms/rad$ |

Tabela 14: Parâmetros do sistema MMAM.

## 9.5 EXERCÍCIOS

1. Calcule e plote a seguinte equação linear (defina os limites),

$$10\frac{dy}{dt} + y = 20 + 7\sin 2t \qquad y(0) = 15$$

2. A equação,

$$10\dot{T} + T = T_b$$

descreve a temperatura $T(t)$ de um determinado objeto imerso em um banho líquido de temperatura constante $T_b$.

Suponha que a temperatura inicial do objeto é $T(0) = 70^oF$ e a temperatura do banho é $T_b = 170^oF$. Plote a temperatura $T(t)$ como função do tempo e defina:

- *Valor em estado estacionário:* limite da resposta quando $t \to \infty$;
- *Tempo de assentamento:* tempo para que a resposta alcance e se mantenha dentro de uma determinada faixa percentual (normalmente 2%) em torno do valor em estado estacionário;
- *Tempo de subida:* tempo necessário para que a resposta vá de 10 a 90% do valor em estado estacionário;
- *Resposta de pico:* o maior valor da resposta;
- *Tempo de pico:* o instante em que a resposta de pico ocorre.

3. Extraído de Palm III [4]. O modelo do circuito RC mostrado na Figura 30 pode ser encontrado a partir da lei das tensões de Kirchhoff e da conservação da carga:

$$RC\dot{y} + y = v(t)$$

Suponha que o valor de RC seja 0,1s. Utilize um método numérico para encontrar a resposta para o caso em que a tensão externa aplicada $v(t) = 0$ é zero e que a tensão inicial do capacitor seja $y(0) = 2V$. Compare os resultados com a solução analítica, que é

$$y(t) = 2e^{-10t}$$

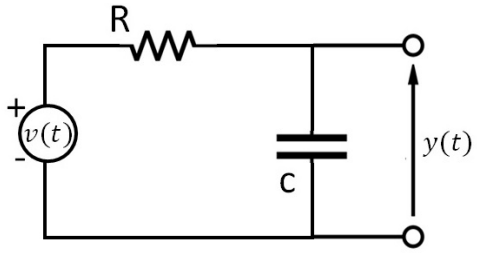



Figura 30: Circuito RC.

4. A equação de movimento para um pêndulo cuja base se move horizontalmente com uma aceleração $a(t)$ é,

$$L\ddot{\theta} + g\sin\theta = a(t)\cos\theta$$

Suponha $g = 9,81 m/s^2$, $L = 1m$ e $\dot{\theta}(0) = 0$. Plote $\theta(t)$ para $0 \leqslant t \leqslant 10s$ para os seguintes casos:

- aceleração constante $a = 1 m/s^2$ e $\theta(0) = 0,5 rad$ ou $\theta(0) = 3 rad$
- aceleração linear com o tempo $a = 0,5t\ m/s^2$ e $\theta(0) = 3 rad$

5. Quando seu corpo é exposto a vibrações, tais como quando passeando em um carro, pessoas que não possuem o pescoço suficientemente rígido freqüentemente respondem a este estímulo com severas dores de cabeça e tonturas. Um fabricante de carro quer projetar um novo carro, no qual estes problemas sejam minimizados . A Figura 31 mostra um modelo mecânico de um corpo humano sentado. As pernas não são modeladas, pois não contribuem para a potencial oscilação do corpo. Monte o modelo matemático do sistema. Considere que a entrada é uma força periódica com freqüência de 1 Hz e a saída de interesse é a distância entre a cabeça e o corpo. Faça uma simulação e comente os resultados.

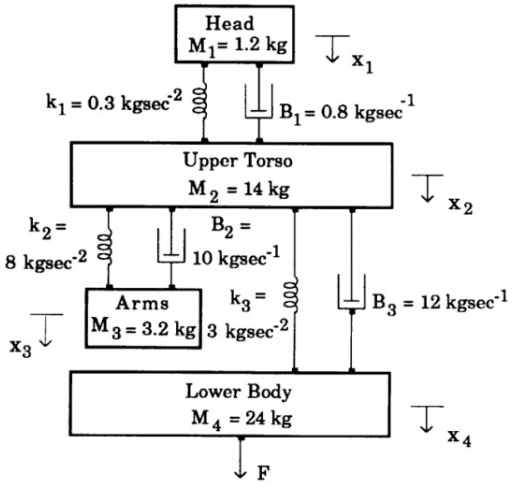



Figura 31: Modelo do corpo humano sentado. Dados médios para um humano adulto

# Parte V

# TRANSFORMADA DE LAPLACE

A transformada de Laplace permite a solução de uma equação diferencial ordinária de coeficientes constantes através da resolução de uma equação algébrica. .

# 10 NÚMEROS COMPLEXOS

## 10.1 O NÚMERO IMAGINÁRIO

O número imaginário $\sqrt{-1}$ já está definido no MATLAB pelas variáveis `i` ou `j`,

```
» j
ans =
        0.0000 + 1.0000i
» i
ans =
        0.0000 + 1.0000i
```

Observa-se que o sinal de multiplicação $< * >$ foi necessário depois da expressão `sqrt(3)`, em `z1`, mas não foi necessário depois do número 2, em `z2`.

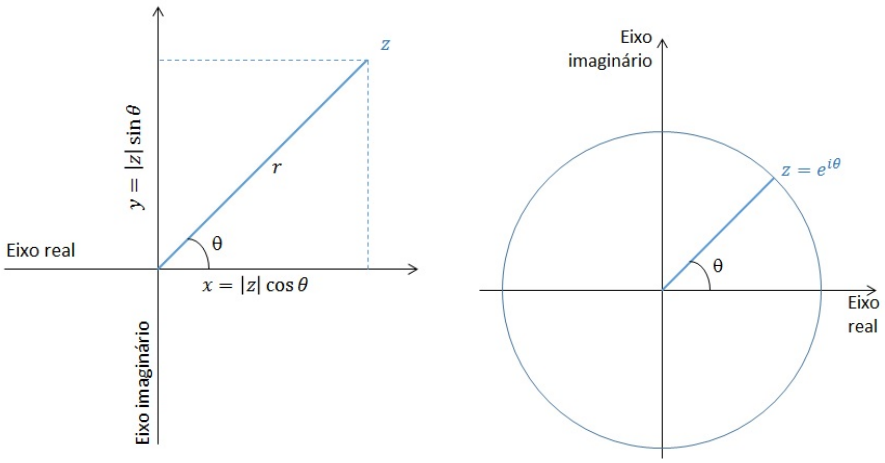
Um número complexo $z \in C$ pode ser escrito na pode ser definido pelo par ordenado $(x, y)$ de números reais $x$ e $y$, ou por suas coordenadas polares $r, \theta$,

$$z = x + yi \qquad z = re^{\theta i} = r(\cos\theta + i\sin\theta)$$

onde $x, y$ são a parte real e imaginária, respectivamente; e $r$ e $\theta$ são números reais e representam, respectivamente, o módulo e o ângulo ou fase de $z$,

$$r = |z| = \sqrt{x^2 + y^2}$$
$$\theta = \angle z = \arctan\frac{y}{x}$$

A representação gráfica de um número complexo $z \in C$ feita no plano complexo está ilustrada na Figura 32a.




(a) *Plano* s. (b) $z = e^{i\theta}$.

Figura 32: (a) O *Plano s*, em coordenadas cartesianas e polares; (b) circunferência de raio 1 centrada na origem do plano s.

Outro resultado bastante útil é,

$$|e^{i\theta}| = 1, \qquad \forall\theta$$

ou seja, $z = e^{i\theta}$ é um ponto de uma circunferência de raio 1, centrada na origem do plano s, cujo ângulo com o eixo real positivo é $\theta$ (Figura 32b).

Para definir um número complexo faz-se, por exemplo:

```
» z1 = -1 + sqrt(3)*i
z1 =
      -1.0000 + 1.7321i
»z2 = -1 - 2i
z2 =
      -1.0000 - 2.0000i
```

## 10.2 OPERAÇÕES MATEMÁTICAS

Seguem as regras básicas para operações de números complexos. Caso tenha alguma dúvida, recorra ao seu material das aulas de cálculo ou [2].

Dados os números complexos $z_1$ e $z_2$,

$$z_1 = x_1 + y_1 i \qquad z_2 = x_2 + y_2 i$$

a soma, subtração, multiplicação e divisão, são dados, respectivamente, por:

$$\begin{aligned}
z_1 + z_2 &= (x_1 + x_2) + i\,(y_1 + y_2) \\
z_1 - z_2 &= (x_1 - x_2) + i\,(y_1 - y_2) \\
z_1 z_2 &= (x_1 x_2 - y_1 y_2) + (x_1 y_2 + x_2 y_1)\,i \\
z_1 + z_2 &= \left(\frac{x_1 x_2 + y_1 y_2}{x_2^2 + y_2^2}\right) + i\left(\frac{y_1 x_2 - x_1 y_2}{x_2^2 + y_2^2}\right), \qquad z_2 \neq 0
\end{aligned}$$

Reescrevendo os números complexos $z_1$ e $z_2$,

$$z_1 = r_1\,(\cos\theta_1 + i\sin\theta_1) \qquad z_2 = r_2\,(\cos\theta_2 + i\sin\theta_2)$$

Pode-se provar que,

$$z_1 z_2 = r_1 r_2\,[\cos(\theta_1 + \theta_2) + i\sin(\theta_1 + \theta_2)]$$

Um caso particular, de interesse, seria o produto em que um dos complexos tem módulo unitário. Neste caso o resultado pode ser interpretado meramente como uma rotação de sua representação polar. Isto é, com $r_2 = 1$, tem-se:

$$z_1 z_2 = r_1\,[\cos(\theta_1 + \theta_2) + i\sin(\theta_1 + \theta_2)]$$

Ainda, exponenciação e logaritmo de um número complexo podem ser definidos como,

$$\begin{aligned}
e^z &= e^a\,(\cos y + i\sin x) \\
\ln z &= \ln r + i\theta
\end{aligned}$$

Assim:

```
» z3 = z1+z2
z3 =
      -2.0000 - 0.2679i
»z4 = z1 - z2
z4 =
      0.0000 + 3.7321i
»z5 = z1*z2
z5 =
      4.4641 + 0.2679i
»z6 = z1/z2
z6 =
      -0.4928 - 0.7464i
```

É sempre bom lembrar que...

```
»z7 = 1/i
z7 =
      0.0000 - 1.0000i
»z8 = 2/i
z8 =
      0.0000 - 2.0000i
»z9 = 1/(2j)^ 2
z9 =
      -0.2500
```

## 10.3 FUNÇÕES

Qualquer função existente no MATLAB pode ser utilizada tanto para números reais como para números complexos. As funções mais comuns são definidas pelos comandos `abs` (módulo), `angle` (fase ou ângulo), `exp` (exponencial), `log` (logaritmo neperiano), Além dessas funções tem-se as funções `real` (parte real de número complexo) e `imag` (parte imaginária de número complexo).

A seguir apresentam-se exemplos de utilização dessas funções:

```
»» z = -1 + i
z =
       -1.0000 + 1.0000i
»imag(z)
ans =
       1
»real(z)
ans =
       -1
»abs(z)
ans =
       1.4142
»angle(z)
ans =
       2.3562
```

Note que a unidade para ângulos utilizada pelo MATLAB é radianos. Se for desejado um ângulo em graus deve-se fazer a conversão fazendo-se:

```
» angle(z)*180/pi
ans =
       135
```

De maneira geral,

```
» exp(z)
ans =
       0.1988 + 0.3096i
» log(z)
ans =
       0.3466 + 2.3562i
```

Ou, finalmente, como discutido na Figura 32b,

```
» exp(0i)
ans =
       1
» exp(pi*i)
ans =
       -1.0000 + 0.0000i
» exp(pi/2*i)
ans =
       0.0000 + 1.0000i
» exp(-pi/2*i)
ans =
       0.0000 - 1.0000i
```

## 10.4 EXERCÍCIOS

1. Use MATLAB para calcular $e^{i\pi/3}$, $e^{1-i}$, $e^{-3\pi i/4}$.
2. Use as funções do MATLAB para calcular a forma polar dos números complexos $2-5i$, $3+7i$, $6+4i$.
3. Use as funções do MATLAB para converter os números complexos da forma polar para forma padrão: $4e^{5i}$, $-6e^{-3i}$, $2e^{\pi 2i}$.
4. Dado $w = 3e^{i\pi/3}$. Use as funções do MATLAB para calcular $w^2$, $w^3$, $1/w$ and $w+1$.

# 11 FUNÇÃO DE VARIÁVEL COMPLEXA

## 11.1 INTRODUÇÃO

Seja $s$ um número complexo qualquer pertencente a um conjunto $S$ de números complexos. Dizemos que $s$ é uma variável complexa. Se, para cada valor de $s$, o valor de outro número complexo $w$ é determinado, então $w$ é uma função de variável complexa $s$ no conjunto $S$:

$$w = F(s)$$

O conjunto $S$ é chamado de domínio de $F$. A função $F(s)$ pode ser expressa pela soma das suas componentes real e imaginária:

$$F(s) = F_x + iF_y$$

Sendo $F(s)$ um número complexo, obedece às mesmas definições e propriedades estabelecidas no Capítulo 10. Em particular:

- Valor absoluto de $F(s)$: $|F(s)| = \sqrt{F_x^2 + F_y^2}$
- Argumento de $F(s)$: $\theta_F = \tan\left(\frac{F_y}{F_x}\right)$

No que segue, utilizaremos uma definição da variável complexa, mais afeita aos desenvolvimentos relativos à teoria de sistemas dinâmicos e sistemas de controle:

$$s = \sigma + i\omega, \tag{10}$$

onde $\sigma$ é a parte real e $i\omega$ a parte imaginária da variável complexa.

## 11.2 LIMITE

Uma vizinhança de um ponto $z_0$ é o conjunto de todos os pontos para os quais:

$$|s - s_0| < \epsilon,$$

onde $\epsilon$ é alguma constante positiva. Portanto, uma vizinhança consiste em todos os pontos de um disco, ou região circular, no plano complexo, inclusive o centro $z_0$, mas, sem incluir o círculo de contorno.

Seja $F$ uma função definida em todos os pontos de uma vizinhança de um ponto $s_0$, exceto, eventualmente, o próprio ponto $s_0$. Dizemos que o limite de $F$, quando $s$ tende a $s_0$, é um número $w_0$, quando o valor de $F$ é arbitrariamente próximo de $w_0$ para todos os pontos $s$ de uma vizinhança de $s_0$, exceto, eventualmente, $s = s_0$, quando essa vizinhança se torna suficientemente pequena. De forma mais precisa,

$$\lim_{s \to s_0} F(s) = w_0$$

se, para cada número positivo $\epsilon$ existe um número positivo $\delta$ tal que:

$$|F(s) - w_0| < \epsilon, \text{sempre que } |s - s_0| < \delta \qquad (s \neq s_0)$$

**Teorema 1** *Sejam*

$$F(s) = u(\sigma, \omega) + iv(\sigma, \omega), \qquad s = \sigma + i\omega, \qquad s_0 = \sigma_0 + i\omega_0$$

*Então,*
*Existe o limite de* $F(s)$ *em* $s_0$ *e é igual a* $u_0 + iv_0$, $\lim_{s\to s_0} F(s) = u_0 + iv_0$, *se e somente se os limites de* $u$ *e* $v$ *existem em* $\sigma_0$ *e* $\omega_0$ *e são iguais a* $u_0$ *e* $v_0$, *respectivamente.*

**Teorema 2** *Sejam,* $F$ *e* $G$ *funções cujos limites existam em* $s_0$ *:*

$$\lim_{s\to s_0} F(s) = w_0 \qquad \lim_{s\to s_0} G(s) = W_0$$

*Então*

$$\lim_{s\to s_0} [F(s) + G(s)] = w_0 + W_0$$
$$\lim_{s\to s_0} [F(s)G(s)] = w_0 W_0$$
$$\lim_{s\to s_0} \left[\frac{F(s)}{G(s)}\right] = \frac{w_0}{W_0} \qquad W_0 \neq 0$$

## 11.3 CONTINUIDADE

Uma função $F$ é contínua num ponto $s_0$ se, e somente se, todas as três condições abaixo são satisfeitas:

$F(s_0)$ existe
$\lim_{s\to s_0} F(s)$ existe
$\lim_{s\to s_0} F(s) = F(s_0)$

## 11.4 DERIVADA E AS RELAÇÕES DE RELAÇÕES DE CAUCHY-RIEMAN

Suponha que,

$$F(s) = u(\sigma, \omega) + iv(\sigma, \omega)$$

onde, conforme já definido, $s = \sigma + i\omega$.

As relações de Cauchy-Rieman são dadas por (ver detalhes em [2]),

$$\frac{\partial u}{\partial \sigma} = \frac{\partial v}{\partial \omega} \qquad \text{e} \qquad \frac{\partial v}{\partial \sigma} = -\frac{\partial u}{\partial \omega}$$

Obedecer às *relações de Cauchy-Rieman* é condição necessária e suficiente para a existência da derivada de uma função em determinado ponto.

**Teorema 3** *Se a derivada* $F'(s)$ *de uma função* $F(s) = u(\sigma, \omega) + iv(\sigma, \omega)$ *existe em um ponto* $s_0$, *então as derivadas parciais de primeira ordem, em relação a* $\sigma$ *e* $\omega$, *de cada uma das partes* $u$ *e* $v$ *existem neste ponto e satisfazem às relações de Cauchy-Rieman. Além disso,* $F'(s)$ *é dada em termos dessas derivadas parciais de acordo com:*

$$\frac{dF(s)}{ds} = \frac{\partial u}{\partial \sigma} + i\frac{\partial v}{\partial \sigma} = \frac{\partial v}{\partial \omega} - i\frac{\partial u}{\partial \omega}$$

**Teorema 4** *Sejam* $u$ *e* $v$ *funções reais e univalentes das variáveis* $\sigma$ *e* $\omega$ *as quais, juntamente com suas derivadas parciais primeiras, são contínuas no ponto* $s_0$. *Se essas derivadas satisfazem às relações de Cauchy-Rieman neste ponto, então* $F'(s)$ *da função* $F(s) = u(\sigma, \omega) + iv(\sigma, \omega)$ *existe, sendo* $s_0 = \sigma_0 + i\omega_0$.

## 11.5 FUNÇÕES ANALÍTICAS

Uma função F de variável complexa s se diz analítica num ponto $s_0$, se sua derivada $F'(s)$ existe não só em $s_0$, como também em todo ponto s da vizinhança de $s_0$. F é analítica num domínio do plano complexo se ela é analítica em todo ponto desse domínio.

Se uma função é analítica em algum ponto de cada vizinhança de um ponto $s_0$ exceto no próprio ponto $s_0$, então o mesmo é chamado *ponto singular*, ou *singularidade da função*. Um ponto singular que resulta em F e suas derivadas tendendo a infinito é chamado de *polo da função*. Por exemplo, para

$$F(s) = \frac{1}{s^2 + 1}$$

Os pontos $s = i$ e $s = -i$ são polos de $F(s)$. Veremos que os polos possuem um papel importantíssimo na análise e projeto de sistemas dinâmicos.

Desde que as hipóteses dos teoremas da seção de derivadas sejam observadas num domínio D os seus resultados são suficientes para garantir que uma função F seja analítica nesse mesmo domínio.

Dadas duas funções analíticas F e G em um domínio D, sua soma é analítica em D, seu produto é analítico e D seu quociente é analítico no mesmo domínio desde que a função do denominador não se anule em D. Em particular, o quociente P/Q de dois polinômios é analítico em qualquer domínio no qual $Q(s) \neq 0$.

## 11.6 DERIVADAS NO MATLAB

As seguintes *functions*,

```
function [zderiv] = dds(f,x,y,z)
syms x y real;
syms s complex;
s = x + i*y;
s_deriv = (diff(f, 'x'))/2 - (diff(f,'y'))*i/2
end
```

```
function [zbarderiv] = ddsbar(f)
syms x y real;
syms s complex;
s = x + i*y;
sbar_deriv = (diff(f, 'x'))/2 + (diff(f, 'y'))*i/2
end
```

calculam, respectivamente, a derivada da função complexa f em função de s e de seu conjugado $\bar{s}$.

Por exemplo, após ativar as funções acima, digite as informações necessárias ao MATLAB, a fim de fazer cálculos complexos,

```
» syms x y real
» syms s complex
» s = x + i*y
```

Depois defina uma função,

```
» f = s^2
```

Finalmente digite `dds(f)`. O MATLAB irá fornecer uma resposta equivalente a $2(x + iy)$, que é a diferenciação de $s^2$ com respeito a s. Se, por outro lado, você

digitar no *prompt* do MATLAB, `ddsbar(f)`, você vai obter uma resposta 0, que é a diferenciação de $s^2$ com respeito a $\bar{s}$.

## 11.7 EXERCÍCIOS

1. Para praticar, dadas as funções $s$ e seu conjugado $\bar{s}$, calcule:

$$\frac{\partial}{\partial s} s^2 \bar{s}^3 \quad \frac{\partial}{\partial s} \sin s\bar{s} \quad \frac{\partial}{\partial \bar{s}} s^2 \bar{s}^3 \quad \frac{\partial}{\partial \bar{s}} e^{s\bar{s}^2}$$

2. Verifique se cada uma dessas funções obedece às *relações de Cauchy-Rieman* onde quer que seja definida:
   - $F(s) = \sin s - \frac{s^2}{s+1}$;
   - $F(s) = e^{2s-s^3} - s^2$;
   - $F(s) = \frac{\cos s}{s^2+1}$;
   - $F(s) = s\,(\tan s + s)$;

3. Verifique se cada uma dessas funções NÂO obedece às *relações de Cauchy-Rieman* onde quer que seja definida:
   - $F(s) = |s|^4 - |s|^2$;
   - $F(s) = \frac{\bar{s}}{s^2+1}$;
   - $F(s) = s\left(\bar{s}^2 - s\right)$;
   - $F(s) = \bar{s} \sin s \cos \bar{s}$;

4. The function $F(s) = s^2 - s^3$ obedece as relações de *Cauchy-Reiman*? A parte real $u$ descreve um fluxo de estado estacionário de calor em um disco unitário. Calcule a parte real $u$. Verifique se $u$ satisfaz a equação diferencial parcial,

$$\frac{\partial}{\partial s} \frac{\partial}{\partial \bar{s}} u(s) \equiv 0$$

   Essa é equação de Laplace.

# 12 TRANSFORMADA DE LAPLACE

## 12.1 INTRODUÇÃO

A Transformada de Laplace é um método operacional que pode ser utilizado para converter funções comuns, como senoidais, exponenciais, etc..., além de diferenciais e integrais, em funções algébricas de uma variável complexa $s$. A edição anterior do livro texto [3] tem o Capítulo 2 dedicado ao estudo da Transformada de Laplace. Aqui, apresentamos apenas as Tabela 15, com, respectivamente,os pares de transformadas de Laplace e as propriedades das transformadas.

| $f(t)$ | $F(s)$ |
|---|---|
| Impulso unitário $\delta(t)$ | $1$ |
| Degrau unitário $1(t)$ | $\frac{1}{s}$ |
| Rampa $t$ | $\frac{1}{s^2}$ |
| $\frac{t^{n-1}}{(n-1)!}$, $n = 1, 2, 3...$ | $\frac{1}{s^n}$ |
| $t^n$, $n = 1, 2, 3...$ | $\frac{n!}{s^{n+1}}$ |
| $e^{-at}$ | $\frac{1}{s+a}$ |
| $te^{-at}$ | $\frac{1}{(s+a)^2}$ |
| $\frac{t^{n-1}}{(n-1)!}t^{n-1}e^{-at}$, $n = 1, 2, 3...$ | $\frac{1}{(s+a)^n}$ |
| $t^{n-1}e^{-at}$, $n = 1, 2, 3...$ | $\frac{1}{(s+a)^{n+1}}$ |
| $\sin \omega t$ | $\frac{\omega}{s^2+\omega^2}$ |
| $\cos \omega t$ | $\frac{s}{s^2+\omega^2}$ |
| $\sinh \omega t$ | $\frac{\omega}{s^2-\omega^2}$ |
| $\cosh \omega t$ | $\frac{s}{s^2-\omega^2}$ |
| $\frac{1}{a}\left(1-e^{-at}\right)$ | $\frac{1}{s(s+a)}$ |
| $\frac{1}{b-a}\left(e^{-at}-e^{-bt}\right)$ | $\frac{1}{(s+a)(s+b)}$ |
| $\frac{1}{b-a}\left(be^{-bt}-ae^{-at}\right)$ | $\frac{s}{(s+a)(s+b)}$ |
| $\frac{1}{ab}\left[1+\frac{1}{a-b}\left(be^{-at}-ae^{-bt}\right)\right]$ | $\frac{1}{s(s+a)(s+b)}$ |
| $\frac{1}{a^2}\left(1-e^{-at}-ate^{-at}\right)$ | $\frac{1}{s(s+a)^2}$ |
| $\frac{1}{a^2}\left(at-1+e^{-at}\right)$ | $\frac{1}{s^2(s+a)}$ |
| $e^{-at}\sin \omega t$ | $\frac{\omega}{(s+a)^2+\omega^2}$ |
| $e^{-at}\cos \omega t$ | $\frac{s+a}{(s+a)^2+\omega^2}$ |
| $\frac{\omega_n}{\sqrt{1-\zeta^2}}e^{-\zeta\omega_n t}\sin\left(\omega_n\sqrt{1-\zeta^2}t\right)$, $(0<\zeta<1)$ | $\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$ |
| $-\frac{1}{\sqrt{1-\zeta^2}}e^{-\zeta\omega_n t}\sin\left(\omega_n\sqrt{1-\zeta^2}t-\phi\right)$, $\phi=\frac{\sqrt{1-\zeta^2}}{\zeta}$ $(0<\zeta<1, 0<\phi<\pi/2)$ | $\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$ |

Tabela 15: Códigos para cores, marcadores e tipos de linha em gráficos no MATLAB.

Seja $s$ um número complexo qualquer pertencente a um conjunto $S$ de números complexos. Dizemos que $s$ é uma variável complexa. Se, para cada valor de $s$, o valor de outro número complexo $w$ é determinado, então $w$ é uma função de variável complexa $s$ no conjunto $S$:

$$w = F(s)$$

O conjunto $S$ é chamado de domínio de $F$. A função $F(s)$ pode ser expressa pela soma das suas componentes real e imaginária:

$$F(s) = F_x + iF_y$$

Sendo $F(s)$ um número complexo, obedece às mesmas definições e propriedades estabelecidas no Capítulo 10. Em particular:

- Valor absoluto de $F(s)$: $|F(s)| = \sqrt{F_x^2 + F_y^2}$
- Argumento de $F(s)$: $\theta_F = \tan\left(\frac{F_y}{F_x}\right)$

No que segue, utilizaremos uma definição da variável complexa, mais afeita aos desenvolvimentos relativos à teoria de sistemas dinâmicos e sistemas de controle:

$$s = \sigma + i\omega, \tag{11}$$

onde $\sigma$ é a parte real e $i\omega$ a parte imaginária da variável complexa.

## 12.2 LIMITE

Uma vizinhança de um ponto $z_0$ é o conjunto de todos os pontos para os quais:

$$|s - s_0| < \epsilon,$$

onde $\epsilon$ é alguma constante positiva. Portanto, uma vizinhança consiste em todos os pontos de um disco, ou região circular, no plano complexo, inclusive o centro $z_0$, mas, sem incluir o círculo de contorno.

Seja $F$ uma função definida em todos os pontos de uma vizinhança de um ponto $s_0$, exceto, eventualmente, o próprio ponto $s_0$. Dizemos que o limite de $F$, quando $s$ tende a $s_0$, é um número $w_0$, quando o valor de $F$ é arbitrariamente próximo de $w_0$ para todos os pontos $s$ de uma vizinhança de $s_0$, exceto, eventualmente, $s = s_0$, quando essa vizinhança se torna suficientemente pequena. De forma mais precisa,

$$\lim_{s \to s_0} F(s) = w_0$$

se, para cada número positivo $\epsilon$ existe um número positivo $\delta$ tal que:

$$|F(s) - w_0| < \epsilon, \text{sempre que } |s - s_0| < \delta \qquad (s \neq s_0)$$

**Teorema 5** *Sejam*

$$F(s) = u(\sigma, \omega) + iv(\sigma, \omega), \qquad s = \sigma + i\omega, \qquad s_0 = \sigma_0 + i\omega_0$$

*Então,*
*Existe o limite de* $F(s)$ *em* $s_0$ *e é igual a* $u_0 + iv_0$, $\lim_{s \to s_0} F(s) = u_0 + iv_0$, *se e somente se os limites de* $u$ *e* $v$ *existem em* $\sigma_0$ *e* $\omega_0$ *e são iguais a* $u_0$ *e* $v_0$, *respectivamente.*

**Teorema 6** *Sejam,* $F$ *e* $G$ *funções cujos limites existam em* $s_0$ *:*

$$\lim_{s\to s_0} F(s) = w_0 \qquad \lim_{s\to s_0} G(s) = W_0$$

*Então*

$$\lim_{s\to s_0} [F(s) + G(s)] = w_0 + W_0$$
$$\lim_{s\to s_0} [F(s)G(s)] = w_0 W_0$$
$$\lim_{s\to s_0} \left[\frac{F(s)}{G(s)}\right] = \frac{w_0}{W_0} \qquad W_0 \neq 0$$

## 12.3 CONTINUIDADE

Uma função $F$ é contínua num ponto $s_0$ se, e somente se, todas as três condições abaixo são satisfeitas:

$F(s_0)$ existe
$\lim_{s\to s_0} F(s)$ existe
$\lim_{s\to s_0} F(s) = F(s_0)$

## 12.4 DERIVADA E AS RELAÇÕES DE RELAÇÕES DE CAUCHY-RIEMAN

Suponha que,

$$F(s) = u(\sigma, \omega) + iv(\sigma, \omega)$$

onde, conforme já definido, $s = \sigma + i\omega$.

As relações de Cauchy-Rieman são dadas por (ver detalhes em [2]),

$$\frac{\partial u}{\partial \sigma} = \frac{\partial v}{\partial \omega} \qquad \text{e} \qquad \frac{\partial v}{\partial \sigma} = -\frac{\partial u}{\partial \omega}$$

Obedecer às *relações de Cauchy-Rieman* é condição necessária e suficiente para a existência da derivada de uma função em determinado ponto.

**Teorema 7** *Se a derivada* $F'(s)$ *de uma função* $F(s) = u(\sigma, \omega) + iv(\sigma, \omega)$ *existe em um ponto* $s_0$*, então as derivadas parciais de primeira ordem, em relação a* $\sigma$ *e* $\omega$*, de cada uma das partes* $u$ *e* $v$ *existem neste ponto e satisfazem às relações de Cauchy-Rieman. Além disso,* $F'(s)$ *é dada em termos dessas derivadas parciais de acordo com:*

$$\frac{dF(s)}{ds} = \frac{\partial u}{\partial \sigma} + i\frac{\partial v}{\partial \sigma} = \frac{\partial v}{\partial \omega} - i\frac{\partial u}{\partial \omega}$$

**Teorema 8** *Sejam* $u$ *e* $v$ *funções reais e univalentes das variáveis* $\sigma$ *e* $\omega$ *as quais, juntamente com suas derivadas parciais primeiras, são contínuas no ponto* $s_0$*. Se essas derivadas satisfazem às relações de Cauchy-Rieman neste ponto, então* $F'(s)$ *da função* $F(s) = u(\sigma, \omega) + iv(\sigma, \omega)$ *existe, sendo* $s_0 = \sigma_0 + i\omega_0$*.*

## 12.5 FUNÇÕES ANALÍTICAS

Uma função $F$ de variável complexa $s$ se diz analítica num ponto $s_0$, se sua derivada $F'(s)$ existe não só em $s_0$, como também em todo ponto $s$ da vizinhança de $s_0$. $F$ é analítica num domínio do plano complexo se ela é analítica em todo ponto desse domínio.

Se uma função é analítica em algum ponto de cada vizinhança de um ponto $s_0$ exceto no próprio ponto $s_0$, então o mesmo é chamado *ponto singular*, ou *singularidade da função*. Um ponto singular que resulta em $F$ e suas derivadas tendendo a infinito é chamado de *polo da função*. Por exemplo, para

$$F(s) = \frac{1}{s^2 + 1}$$

Os pontos $s = i$ e $s = -i$ são polos de $F(s)$. Veremos que os polos possuem um papel importantíssimo na análise e projeto de sistemas dinâmicos.

Desde que as hipóteses dos teoremas da seção de derivadas sejam observadas num domínio D os seus resultados são suficientes para garantir que uma função $F$ seja analítica nesse mesmo domínio.

Dadas duas funções analíticas $F$ e $G$ em um domínio D, sua soma é analítica em D, seu produto é analítico e D seu quociente é analítico no mesmo domínio desde que a função do denominador não se anule em D. Em particular, o quociente $P/Q$ de dois polinômios é analítico em qualquer domínio no qual $Q(s) \neq 0$.

## 12.6 DERIVADAS NO MATLAB

As seguintes *functions*,

```
function [zderiv] = dds(f,x,y,z)
syms x y real;
syms s complex;
s = x + i*y;
s_deriv = (diff(f, 'x'))/2 - (diff(f,'y'))*i/2
end
```

```
function [zbarderiv] = ddsbar(f)
syms x y real;
syms s complex;
s = x + i*y;
sbar_deriv = (diff(f, 'x'))/2 + (diff(f, 'y'))*i/2
end
```

calculam, respectivamente, a derivada da função complexa `f` em função de `s` e de seu conjugado $\bar{s}$.

Por exemplo, após ativar as funções acima, digite as informações necessárias ao MATLAB, a fim de fazer cálculos complexos,

```
» syms x y real
» syms s complex
» s = x + i*y
```

Depois defina uma função,

```
» f = s^2
```

Finalmente digite `dds(f)`. O MATLAB irá fornecer uma resposta equivalente a $2(x + iy)$, que é a diferenciação de $s^2$ com respeito a $s$. Se, por outro lado, você digitar no *prompt* do MATLAB, `ddsbar(f)`, você vai obter uma resposta 0, que é a diferenciação de $s^2$ com respeito a $\bar{s}$.

## 12.7 EXERCÍCIOS

1. Para praticar, dadas as funções $s$ e seu conjugado $\bar{s}$, calcule:

$$\frac{\partial}{\partial s} s^2 \bar{s}^3 \quad \frac{\partial}{\partial s} \sin s\bar{s} \quad \frac{\partial}{\partial \bar{s}} s^2 \bar{s}^3 \quad \frac{\partial}{\partial \bar{s}} e^{s\bar{s}^2}$$

2. Verifique se cada uma dessas funções obedece às *relações de Cauchy-Rieman* onde quer que seja definida:
   - $F(s) = \sin s - \frac{s^2}{s+1}$;
   - $F(s) = e^{2s-s^3} - s^2$;
   - $F(s) = \frac{\cos s}{s^2+1}$;
   - $F(s) = s\,(\tan s + s)$;

3. Verifique se cada uma dessas funções NÂO obedece às *relações de Cauchy-Rieman* onde quer que seja definida:
   - $F(s) = |s|^4 - |s|^2$;
   - $F(s) = \frac{\bar{s}}{s^2+1}$;
   - $F(s) = s\left(\bar{s}^2 - s\right)$;
   - $F(s) = \bar{s} \sin s \cos \bar{s}$;

4. The function $F(s) = s^2 - s^3$ obedece as relações de *Cauchy-Reiman*? A parte real $u$ descreve um fluxo de estado estacionário de calor em um disco unitário. Calcule a parte real $u$. Verifique se $u$ satisfaz a equação diferencial parcial,

$$\frac{\partial}{\partial s}\frac{\partial}{\partial \bar{s}} u(s) \equiv 0$$

   Essa é equação de Laplace.

# Parte VI

# SIMULINK

O SIMULINK é uma ferramenta amigável, utilizada para Modelagem, Simulação e Análise de Sistemas Dinâmicos. O programa se aplica a sistemas lineares e não lineares, discretos e contínuos no tempo.

# 13 SIMULINK

O SIMULINK é um programa que funciona de forma integrada ao MATLAB, usado para modelagem e simulação de sistemas dinâmicos lineares ou não-lineares, em tempo contínuo, tempo discreto ou uma combinação dos dois modos. Os resultados das simulações podem ser visualizados, gravados em variáveis do MATLAB ou em arquivos de dados.

As simulações realizadas com os comandos de linha do MATLAB ou resolvendo-se as equações diferenciais do sistema, como foi visto em itens anteriores, são em geral muito mais simples de serem realizadas no SIMULINK.

## 13.1 ACESSANDO SIMULINK

Inicie o SIMULINK a partir da linha de comando do MATLAB digitando `simulink`, ou clicando no ícone do programa na barra de comandos do MATLAB (Figura 33). A janela principal do SIMULINK será exibida com as bibliotecas de blocos disponíveis para uso como mostra a Figura 34.


Figura 33: Ícone do SIMULINK.

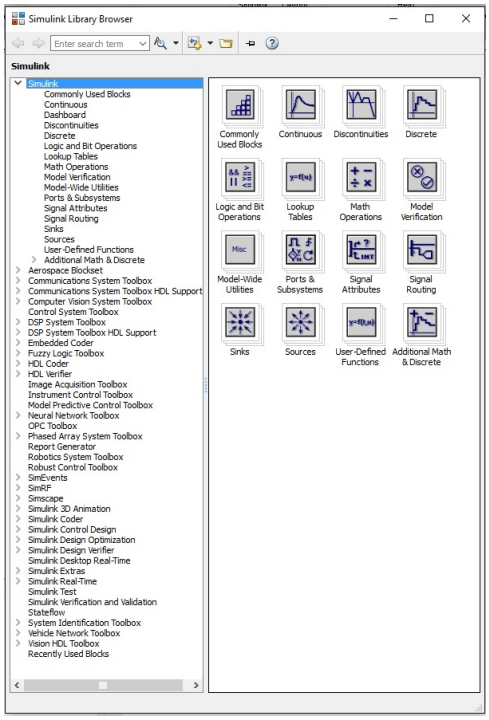



Figura 34: Biblioteca de blocos do SIMULINK.

Para abrir uma janela para edição de um novo modelo clique no ícone `New Model` na janela do SIMULINK, ou, se preferir, utilize a tecla de atalho `CTRL+N`. A Figura 35 a janela (untitled) aberta.

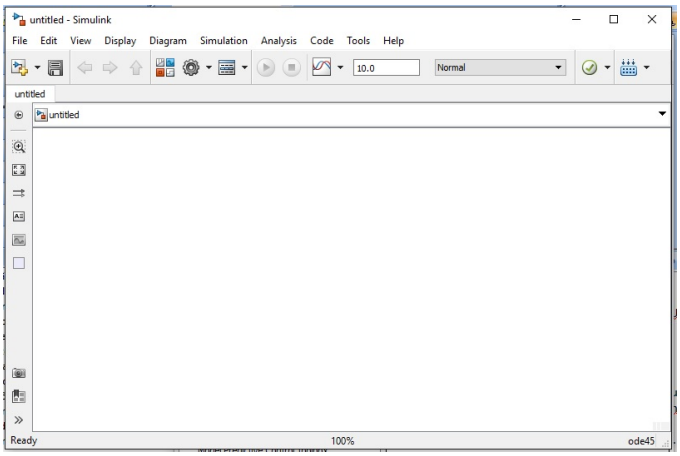



Figura 35: Área de trabalho.

Para armazenar o modelo clique em `File` e `Save` ou pressione `CTRL+S`. A extensão do arquivo é tipo do arquivo é `slx`.

## 13.2 COMPONENTES DE UM MODELO

Um modelo no SIMULINK consiste em três componentes: fontes, diagrama de blocos e saídas. As fontes são as entradas do sistema e estão presentes na biblioteca `Source`, o diagrama de blocos é a modelagem das equações do sistema; e as saídas são os blocos de verificação do comportamento e estão presentes na biblioteca `Sinks`.

### 13.2.1 *Fontes*

As fontes mais comuns são:

CONSTANT - bloco que produz um sinal uniforme. A magnitude pode ser escolhida com um duplo clique sobre o bloco;

STEP - produz uma função degrau. Pode-se configurar o instante em que se aplica o degrau, assim como sua magnitude antes e depois da transição.

SINE WAVE - gera uma senóide com os seguintes parâmetros a serem configurados: amplitude, fase e freqüência da onda senoidal.

SIGNAL GENERATOR - pode produzir ondas senoidais, quadradas, dente de serra ou sinais aleatórios.

A Figura 36 mostra a fonte `Step` sendo adicionada ao modelo. Basta arrastar da biblioteca à area de trabalho. Outros sinais podem ser gerados a partir de combinações destes blocos apresentados. Para criar um sinal de entrada personalizado, consulte, por exemplo, a referência [5].

### 13.2.2 *Diagrama de blocos*

O modelo do sistema contínuo está mostrado nos blocos da Figura 37. Verifique a opção de introdução de Equações de Estado ou Transformada de Laplace.

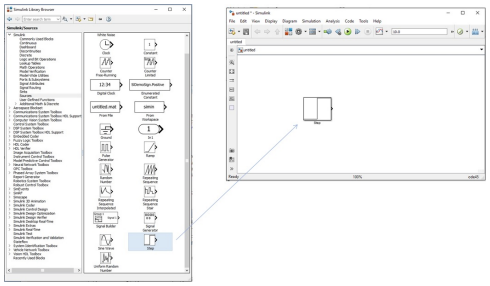


Figura 36: Geração de uma fonte no modelo.

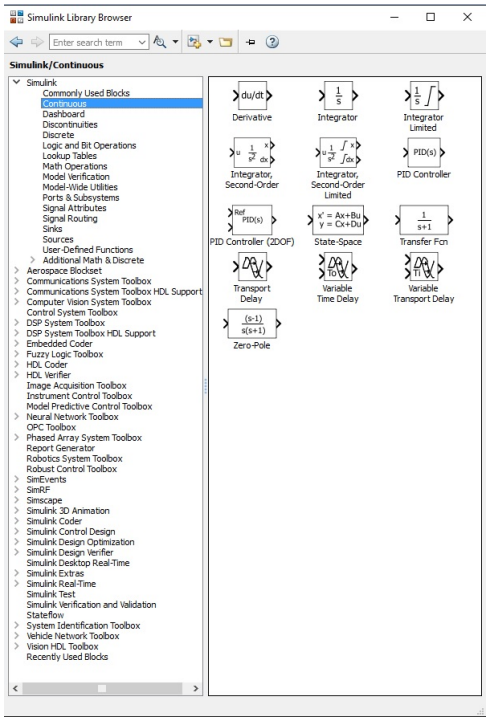



Figura 37: Diagrama de blocos.

### 13.2.3 *Saídas*

Os dispositivos de saída são os blocos que permitem verificar o comportamento do sistema, estes blocos são encontrados na biblioteca de dispositivos de saída (Sinks).

SCOPE O osciloscópio produz gráficos a partir de dados do modelo. Não existem parâmetros a serem configurados.

XY GRAPH O bloco de XY Graph produz um gráfico idêntico ao gráfico produzido pelo comando plot do MATLAB. Para isso, devem-se configurar os valores de mínimos e máximos, da horizontal e vertical.

DISPLAY O bloco Display produz uma amostragem digital do valor de sua entrada.

TO FILE Pode-se ainda armazenar os dados em arquivos do MATLAB para usos posteriores. Deve-se definir o nome do arquivo a ser criado.

TO WORKSPACE Pode-se ainda enviar os dados para a área de trabalho do MATLAB utilizando o bloco To Workspace Block. Deve-se definir o nome da matriz.

STOP SIMULATION O bloco de parada (Stop Simulation) causa a parada da simulação quando a sua entrada for diferente de zero.

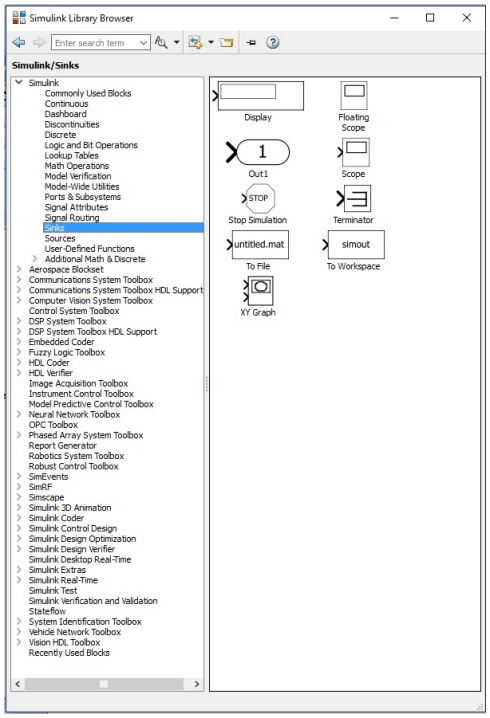



Figura 38: Geração do modo de saída do modelo.

## 13.3 SIMULANDO...

A criação de modelos no SIMULINK é feita de forma gráfica pelo posicionamento, interligação e configuração de blocos funcionais. Após carregar o SIMULINK e abrir a janela da área de trabalho, os itens abaixo mostram os passos para se

criar modelos de sistemas dinâmicos, através do uso de um gerador de sinais e de equações no espaço de estados. Para ilustrar, será mostrado como obter a simulação da resposta do sistema Massa Mola Amortecedor (MMA) ilustrado na Figura 39 a uma função degrau, conforme representado na Figura 40.

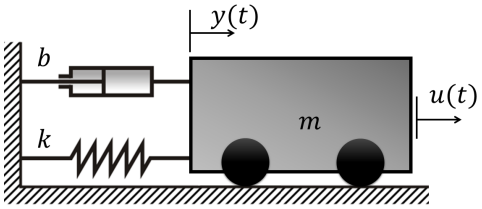



Figura 39: Sistema MMA.

Sendo a massa do corpo $m = 5kg$, coeficiente de amortecimento $b = 1Ns/m$ e a constante elástica da mola $k = 2N/m$, as matrizes do sistema, conforme definido no Capítulo 9, são,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2/5 & -1/5 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1/5 \end{bmatrix} u(t) \tag{12}$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

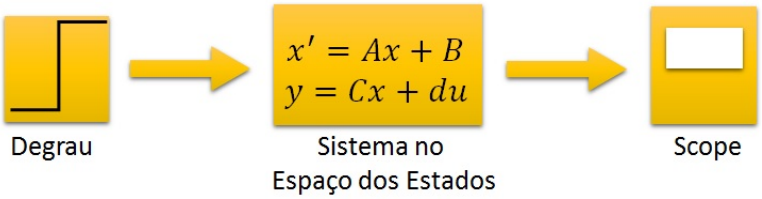



Figura 40: Modelo de sistema dinâmico no SIMULINK.

### 13.3.1 *Gerador de Sinais*

1. Insira o gerador de sinais,
   - Entre a lista de opções do `Simulink Library Browser`, selecione na biblioteca de blocos `Simulink`;
   - Escolha um tipo de bloco de fonte de sinal. Por exemplo, `Source`;
   - Selecione `Step` e arraste este bloco para a área de trabalho, conforme já ilustrado na Figura 36;
   - Dê um duplo clique sobre o signal generator ou clique com o botão direito e selecione os parâmetros de sua função `Step` (Figura 41).
2. Insira os blocos do sistema modelado:
   - Qualquer bloco no simulink pode ser pesquisado na linha de comando (Figura 42). O bloco é adicionado ao modelo clicando-se com o botão direito sobre o bloco e escolhendo-se a opção de adicionar ao arquivo (no caso, com nome *Exemplo1*).

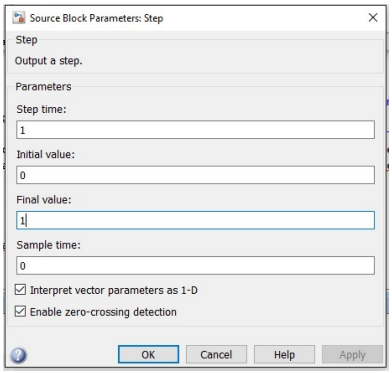



Figura 41: Setup da função `Step`.

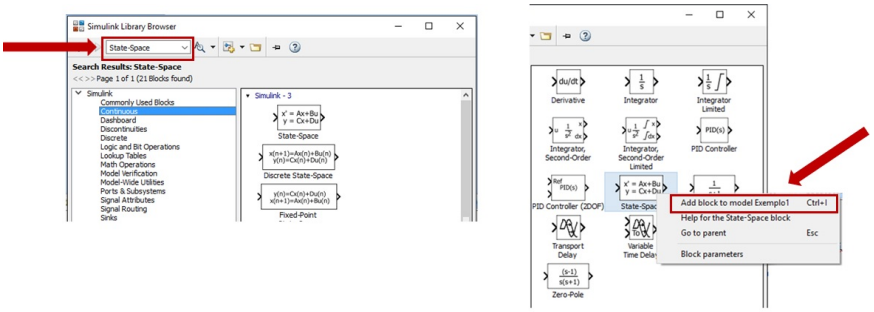



Figura 42: Como adicionar um bloco.

- Dê um duplo-clique no bloco `State-Space` editar suas propriedades. Após inserir as matrizes da mesma forma como é feito nas linhas de comando do MATLAB clique em OK (ver Figura 43).

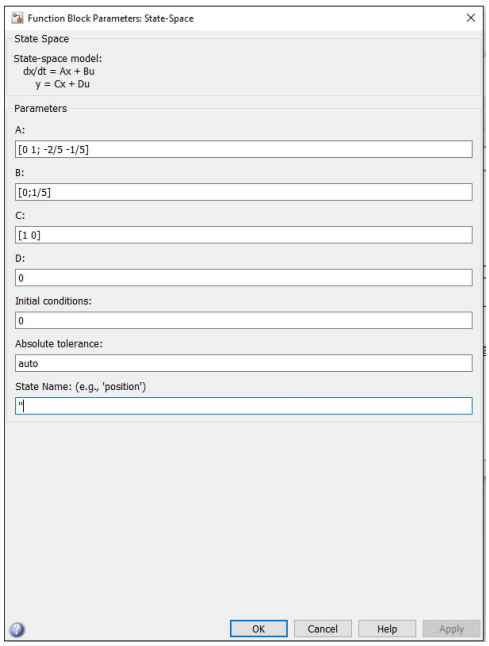



Figura 43: Propriedades conforme Equação 12.

3. Insira os dispositivos de saída, por exemplo, o `Scope` (osciloscópio),
   - Clique em `Commonly Used Blocks` ou `Sinks` e insira o `Source`.

4. Deve-se criar uma conexão entre os blocos,
   - Crie uma ligação entre o blocos posicionando o mouse sobre a saída do primeiro bloco e arraste o cursor (que muda para a forma de uma cruz) até a entrada do segundo bloco. Ao fazer isso a linha pontilhada se tornará contínua, com uma seta de direcionamento do primeiro ao segundo bloco. Outra opção para ligar os blocos é clicar no bloco de origem, segurar a tecla `ctrl` e clicar no bloco destino.
   - Repita o caminho com o mouse ligando os blocos;
   - Em nosso exemplo, complete as ligações até obter um modelo semelhante ao da Figura 44.

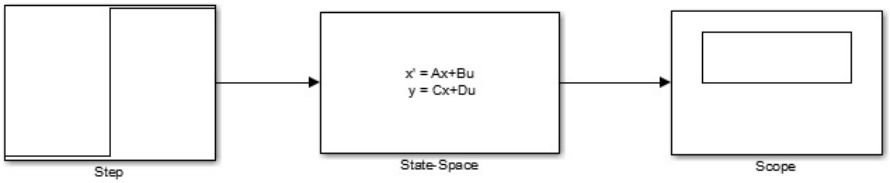



Figura 44: Modelo final da área de trabalho, de acordo com exemplo ilustrado na Figura 40.

5. Para realizar uma simulação de acordo com o desejado, deve-se antes configurar os parâmetros de simulação. Para isso clique no ícone de configuração e depois em `Model Simulation Parameter` e `Data Import/Export`, para acessar as mais importantes opções de simulação (veja Figura 45).

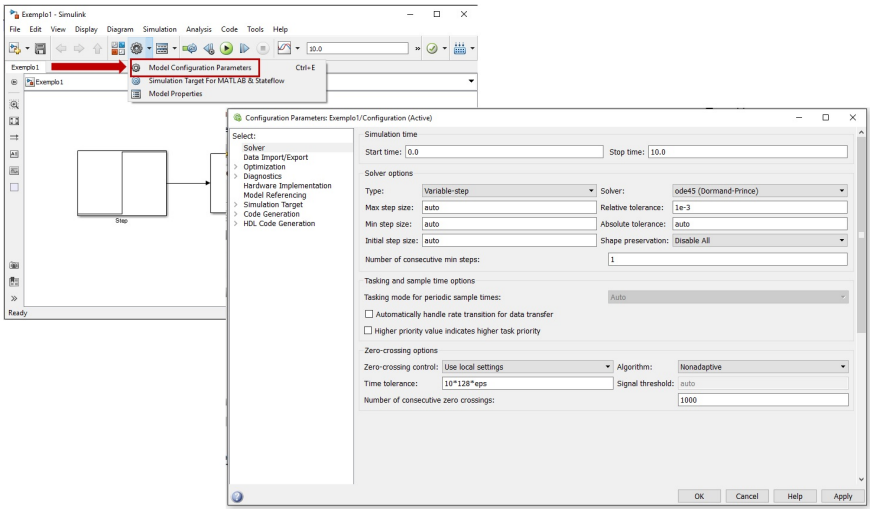



Figura 45: Parâmetros de simulaçao.

Vale a pena ressaltar algumas opções de interesse:

- `Simulation time` Refere-se ao intervalo de tempo em que a resposta dinâmica deve ser analisada. O tempo que a simulação leva para ser completada não é controlado, pois depende de fatores como a complexidade do modelo e capacidade de processamento do computador. Configure o intervalo de 80s para nosso exemplo.
- `Solvers` A simulação de um sistema dinâmico envolve a integração numérica de sistemas de equações diferenciais ordinárias. Para isso, o

SIMULINK oferece vários métodos de resolução com passos de integração fixos ou variáveis. Normalmente, o algoritmo de passo variável ode45, (já visto no Capítulo 9) fundamentado no método de Runge-Kutta, fornece bons resultados.

- `Step sizes` É possível controlar os valores dos passos de integração dos algoritmos de passo variável, como ode45. Como regra geral, pode-se deixar o controle desses valores a cargo do SIMULINK em uma primeira simulação e alterá-los caso os resultados obtidos não sejam adequados. De preferência, devem-se manter valores iguais para o passo máximo e inicial. Esta regra prática funciona de forma conveniente para a maioria dos problemas de simulação, embora não seja a única nem a mais adequada para todos os casos. Em muitas situações é possível melhorar os resultados de uma simulação ajustando-se o fator de refinamento da simulação, como será discutido adiante.
- `Tolerance` Os algoritmos de resolução usam técnicas de controle de erro a cada passo de simulação. Os valores estimados dos erros são comparados com um erro aceitável, definido pelos valores de `Relative tolerance` e `Absolute tolerance`, indicados na caixa de diálogo. Os algoritmos de passo variável reduzem o passo de integração automaticamente se o erro for maior que o aceitável. Em geral não é preciso alterar estes parâmetros.
- `Zero Crossing` O SIMULINK utiliza uma técnica conhecida como a *detecção de passagem por zero* ou *zero-crossing detection* para localizar com precisão uma descontinuidade sem recorrer a intervalos de tempo excessivamente pequenos. Normalmente, esta técnica melhora o tempo de simulação, mas pode, eventualmente, levar a uma parada de simulação antes do tempo de análise definido pelo usuário. Dois algoritmos de *zero-crossing detection* estão disponíveis: não adaptativo e adaptativo. Para obter informações sobre essas técnicas, consulte o manual do MATLAB, em *zero-crossing algorithm*.
- `Save options`, em `Data Import/Export`. Permite o controle dos instantes de tempo em que serão gerados os resultados da simulação. A opção mais útil é a do controle do fator de refinamento, `Refine factor`, que permite obter um número adicional de pontos de simulação entre aqueles que o algoritmo usaria normalmente. Por exemplo, se o fator de refinamento for definido como 5, cada passo de integração (de tamanho variável) será divido em 5 subintervalos. Na prática, é mais simples e eficiente (do ponto de vista computacional) melhorar os resultados de uma simulação aumentando o fator de refinamento do que reduzindo o tamanho do passo de integração.

Simule a resposta do modelo, clicando em `Simulation` e `Run` ou no ícone na barra de ferramentas, conforme ilustra a Figura 46. O programa avisa que a simulação terminou emitindo um beep e exibindo a palavra `Ready` na parte inferior da janela do modelo. Dê um duplo clique no bloco do osciloscópio (`Scope`) para ver a simulação do sinal de saída. O resultado deve ser como mostrado na Figura 47.

É possível alterar as escalas dos eixos a partir das opções de configuração do bloco (clicando com o botão direito do mouse em algum ponto do gráfico), mas geralmente basta clicar no botão de escala automática, indicado na Figura 47. O resultado final já está apresentado com o ajuste de escala automático.

Se o resultado da simulação parecer pouco preciso (o que você acha!?!) aumente o fator de refinamento (3 ou 5 costumam ser valores adequados) e simule novamente. Como padrão, o SIMULINK armazena o vetor de tempo usado na simulação em variável do workspace chamada `tout`. A criação desta variável, incluindo

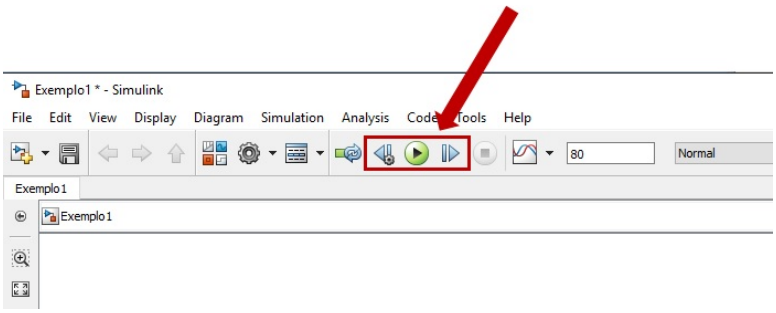



Figura 46: Para rodar o modelo.

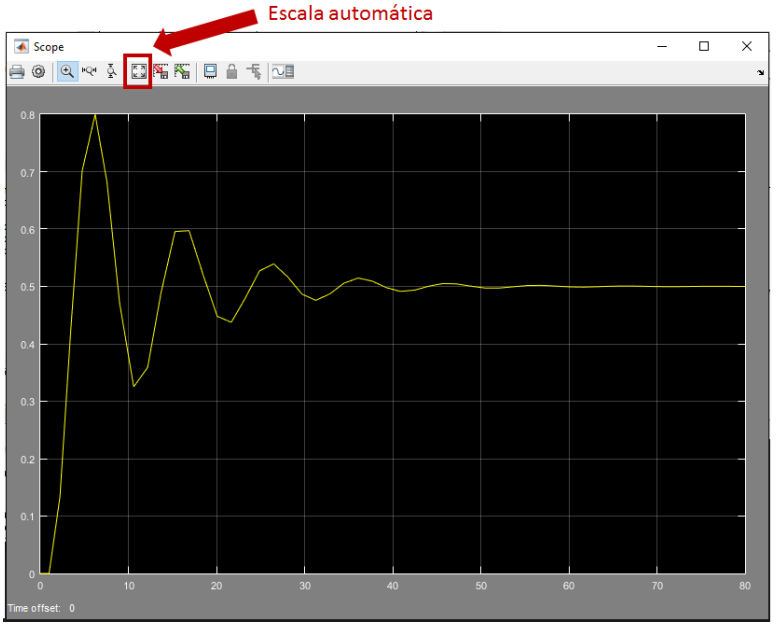



Figura 47: Para rodar o modelo.

seu nome, pode ser ajustada na aba `Data Import/Export` em `Configuration Parameters`.

Para enviar o resultado da simulação para a *workspace* do MATLAB deve-se incluir o bloco `To Workspace` (navegue pela biblioteca `Sinks` para obter esse bloco). É importante configurar o bloco `To Workspace` para gerar valores de saída no formato *array* (o formato padrão é *Structure*).

É possível também usar variáveis do workspace como entradas para sistemas do SIMULINK, usando o bloco `From Workspace` da biblioteca `Sources`.

Pode-se também resolver o exemplo da Figura 39 por Laplace. Tem-se,

$$m\ddot{y} = u(t) - b\dot{y} - ky$$

com condições iniciais $y(0) = 0$ e $\dot{y} = 0$

Realizando a Transformada de Laplace,

$$F(s) - scY(s) - kY(s) = ms^2Y(s)$$

a função de transferência resulta em,

$$G(s) = \frac{Y(s)}{F(s)} = \frac{\frac{1}{m}}{s^2 + s\frac{c}{m} + \frac{k}{m}}$$

Utiliza-se o bloco função de transferência `Transfer Fcn`, em `Continuous`. Deve-se preencher os parâmetros do bloco, Figura 48, com numerador $[1/5]$ e denominador $\left[11/52/5\right]$. A resposta do modelo deve ser idêntica àquela mostrada na Figura 47.

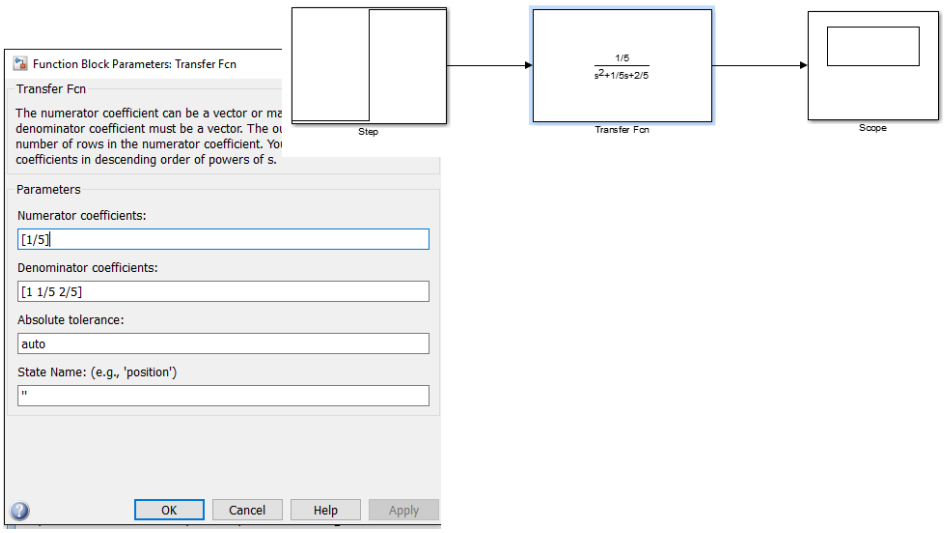



Figura 48: Modelo SIMULINK. Solução por Laplace.

## 13.4 EXERCÍCIOS

1. Resolva o sistema do exemplo MMA ilustrado na Figura 39, agora sem entrada no sistema, apenas com deflexão inicial $x_o = 1m$. Dica: use a opção `Integrator` duas vezes para achar velocidade e deslocamento.

2. Essa atividade consiste resolver a equação diferencial que representa a dinâmica de um sistema massa-mola-amortecedor não linear. Um sistema massa-mola-amortecedor não linear é representado pelas seguinte equação diferencial:

$$\ddot{x}(t) + 2\dot{x}^2(t) + 3\ln x(t) = u(t)$$

onde $x$ é a posição da massa, $v$ é a velocidade da massa e $u$ é a força aplicada na massa. Esse sistema pode ser escrito na forma $\dot{x}(t) = f(x, u, t)$ como segue:

$$\dot{x}(t) = v(t) = f_1(t, x, v, u)$$
$$\dot{v}(t) = -2v^2(t) - 3\ln x(t) + u(t) = f_2(t, x, v, u)$$

Simule o sistema para a condição inicial $x(0) = -0,1\text{m}$ e $v(0) = 0\text{m/s}$ e para a força $u(t)$ variando na forma de um degrau de amplitude igual a 50N no intervalo de tempo entre 0 e 10 segundos. Apresente como resultado o arquivo .m que implementa o vetor de funções f e os gráficos da posição, velocidade e força.

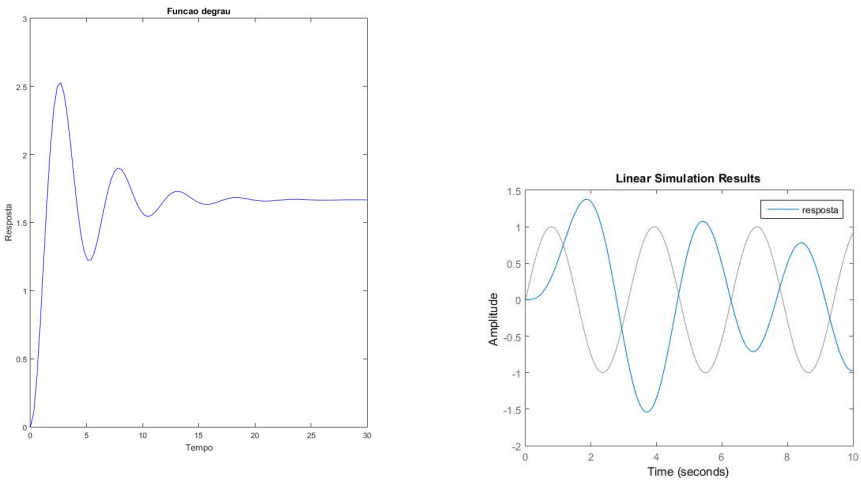
Parte VII

# APPENDIX

# GABARITO

## A.1 CAPÍTULO 8

1. A Figura 49 mostra as respostas em forma gráfica.




(a) *Função Degrau.* (b) *Função Senoidal.*

Figura 49: Resposta do sistema.

```
    %% Exercicios propostos cap. Sistemas Dinamicos Lineares
% Equacao diferencial: 2yddot+ydot+3y=5u

%% Defina o sistema no MATLAB usando uma variavel do tipo sys
m=2; b=1; k=3;
A=[0 1; -k/m -b/m];
B=[0 ; 1/m];
C=[1 0];
D=[0];
sys = ss(A,B,C,D);

%% Simule a resposta para uma funcao degrau unitario no ...
    intervalo 0-30s.
%Apresente os graficos da entrada degrau e da variavel y(t).

step(sys,30);

%% Defina um vetor de entrada senoidal com frequencia 2rad/s ...
    no intervalo
%0-10 s. Note que a funcao senoidal eh dada por sin(2t). Use ...
    um vetor de
%tempo com incremento de 0.01s.

t=0:0.01:10;
u=5.*sin(2*t);
lsim(sys,u,t);
legend('show');
```

2. Para os dados: m = 100Kg, b = 500Ns/m e k = 200N/m, a Figura 50 mostra a resposta em forma gráfica para função degrau (a outra fica por sua conta...).

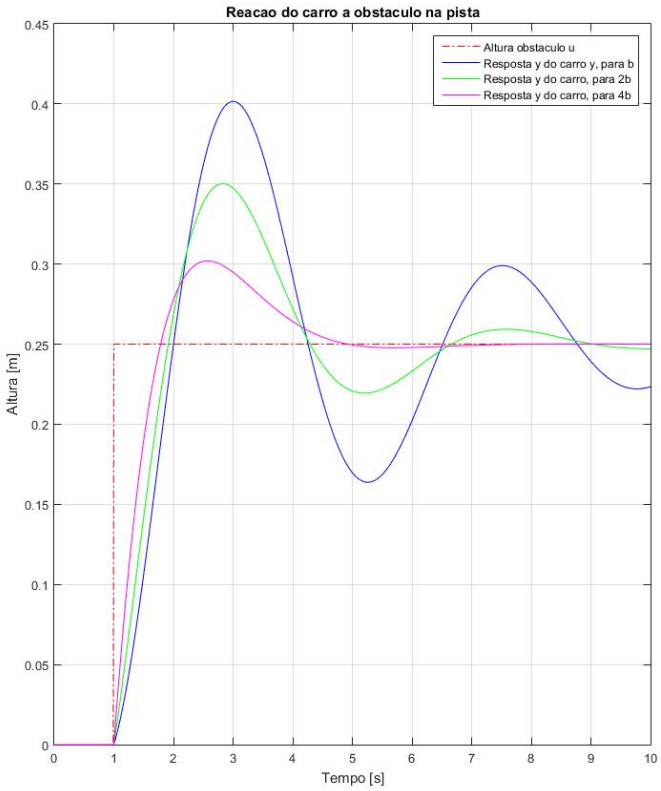



Figura 50: Modelo massa-mola-amortecedor simplificado de 1/4 de veículo, com dois casos de pista.

```
%% Exercicios propostos cap. Sistemas Dinamicos Lineares
% resposta do veiculo a obstaculos na pista

%%Parametros
m=1000;
k = 2000;
b = 500;
t = 0:0.01:10;
u = 0.25.*[zeros(1,100),ones(1,length(t)-100)];
p1=plot(t,u,'r-.');
hold on

%% Equacao diferencial:  ddot y + b dot y + k/m y = g + b/m ...
    dot u + k/m u
% y=lsim(b,a,u,t)
%
% b=[b_m,b_{m-1},b_{m-2},...,b_1,b_0] eh o vetor de coeficientes
%especificados no lado direito da equacao, que eh a equacao de ...
    interesse;
% a=[1, a_{n-1},a_{n-2},...,a_1,a_0] eh o vetor de ...
    coeficientes do lado
%esquerdo da equacao;
% u= eh o vetor de instantes conhecidos do sinal u(t) ...
    especificados
```

```
%na equacao;
% t= vetor da mesma dimensao de u, o k-esimo elemento t(k) de ...
    t eh o tempo,
%em segundos, no qual ocorre a entrada u(k);
% y= vetor da mesma dimensao de u e t que representa instantes ...
    do sinal
%$y(t)$ que satisfazem a equacao.

a1=b/m; a0=k/m; b1=b/m; b0=k/m;
a=[1,a1,a0];
b=[b1,b0]
y=lsim(b,a,u,t)
p2=plot(t,y,'b-');
grid on
hold on

%% Duplicando o amortecimento
m=1000;
k = 2000;
b = 2*500;
a1=b/m; a0=k/m; b1=b/m; b0=k/m;
a=[1,a1,a0];
b=[b1,b0]
y=lsim(b,a,u,t)
p3=plot(t,y,'g-');
hold on

%% Duplicando o amortecimento de novo....
m=1000;
k = 2000;
b = 4*500;
a1=b/m; a0=k/m; b1=b/m; b0=k/m;
a=[1,a1,a0];
b=[b1,b0]
y=lsim(b,a,u,t)
p4=plot(t,y,'m-');
xlabel('Tempo [s]');
ylabel('Altura [m]');
title('Reacao do carro a obstaculo na pista');

legend([p1,p2,p3,p4],'Altura obstaculo u',...
'Resposta y do carro y, para b','Resposta y do carro, para 2b',...
'Resposta y do carro, para 4b');
hold off

%% Solucao alternativa em espaco de estados:sistema sys
figure(2)
[A,B,C,D]=tf2ss(b,a)
sys=ss(A,B,C,D);
y = lsim(sys,u,t);
p5=plot(t,y);
%verifique que o resultado eh precisamente o mesmo utilizando-se
%lsim(b,a,u,t) ou lsim(sys,u,t)
```

## A.2 CAPÍTULO 9

1. A Figura 51 mostra as respostas em forma gráfica.

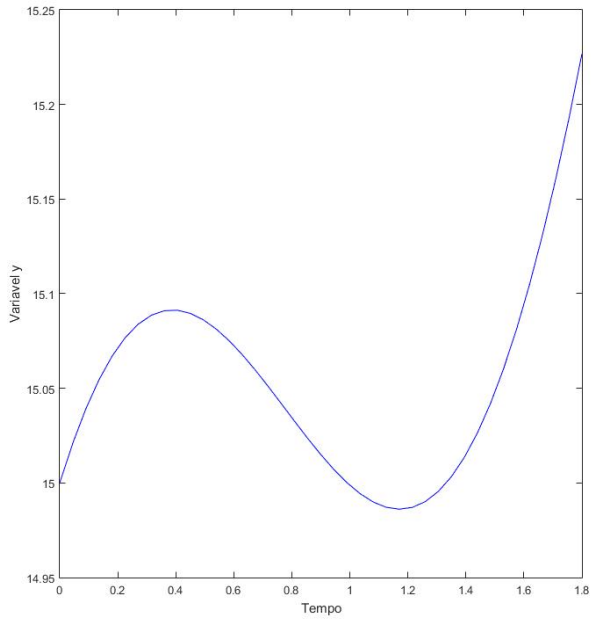



Figura 51: Resposta do sistema.

```
%% Equacao linear
% 10dy+y=20+7 sin(2t), y(0)=15

%% Solucao
[t,y]=ode45(@Func_ex1, [0, 1.8],15);
plot(t,y,'b-'),xlabel('Tempo'),ylabel('Variavel y');
```

```
function  ydot=Func_ex1(t,y);
%Function para resolver o primeiro exercicio proposto
 ydot=(20-7*sin(2*t)-y)/10;
end
```

2. A Figura 52 mostra a variação da temperatura no tempo.

```
%% Equacao linear
% 10 dot T+T=Tb, Tb=170 Celsius, T(0)=70 Celsius

%% Solucao
[t,y]=ode45(@Temp, [0,100],70);
plot(t,y,'b-'),xlabel('Tempo [s]'),ylabel('Temperatura ...
    [Celsius]');

%% Constantes
[t,y]=ode45(@Temp, [0,1000],70);

%Valor estacionario
Tinf=y(length(t))

%Tempo de assentamento
```

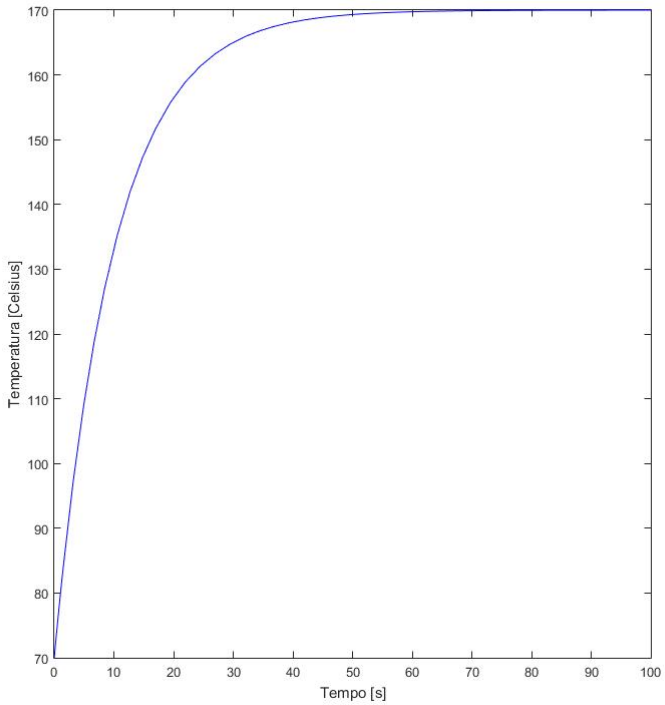



Figura 52: Temperatura no tempo.

```
aux=0;
for icont=1:length(t)
    if y(icont) >= 0.8*Tinf & aux==0
        aux=1;
        tassent=t(icont);
    end
end
tassent

%Tempo de subida
aux1=0;
aux2=0;
for icont=1:length(t)
    if y(icont) >= 0.2*Tinf & aux1==0
        aux1=1;
        t20=t(icont);
    end
    if y(icont) >= 0.9*Tinf & aux2==0
        aux2=1;
        t90=t(icont);
    end
end
tsubida=t90-t20

%Resposta e tempo de pico
Tpico=0;
for icont=1:length(t)
    if y(icont) >= Tpico
        tpico=t(icont);
        Tpico=y(icont);
    end
end
tpico
Tpico
```

```
function  ydot=Temp(t,y);
%Function para resolver o primeiro exercicio proposto
 ydot=(170-y)/10;
end
```

3. As soluções numérica (MATLAB) e analítica são mostradas na Figura 53.

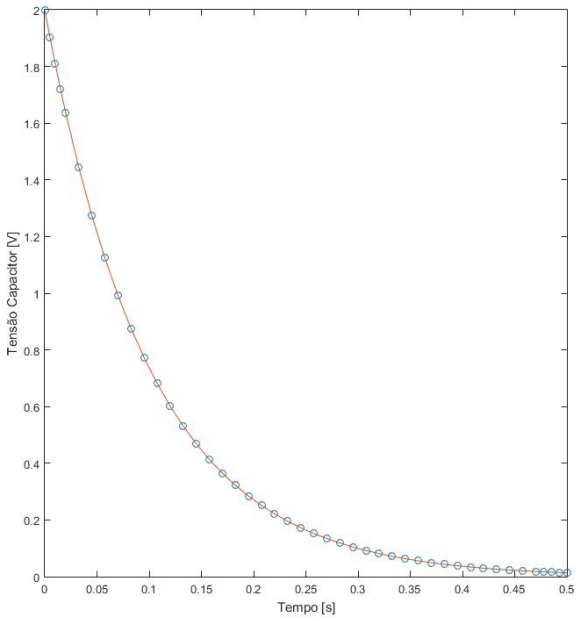


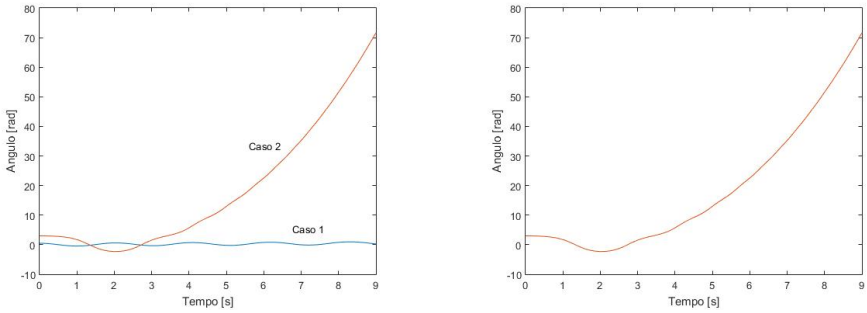

Figura 53: Solução numérica e analítica para o circuito RC.

```
%% Circuito RC, exemplo 9.3-1, pg. 384, livro Palm III

% Solucaoo circuito RC ydot+y=v(t)
% Solucao analitica: y(t)=2e^(-10t);

[t,y]=ode45(@RC_circuit, [0, 0.5],2);
y_true=2*exp(-10*t);
plot(t,y,'o',t,y_true),xlabel('Tempo [s]'),ylabel('Tensao ...
    Capacitor [V]');
```

```
function  ydot=RC_circuit(t,y);
%Modelo de um circuito RC sem nenhuma tensao aplicada
 ydot=-10*y;
end
```

4. A Figura 54 mostra a resposta no tempo.

```
%% Caso 1: aceleracao constante
[ta,xa]=ode45(@pend2,[0,9],[0.5,0]);
[tb,xb]=ode45(@pend2,[0,9],[3,0]);
plot(ta,xa(:,1),tb,xb(:,1));xlabel('Tempo [s]'); ...
    ylabel('Angulo [rad]');...
    gtext('Caso 1'),gtext('Caso 2');

%% Caso 2: aceleracao linear
```




(a) *Aceleração constante.* (b) *Aceleração linear.*

Figura 54: Exemplo de uso do mapa de cores.

```
[ta,xa]=ode45(@pend2,[0,9],[3,0]);
figure(2)
plot(ta,xa(:,1),tb,xb(:,1));xlabel('Tempo [s]'); ...
    ylabel('Angulo [rad]');
```

```
function xdot=pend2(t,x)
 g=9.81;  L=1;  acel=0.5*t %acel=1;
 xdot=[x(2); (1/L)*(acel-g*sin(x(1)))];
end
```

# Parte VIII

# APOIO E REFERÊNCIAS BIBLIOGRÁFICAS

If I have seen further, it is by standing upon the shoulders of giants.
Sir Isaac Newton

# BIBLIOGRAFIA


[1] R J Braun. Beginning Matlab Exercises. Technical report, Department of Mathematical Sciences - University of Delaware, 2008.

[2] R V Churchill. *Variáveis Complexas e suas Aplicaćões*. McGraw Hill, 1975.

[3] Katsihiko Ogata. *Engenharia de Controle Moderno*. Pearson Education, 5 edition, 2010.

[4] William J Palm III. *Introdução ao {MATLAB} para engenheiros*. Mc Graw Hill, third edition, 2013.

[5] UFES PET Engenharia de Computação. Mini-curso de Simulink, 2009. URL pet.inf.ufes.br/{~}pet/pet{_}site/matlab-octave/controle/Simulink.pdf.

[6] Singiresu Rao. *Vibrações Mecânicas*. Pearson Education, 4 edition, 2008.