SISTEMAS DINÂMICOS PARA MECATRÔNICA I

Aula de Laboratório

LARISSA DRIEMEIER E MARCÍLIO ALVES

Universidade de São Paulo

Escola Politécnica

CONTEÚDO

Parte I

AULA 01 - HELLO MODELLING WORLD!

Os novos recursos computacionais disponíveis contribuem comprovadamente para o sucesso e progresso da simulação de sistemas dinâmicos. Porém, existem conceitos obrigatórios na utilização dessa tecnologia. Muitos profissionais que iniciam suas aplicações encontram dificuldades, pois o aprendizado de uso de software é feito sem base conceitual, confundindo o aprendizado de manuseio de programa com o conhecimento teórico. Justifica-se portanto, a filosofia de abordagem do Prof. Avelino, [8]: *Se o engenheiro não sabe modelar o problema sem ter o computador, ele não deve fazê-lo tendo o computador!* 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 MODELAMENTO

Freeman John Dyson, físico e matemático inglês que trabalhou para o British Bomber Command durante a Segunda Guerra Mundial, descreve sua conversa, em 1953, com Enrico Fermi, físico italiano naturalizado estadunidense que se destacou pelo seu trabalho sobre o desenvolvimento do primeiro reator nuclear, e recebeu o Prêmio Nobel de Física em 1938:

> ... I asked Fermi whether he was not impressed by the agreement between our calculated numbers and his measured numbers. He replied, *How many arbitrary parameters did you use for your calculations?* I thought for a moment about our cut-off procedures and said, *Four.* He said, *I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk.*

O mundo é tridimensional, dinâmico e não linear. Se esta afirmação fosse levada ao pé da letra, poucos problemas seriam resolvidos em engenharia. Na verdade, dependendo do problema, algumas grandezas são menos importantes que outras e podem ser desprezadas, simplificando-se a solução. Assim, neste âmbito, modelar um problema significa adotar hipóteses que o simplifiquem e permitam equacioná-lo para se chegar a uma solução, ainda que aproximada.

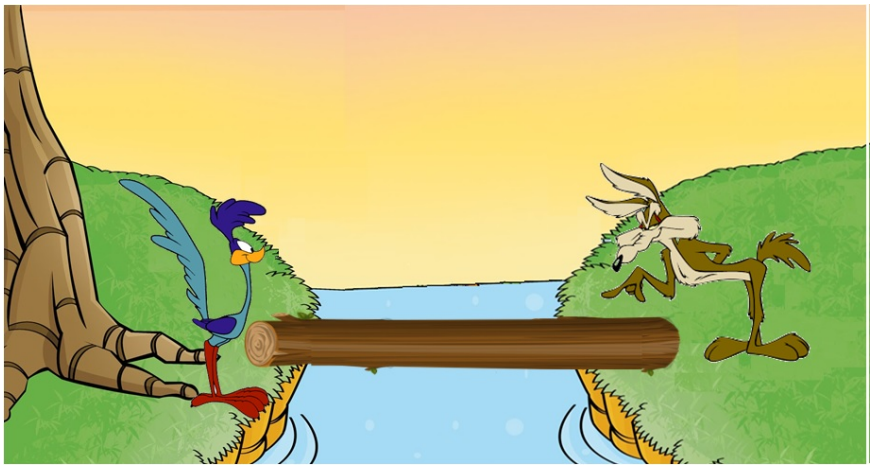
A título de exemplo, tomemos o seguinte problema exemplo: um trem sai da cidade A com uma velocidade média estimada de 100 Km/h, dirigindo-se à cidade B distante, em linha reta, cerca de 100 km. Em quanto tempo o trem chegará na cidade B? Se o engenheiro, na solução deste problema, tiver que se preocupar com variações de velocidade e com as curvas da via férrea ao longo do caminho, encontrar uma solução satisfatória poderia ser uma tarefa difícil se não impossível. Se, por outro lado, ficarmos satisfeitos com uma resposta *menos precisa*, o modelo de cálculo fica bastante simplificado e a resposta é simplesmente: cerca de 1 hora.

Claro que nem todos os problemas são tão simples. A resposta procurada no exemplo anterior é um escalar, mas há problemas que necessitam de soluções vetoriais, enquanto outros necessitam de soluções em forma de uma função. O papel do engenheiro, portanto, será de *construir* um modelo, a partir de um problema que não possui solução exata, e achar uma solução aproximada ótima.

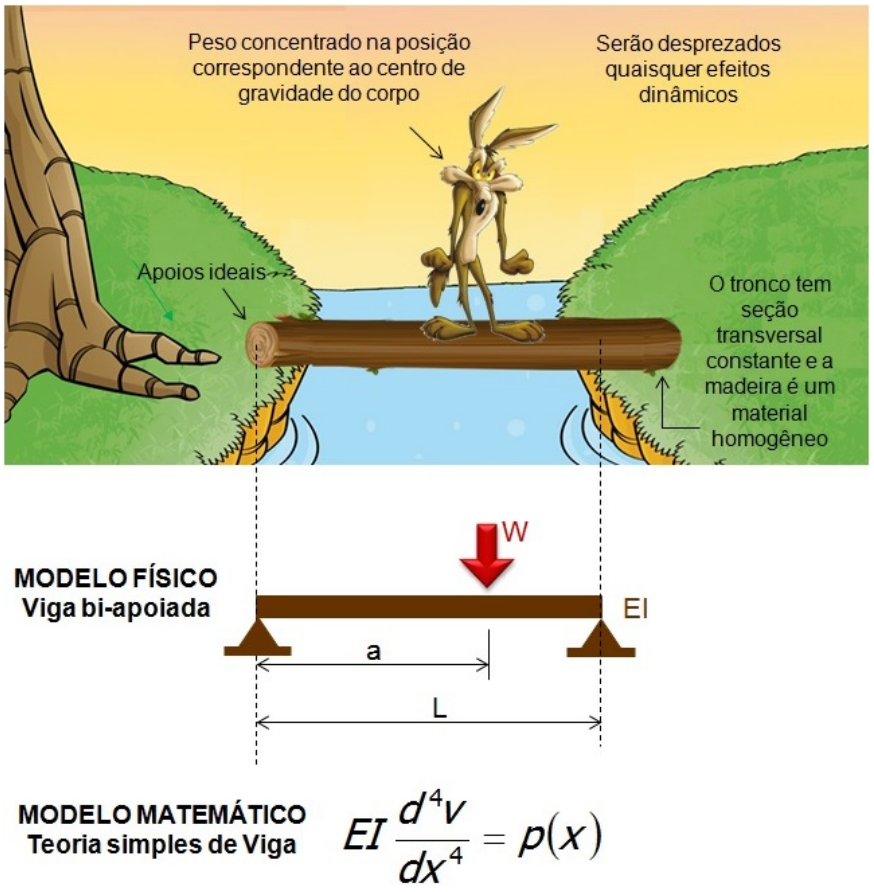
Imagine que um tronco de árvore seja colocado sobre um rio com o intuito de servir como um ponte improvisada e provisória, como mostra a Figura 1a. Que tipo de avaliação pode ser feita no sentido de dar alguma confiança à pessoa que deseja fazer a travessia?

Para resolver o problema, algumas hipóteses precisam ser inicialmente formuladas (Figura 1b), tais como:

1. O tronco tem seção transversal constante e a madeira é homogênea;
2. O tronco se encontra sobre apoios ideais, firmemente assentado sobre as margens;
3. O peso do transeunte é suposto concentrado na posição correspondente ao centro de gravidade de seu corpo;
4. Serão desprezados quaisquer efeitos dinâmicos, pressupondo-se que a pessoa, até porque deverá estar receosa, se moverá lentamente.



(a) *Sistema real.*



(b) *Modelo.*

Figura 1: Modelamento em Engenharia.

Com essas hipóteses, *tanto quanto possível* apenas uma versão simplificada da realidade, está formulado um **modelo físico**, representando os aspectos mais importantes do sistema, correspondente a uma viga biapoiada. É preciso ter uma boa capacidade de discernimento em termos de engenharia para propor um modelo físico adequado. O ideal em primeiro lugar usa-se um modelo mais elementar, que será refinado gradativamente para obter resultados tão mais precisos quanto necessáio. Vale ainda ressaltar que, em decorrência das simplificações, os resultados deverão ser interpretados à luz de apropriado coeficiente de segurança pois, como é óbvio notar, em quase todas as hipóteses existe razoável margem de incerteza.

O próximo passo consiste em adotar um **modelo matemático** que conduza ao equacionamento do modelo físico proposto. As equações que descrevem o sistema estão normalmente na forma de um conjunto de equações diferenciais parciais.

O problema proposto na Figura 1 é muito bem representado pela Teoria Simples de Viga que, nunca é de mais lembrar, possui, e introduz, novas hipóteses simplificadoras, tais como:

1. Pequenos deslocamentos e deformações;
2. O eixo neutro é inextensível na flexão;
3. Secções planas permanecem planas;
4. Apenas deformações por flexão são contabilizadas, desprezando-se a deformação por cisalhamento.

Por essa teoria, uma carga distribuída sob o vão livre (a exemplo do peso próprio do tronco) relaciona-se com a flecha (deslocamento lateral como resultado da flexão) através da seguinte equação diferencial:

$$EI\frac{d^4v}{dx^4} = p(x) \tag{1}$$

que é obtida dos seguintes resultados conhecidos da Resistência dos Materiais :

$$\begin{aligned} p(x) &= \frac{dQ}{dx} \\ Q(x) &= \frac{dM}{dx} \\ M(x) &= EI\frac{d^2v}{dx^2} \end{aligned} \tag{2}$$

onde, por definição,

- $v(x)$ deslocamento lateral ou flecha mm;
- $M(x)$ momento fletor Nmm;
- $Q(x)$ esforço cortante N;
- $p(x)$ carregamento distribuído N/mm (peso próprio da viga, por exemplo);
- E módulo de elasticidade do material $N/mm^2 = MPa$;
- I inércia à flexão da seção transversal mm^4;
- EI rigidez à flexão Nmm^2.

O carregamento W entra como condição de contorno. A resolução da equação diferencial (1), de equilíbrio estático de uma viga em flexão com um peso concentrado W na posição $x = a$, leva a uma família de duas funções com quatro constantes de integração cada, constantes estas que somente serão conhecidas

quando forem aplicadas as condições de contorno arbitradas no modelo físico. Assim, tomando-se a viga biapoiada e sendo v_1 a flecha para o trecho $0 \leqslant x \leqslant a$, à esquerda da carga concentrada W e, analogamente, v_2 a flecha para o trecho à direita $a \leqslant x \leqslant L$, temos:

1. $v_1(x = 0) = 0$, flecha nula no apoio à esquerda;
2. $v_2(x = L) = 0$, flecha nula no apoio à direita;
3. $M_1(x = 0) = 0$, momento nulo no apoio à esquerda;
4. $M_2(x = 0) = 0$, momento nulo no apoio à direita;
5. $v_1(x = a) = v_2(x = a)$, flechas iguais no ponto da carga;
6. $v_1'(x = a) = v_2'(x = a)$, rotações iguais no ponto da carga;
7. $M_1(x = a) = M_2(x = a)$, momentos iguais no ponto da carga;
8. $Q_1(x = a) - Q_2(x = a) = W$, balanço de forças verticais no ponto da carga.

A solução do problema assim formulado é um caso clássico da Resistência dos Materiais e não será aqui visto em detalhes. Por se tratar de estrutura isostática, é possível, alternativamente, obter o equilíbrio diretamente das equações da estática no plano. Assim, se a título de uma nova simplificação, desprezarmos o efeito do peso próprio, o máximo momento fletor ocorre no ponto de aplicação da carga concentrada, valendo:

$$M = Wa\left(1 - \frac{a}{L}\right) \tag{3}$$

do qual se pode calcular a máxima tensão normal na flexão, σ_{max}, para compará-la com a tensão admissível:

$$\sigma_{max} = \frac{M}{I}y \leqslant \sigma_{ad} = \frac{\sigma_r}{\eta} \tag{4}$$

onde,

- y é a distância da fibra mais externa ao eixo neutro;
- σ_{ad}, σ_r são, respectivamente, a tensão admissível máxima e a tensão de ruptura da madeira;
- η é o coeficiente de segurança (> 1).

Se a desigualdade expressa por (4) for satisfeita para um coeficiente de segurança, digamos, igual a 4, é razoável admitir que a travessia possa ser feita sem sobressaltos. Caso contrário, dadas todas as incertezas do cálculo, é mais sensato não a recomendar. Note-se que, mesmo nesta parte final da avaliação, novas incertezas aparecem, a exemplo do valor da tensão de ruptura que será *arbitrado* para a madeira. Com base em dados encontrados em catálogos, vê-se que esse valor varia muito como função do tipo e estado da madeira: admitir qualquer coisa acima de $0,5\ Kgf/cm^2$ (1/8 do valor médio do aço) já seria temerário.

1.1 MODELAMENTO EM SISTEMAS DINÂMICOS

O modelamento clássico considerado neste curso *discretiza o sistema* em elementos básicos,

- Elementos de armazenamento de energia:

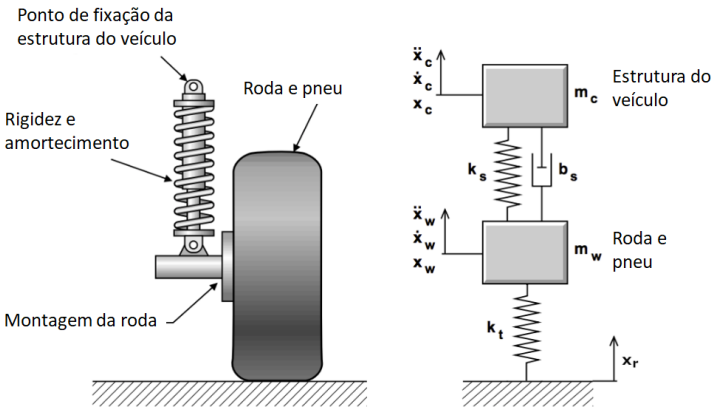


Figura 2: Modelamento em Engenharia - 1/4 de um veículo.

  - Elementos de inércia (massas);
  - Elementos de rigidez (molas).
- Elementos de dissipação de energia:
  - Amortecedor;
  - Atrito.

Por exemplo, a Figura 2 mostra a suspensão de 1/4 do veículo, que pode ser modelada aplicando-se a segunda lei de Newton às massas m_w e m_c, respectivamente,

$$\begin{aligned} m_w\ddot{x}_w + b_s\dot{x}_w + (k_t + k_s)\,x_w &= b_s\dot{x}_c + k_s x_c + k_t x_r \\ m_c\ddot{x}_c + b_s\dot{x}_c + k_s x_c &= b_s\dot{x}_w + k_s x_w \end{aligned} \tag{5}$$

O modelo matemático, como mencionado, envolve a solução de equações diferenciais parciais. Soluções analíticas exatas (fechadas) existem em casos especiais de geometria e condições de contorno simples, para certos tipos de carregamento e material homogêneo (como nosso caso exemplo). Algumas árduas soluções analíticas para materiais anisotrópicos e geometrias complicadas aparecem na literatura, como herança de uma época em que os recusros computacionais eram escassos.

Quando o problema se torna complexo, ao invés de seguir pelo médoto sugerido na Figura 3, o engenheiro deve utilizar **métodos numéricos**, que são aproximações dos modelos matemáticos.

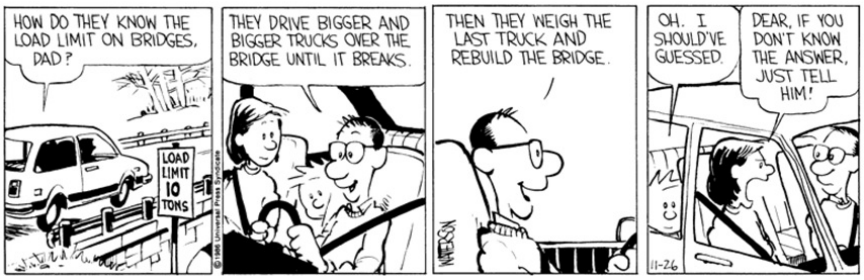


Figura 3: Calvin na engenharia.

Por fim, a solução das equações governantes fornece os deslocamentos, velocidades e acelerações das várias massas do sistema. Simulação e modelagem por

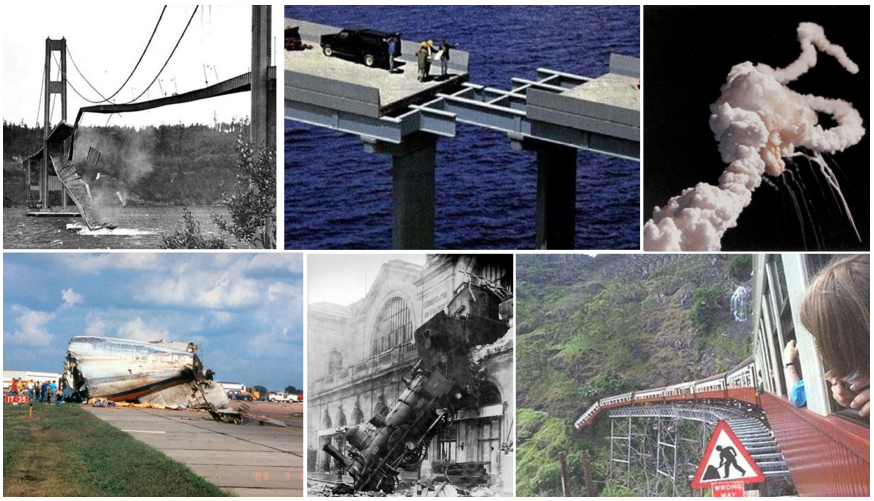


Figura 4: Erros de projeto custam vidas.

Figura 5: Resposta errada óbvia!

computadores são mecanismos de aumento de produtividade para o ocupado engenheiro e cientista. Eles permitem *trabalhar de forma mais inteligente* na solução de sistemas dinâmicos complexos. Mais tempo pode ser gasto pensando no modelamento problema real, em vez de trabalhar nos detalhes da análise numérica.

Porém, nenhum método de simplificação deve ser uma caixa preta ao engenheiro; é uma ferramenta que não se deve abusar... É uma aproximação! Cuidado no modelamento e análise e interpretação de resultados!

Para um engenheiro, não existe *um pequeno erro de projeto* - Figura 4. Se você já escutou a frase *You aren't a real engineer until you make one $50000 mistake!!!!*, por favor, esqueça-a!!!!

Na Figura 5, por exemplo, seu conhecimento de vida sabe informar o que tem de errado. Em uma análise dinâmica, a resposta errada pode não ser tão óbvia. Avalie os resultados sempre com uma certa dose de ceticismo... Não se impressione pelas cores que você aprenderá a colocar nos gráficos no Capítulo 5 (*Fancy, colorful contours can be produced by any model, good or bad!!*), e sim pela resposta que parece ser convincente.

Os resultados devem ser interpretados à luz da finalidade da análise e das possíveis implicações dos resultados no projeto. Bom senso e julgamento de um bom engenheiro projetista são muito mais importantes que os resultados do computador. Para isso, o engenheiro precisa

- de conhecimento da área (estrutural, térmica, etc.);
- de entendimento físico do problema e habilidade para resolver uma versão simplificada via métodos analíticos;
- de conhecimento das limitações e aproximações utilizadas no software;
- discutir modelo e resultados com colegas.

Para encerrar a introdução, citamos dois alertas de grandes engenheiros, pesquisadores da área de Elementos Finitos,

ENG. P. E. JOHN MCLEOD: Solving scientific and engineering problems should be fun and not a tedious chore. Modern high level computer simulation environments make this a reality. However, always remember, *Computer Simulation is a hopeful fake when the real thing is just too much...* (Founder of the International Society for Modeling and Simulation).

PROF. DR. ROBERT D. COOK: The Finite Element Method is a very useful tool which can make a good engineer great, but it can make a bad engineer dangerous (texto do Prof. Cook, University of Wisconsin, Madison, autor de vários livros em EF).

2 INTRODUÇÃO

2.1 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.

2.2 OBJETIVO

O nosso objetivo neste laboratório 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, [16].

3 INICIANDO O MATLAB

Execute o MATLAB, clicando no ícone. A tela principal do programa é mostrada na Figura 6. 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.

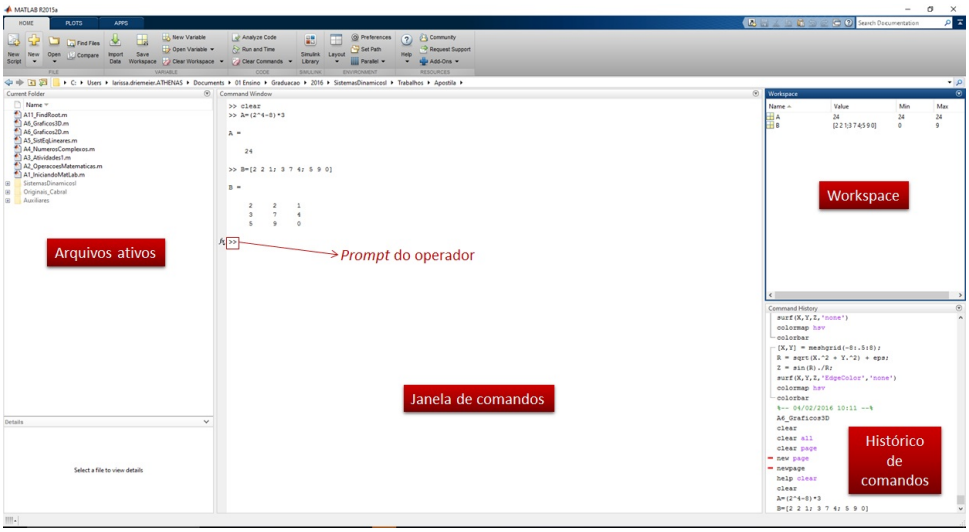


Figura 6: Tela principal do MATLAB2015a.

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 7 e ajuste o tamanho da fonte para (no mínimo) 12 pontos. Acredite: muitos erros de digitação pode ser evitados com esta simples providência!

3.1 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;

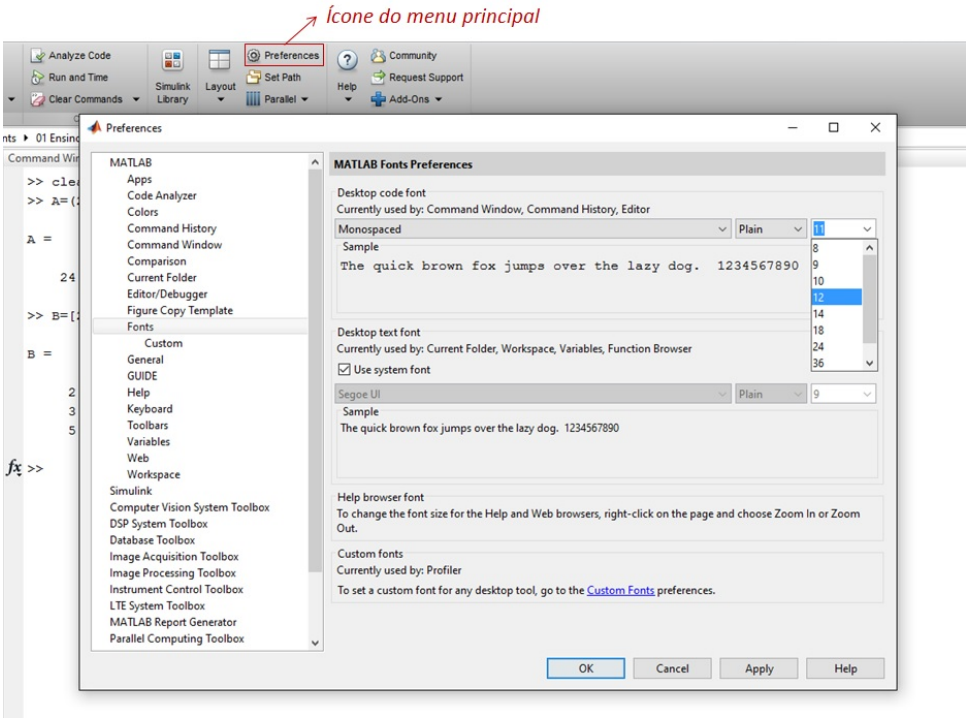


Figura 7: Ajuste da fonte usada na Janela de Comandos.

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;

3.2 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 < \ >,

```
» a=5\10
a =
      2
```

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

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 [16].

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

```
» 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`.

3.3 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
```

3.4 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 $\pi \approx 3,1416$),`i, j` (número imaginário $\sqrt{-1}$), `ans` (variável padrão para o último resultado), etc... Você pode redefinir estas variáveis, mas não convém...

3.5 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.

3.6 FUNÇÕES COMUMENTE UTILIZADAS

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

3.7 EXERCÍCIOS

1. Calcule as seguintes expressões e confira usando MATLAB,
   - $2/2*3$
   - $6-2/5+7\text{^}2-1$
   - $10/2\backslash 15-3+2*4$

Formato	Resultado	Exemplo
format short	Ponto fixo; 4 casas decimais	0.2857
format short e	Not. científica; 4 casas decimais	2.8571e-01
format long	Ponto fixo; 14 casas decimais	0.285714285714286
format long e	Not. científica; 14 casas decimais	2.857142857142857e-01
format hex	Hexadecimal	3fd2492492492492
format rat	formato racional (aprox.), i.é, razão de inteiros	2/7
format bank	valor monetário 2 casas decimais	0.29
format +	símbolos +,- e espaços em branco	+

Tabela 3: Formatos numéricos para 2/7.

- 3^2/4
- 3^2^3

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

$\sin(x)$	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	$\operatorname{sech}(x)$	secante hiperbólica de x
$\csc(x)$	cosecante de x	$\operatorname{csch}(x)$	cosecante hiperbólica de x
$\operatorname{asin}(x)$	arco cujo seno é x	$\operatorname{asinh}(x)$	arco cujo seno hiperbólico é x
$\operatorname{acos}(x)$	arco cujo cosseno é x	$\operatorname{acosh}(x)$	arco cujo coseno hiperbólico é x
$\operatorname{atan}(x)$	arco cuja tangente x	$\operatorname{atanh}(x)$	arco cuja tangente hiperbólica é x
$\operatorname{acot}(x)$	arco cuja cotangente x	$\operatorname{acoth}(x)$	arco cujo cotangente hiperbólica é x
$\operatorname{acsc}(x)$	arco cuja cosecante x	$\operatorname{acsch}(x)$	arco cujo cosecante hiperbólica é x
$\operatorname{asec}(x)$	arco cuja secante x	$\operatorname{asech}(x)$	arco cujo secante hiperbólica é x

Tabela 4: Funções trigonométricas.

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

Tabela 5: Funções elementares.

4 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 8. Note que o comando passou a fazer parte do histórico do programa e que a matriz foi armazenada na área de trabalho.

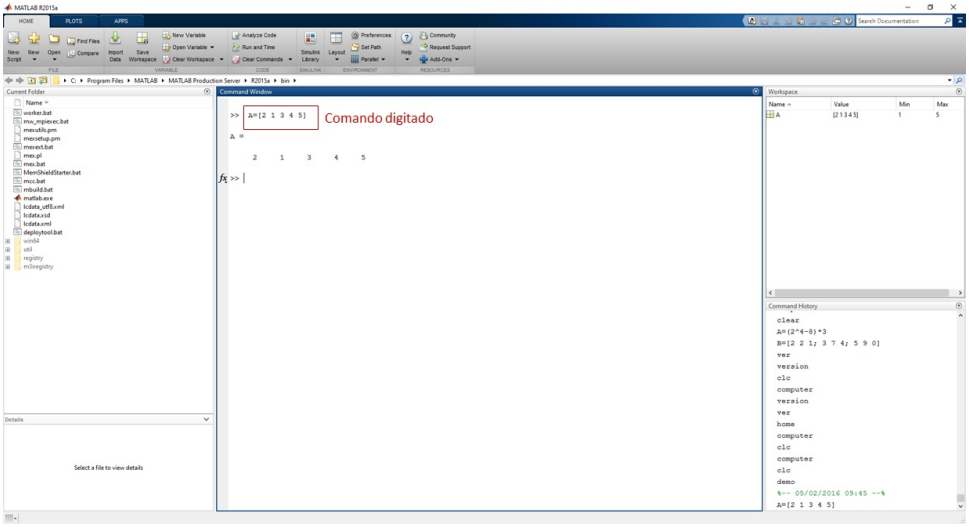


Figura 8: Atribuição de valores.

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)
v2 =
       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*.

4.1 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);
```

Conforme mencionado anteriormente, 'pi' é uma função interna do MATLAB que retorna o valor de π. 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 essa variação 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);
```

4.2 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	$[6\ 3].\backslash[4\ 8] = [4/6\ 8/3] = [0.667\ 2.667]$
.^	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 [16].

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. $C = A.\hat{}2$ (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, no item C, 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.

4.3 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 (esq para dir, ...
    sup para inf).
```

```
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. Veja um exemplo lógico em Tabela 10. 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` analisam 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,

Operadores relacionais	
<	Menor que
<=	Menor ou igual
>	Maior que
>=	Maior ou igual
==	Igual a
~ =	Diferente de
Operadores lógicos	
&	*AND*
|	*OR*
~	*NOT*
xor	*OR EXCLUSIVO*

Tabela 9: Operadores relacionais e lógicos.

Entrada		AND	OR	NOT	XOR
A	B	A&B	A|B	~A	xor(A, B)
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

Tabela 10: Exemplo de resposta de operadores lógicos.

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

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
```

4.4 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 $< \backslash >$ 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.

4.5 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 [2]):
   - $g = \begin{bmatrix} 1; & 2; & 3; & 4 \end{bmatrix}$
   - $g = \begin{bmatrix} 1 & 2 & 3 & 4; & 5 & 6 & 7 & 8 \end{bmatrix}$
   - $g - 2$
   - $2 * g - 1$
   - $g = \begin{bmatrix} 1 & 2 & 3 & 4; & 5 & 6 & 7 & 8; & 9 & 10 & 11 & 12 \end{bmatrix}$
   - $h = \begin{bmatrix} 1 & 1 & 1 & 1; & 2 & 2 & 2 & 2; & 3 & 3 & 3 & 3 \end{bmatrix}$
   - $g - h$
   - $g * h$
   - $h * g$
   - $g. * h$
   - $g./h$

- h.\g
- g * h'
- g' * h
- e = [1 2 3; 4 5 6; 7 8 0]
- f = [9 8 7; 6 5 4; 3 2 1]
- e * f
- f * e
- q = rand(3)

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 5×5 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 [16]) 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 $[30 \quad 21 \quad 19 \quad 1]$.
    - 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 4.
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.
14. Extraído de [13]. *Melancolia I* é uma famosa gravura de 1514 elaborada pelo mestre alemão renascentista e matemático amador Albrecht Dürer (Figura 9). Mostra muitos objetos matemáticos, incluindo uma esfera, um romboedro [1] truncado e, no canto superior direito, um quadrado mágico de ordem 4.

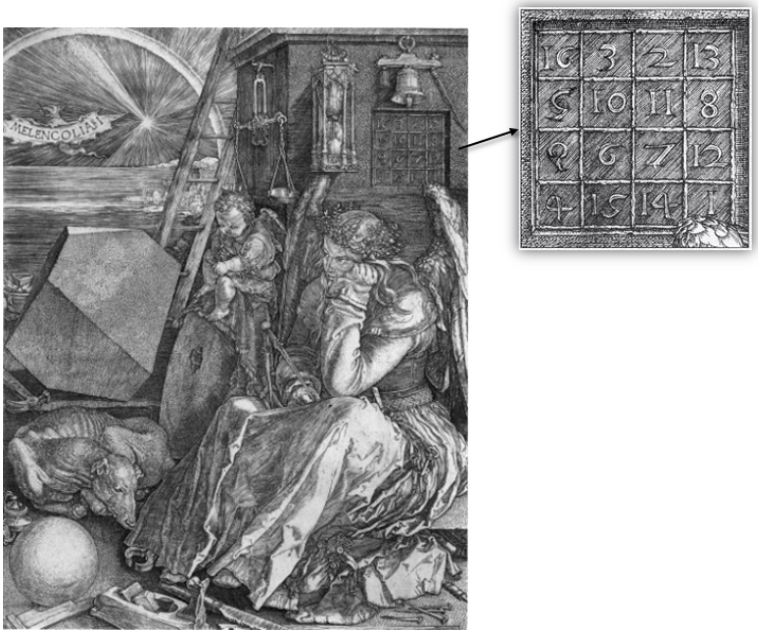


Figura 9: Melancolia I.

1 Um romboedro é um poliedro de seis faces (hexaedro), todas elas losangos idênticos.

Porém, os quadrados mágicos são ainda anteriores à história registrada. Uma antiga lenda chinesa conta que uma tartaruga emergiu do rio Lo durante uma inundação. O casco da tartaruga mostrou um padrão muito incomum - uma grade três por três contendo vários números de manchas (Figura 10). Cada uma das três linhas, as três colunas e as duas diagonais contêm um total de 15 pontos. As referências a Lo Shu e ao padrão numérico Lo Shu ocorrem ao longo da história chinesa. Hoje, é a base matemática para o Feng Shui.

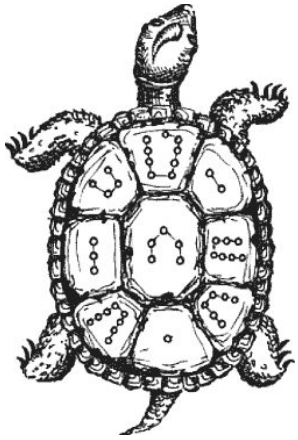


Figura 10: Lo Shu. Extraído do site http://www.formation-fengshui-romandie.ch/historique-loshu.html.

O comando `A = magic(3)` define a matriz,

```
A =  8  1  6
     3  5  7
     4  9  2
```

O comando `sum(A)` mostra que a soma de cada coluna vale 15; e o comando `sum(diag(A))` soma a diagonal principal e mostra que o valor também é 15. A diagonal oposta, menos importante na Algebra, pode ser somada com o seguinte comando: `sum(diag(flipud(A)))`. Veja que a soma de linhas e colunas também dá 15, `sum(A(1,:))`.

O comando `A = magic(4)` define a matriz,

```
A =  16   2   3  13
      5  11  10   8
      9   7   6  12
      4  14  15   1
```

Novamente, os comandos

`sum(A), sum(A'), sum(diag(A)), sum(diag(flipud(A)))`

mostram que esse é, de fato, um quadrado mágico, porém não é o mesmo de Dürrier. Para tal, devemos trocar a segunda e terceira coluna,

`A=A(:,[1 3 2 4])`.

Para explorar mais os quadrados mágicos, recomenda-se [13].

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

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

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

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

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

16. Considere a Figura 11 representando um sistema de 4 molas ligadas em série sujeito a uma força F de 270N. Determine as relações de equilíbrio, e os deslocamentos x_i no MATLAB, dadas as constantes das molas (em N/m): $k_1 = 150$, $k_2 = 50$, $k_3 = 75$ e $k_4 = 225$.

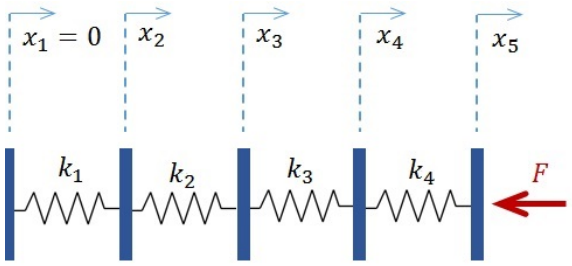


Figura 11: Molas em série.

17. 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	1	1
2	0	1	2
3	2	1	1

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

Supondo que cada caminhão vai com sua carga máxima, quantos caminhões de cada tipo devemos enviar para transportar exatamente 27 máquinas A, 23 máquinas B e 31 máquinas C?

Parte II

AULA 02 - 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.

5 GRÁFICOS

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

5.1 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 *sinc*:

$$y = \sin(x)/x$$

no intervalo $x \in [-15, 15]$, pode-se utilizar a seguinte sequência de comandos:

```
x = -15:0.1:15;
y = sin(x)./x;
plot(x,y)
```

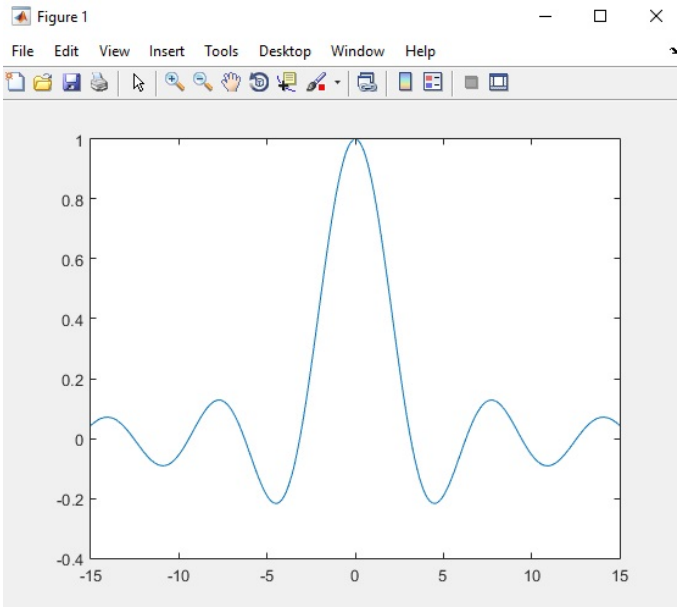


Figura 12: Exemplo de resultado gráfico da função `plot`.

O resultado, apresentado na Figura 12, é exibido em uma janela de figura identificada por um número (`Figure 1`). O mesmo resultado é obtido com a função `fplot`, basta identificar a string a ser representada no domínio do gráfico,

```
y='sin(x)./x';
fplot(y,[-15 15])
```

Com a função `plot` você precisa definir manualmente os valores e calcular a função correspondente e dada. Com `plot` você define a função de forma genérica, por exemplo como uma função anônima (*Anonymous Functions*)[1],

```
xt = @(t) cos(3.*t);   % Funcao anonima
yt = @(t) sin(2.*t);   % Funcao anonima
fplot(xt,yt)
```

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`. Teste os seguintes comandos:

```
figure;
f = @(x) sin(1/x);   % Funcao anonima
fplot(f, [-1 1]);
figure(2);
semilogx(x);
```

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 13.

```
figure(1);
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
title('Graficos sobrepostos')   % Atribui um titulo ao grafico
legend('y','z') % legenda
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 11 mostra as configurações de cores, marcadores e linhas, opções essas válidas para plotar gráficos no MATLAB, em 2D e 3D. Todos os resultados gráficos aparecem na janela de figura ativa. Como você já deve ter percebido, 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 e a torna ativa.

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

```
plot(x,y,'LineWidth',1);    % Desenha o grafico de uma funcao
hold on     % Ativa a 'trava' de exibicao grafica
plot(x,z,'LineWidth',2);    % Desenha outro grafico na mesma janela ...
    em outra espessura
```

1 As funções anônimas permitem que você defina uma função sem criar um programa. Uma aplicação comum de funções anônimas é definir uma expressão matemática.

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

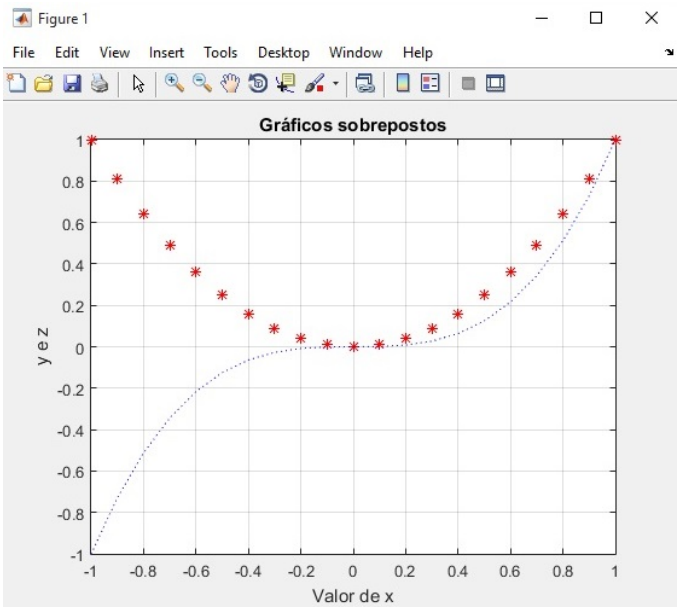


Figura 13: Gráfico com duas funções superpostas.

```
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 14.

```
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
```

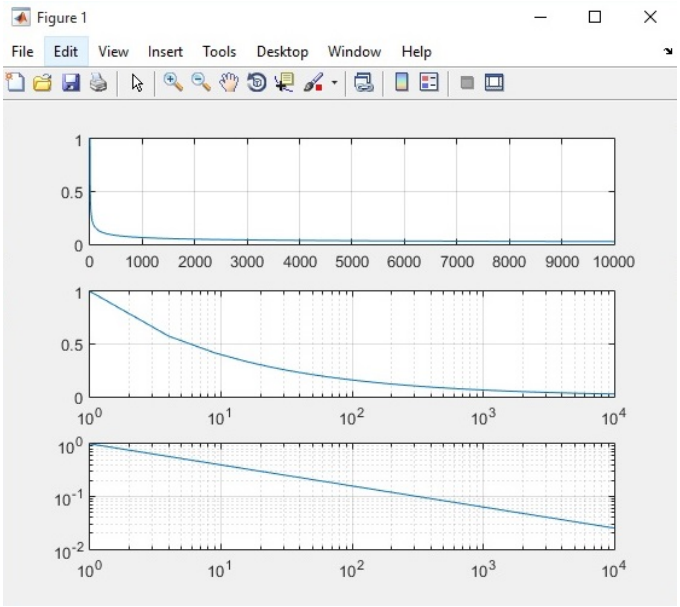


Figura 14: Função `subplot`.

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))`. Números complexos serão tratados mais adiante.

5.2 GRÁFICOS TRIDIMENSIONAIS

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 o gráfico de uma hélice tridimensional. Note o uso da função `zlabel` para nomear o eixo z do gráfico. O resultado está representado na Figura 15.

```
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;
```

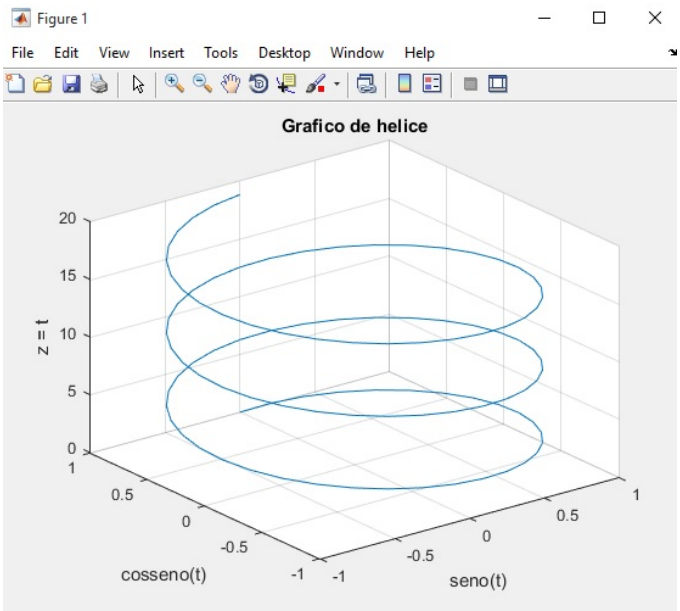


Figura 15: Gráfico de linha tridimensional.

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 variáveis 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',
y = linspace(3,5,2) % Geracao de valores para 'y'
[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 16):

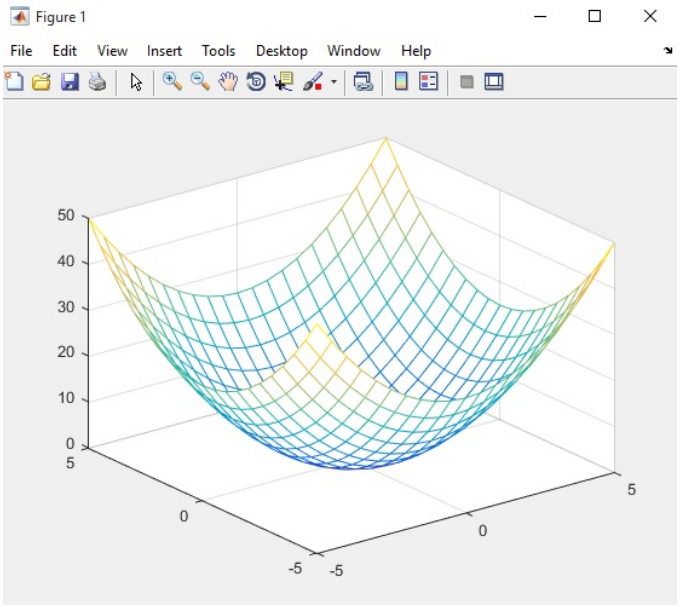


Figura 16: Parabolóide elíptico gerado com a função `mesh`.

```
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 (veja Figura 17). Outros mapas de cores estão listados na Tabela 12.

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

A Figura 18 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])
```

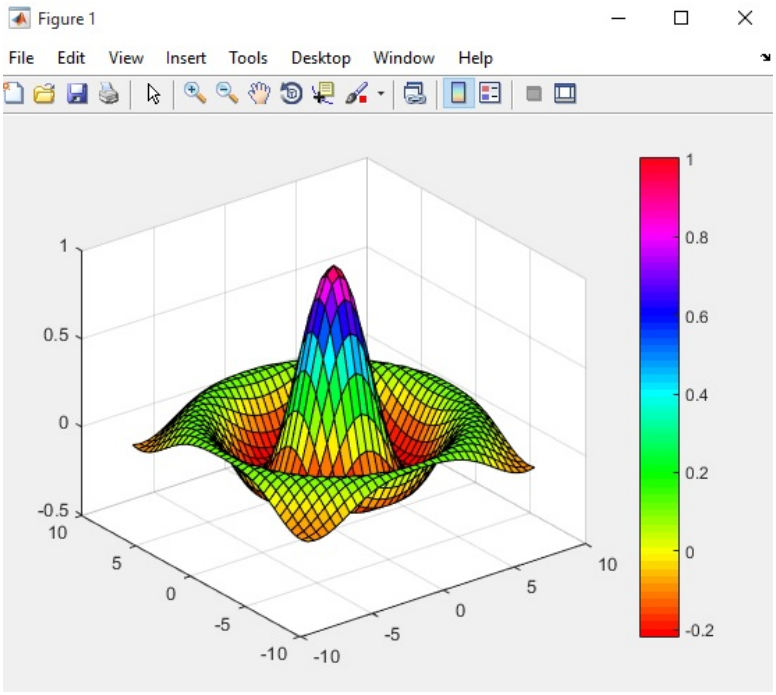
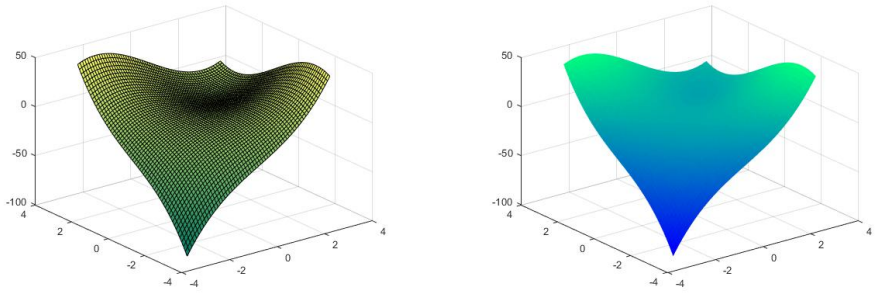


Figura 17: Superfície gerada pela função `surf`.

Função	Mapa de cores
hsv	Escala com cores saturadas
hot	Preto-vermelho-amarelo-branco
gray	Escala 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 12: Mapas de cores utilizados pelo MATLAB.



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

Figura 18: Exemplo de uso do mapa de cores.

5.3 EXERCÍCIOS

1. Plote a função 2D $\sin(\pi/x)$, para $x = -1..1$ (atenção ao intervalo adotado entre os pontos a serem plotados...).

2. Veja o exemplo abaixo:

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

A Tabela 13 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 13: 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 item 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 19 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 19b.

   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 geometria, 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, ρ 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 e C_D contra o ângulo de ataque α.

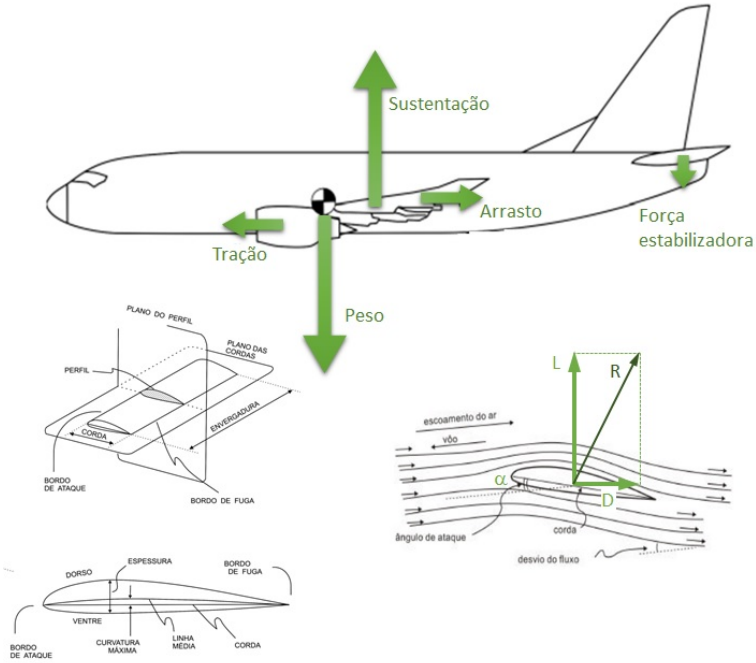


Figura 19: Forças atuantes em um avião e detalhamento do aerofólio.

[Extraído de Palm III [16]] 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 α 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 30m^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 α para $-2^o \leqslant \alpha \leqslant 22^o$. Determine o ângulo de ataque que maximiza a eficiência.

  As seguintes linhas de código ajudam a encontrar o valor máximo (pesquise na Internet, caso o código não fique claro para você),

```
maxF = max(F);
indexOfFirstMax = find(F == maxF, 1, 'first');
maxY = F(indexOfFirstMax)
maxX = alpha(indexOfFirstMax)
```

5. Dado o amortecedor com êmbulo mostrado na Figura 20, a seguinte expressão para a constante c de amortecimento pode ser deduzida [18],

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

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 μ; d é a folga entre o pistão e a parede do cilindro.

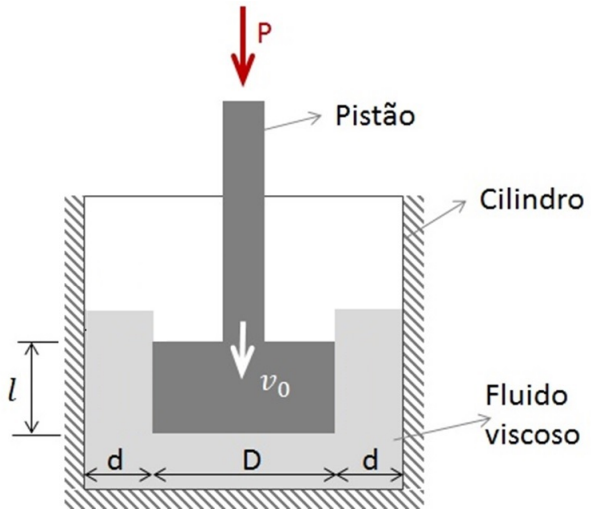


Figura 20: Amortecedor de êmbulo. Figura adaptada de Rao [18].

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

6 PROGRAMAÇÃO

Um dos aspectos mais importantes para nosso curso 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 14.

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 14: 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**: HOME → New → Script ou HOME → New → Function. Neste arquivo aberto em uma nova janela que deve ser escrito o programa. Uma vez escrito o programa a ser executado, deve-se salvar o arquivo: EDITOR → Save e rodar EDITOR → Run ou F5.

6.1 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;
% g = gravity (m/s^2)
% m = mass (kg)
```

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.

```
% t = time (s)
% cd = second-order drag coefficient (kg/m)
v = sqrt(g * m / cd) * tanh(sqrt(g * cd / m) * t)
```

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
```

6.2 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'.
```

6.3 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].

6.3.1 *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
```

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

As expressões podem incluir operadores relacionais e lógicos, como mostrado na Tabela 9, definida no Capítulo 4. Exemplo:

```
if x == 0
   y = sin(3*t+a);
else
   y = cos(3*t-a);
end
```

6.3.2 *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;
i = 1;
while i < = length(V)
  S = S+V(i);
  i = i+1;
end   % Neste ponto, S = sum(V)
```

6.3.3 *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);
for i = 2:length(V)    % O incremento foi omitido!
  if V(i) > M
    M = V(i);
  end
end   % Neste ponto, M = max(V)
```

6.3.4 *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
    case valor2
        grupo de sentencas 2
  ...
    otherwise
        grupo de sentencas n
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
```

6.4 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.

6.5 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,

```
» a=1:3;
» v=input('Enter Matlab vector expression for a: ')
Enter Matlab vector expression for a: a.^2
v =
     1   4   9
```

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
```

6.6 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*,

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.

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

The velocity is 50.62 m/s
```

6.7 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.

6.8 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 ax^2+bx+c=0
% variavel v=[a b c];

% 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);
```

6.9 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 que transforme a temperatura de Celsius para Fahrenheit e, utilizando a função `switch`, defina a temperatura como *high, moderate, low* ou *very low*.

3. 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 a 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 o seu script crie uma matriz com dimensão nxn pequena e calcule a sua exponencial com diferentes números de termos na série (diferentes valores de N). Utilize, por exemplo, N variando de 3 a 100. 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. Verifique o `script` listado abaixo, que calcula e traça a posição, velocidade e aceleração do pistão, para duas revoluções do mecanismo biela-manivela da Figura 21. Assume-se condições iniciais nulas.

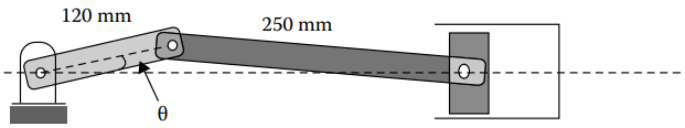


Figura 21: Extraído de [3].

```
% Matlab codes to simulate the Piston-Crank mechanism
```

```
% Font: Modeling and Simulation od systems using MATLAB and ...
    Simulink, pg. 96
% author: D. K. Chaturvedi
clear all
theta_dot=500; % rpm
r=0.12; c=0.25; % m
t_rev=2*pi/ theta_dot % Time for one revolution
t=linspace(0,t_rev, 200); % time vector
theta=theta_dot*t ; % calculate theta at each time.
h=r*sin(theta);
d2s=c^2-r^2*sin(theta).^2; %calculate d2 square
d2=sqrt(d2s);
x=r*cos(theta)+d2; % calculate x for each theta
x_dot=-r*theta_dot*sin(theta)-(r^2*theta_dot*sin(2*theta)./(2*d2));
x_dot_dot=-r*theta_dot^2*cos(theta)-(4*r^2*theta_dot^2* ...
cos(2*theta).*d2s+(r^2*sin(2*theta)*theta_dot).^2)./(4*d2s.^(3/2));
subplot(2,2,1)
plot(t,x) % plot Position vs t
xlabel('Time(s)')
ylabel('Position(m)')
subplot(2,2,2)
plot(t,x_dot) % plot Velocity vs t
xlabel('Time(s)')
ylabel('Velocity(m/s)')
subplot(2,2,3)
plot(t,x_dot_dot) % plot Acceleration vs t
xlabel('Time(s)')
ylabel('Acceleration(m/s^2)')
subplot(2,2,4)
plot(t,h) % plot h vs t
xlabel('Time(s)')
ylabel('h(m)')
```

5. Crie uma `function` que calcule a constante elástica equivalente de um tambor de içamento equipado com cabo de aço e montado conforme a Figura 22.

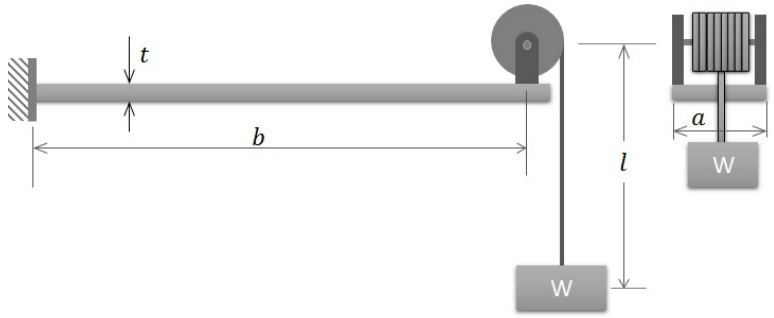


Figura 22: Tambor de içamento.

6. Crie um `script` que plote a constante elástica torcional equivalente de um eixo de um propulsor a hélice de aço (E = 200 GPa e G = 80 GPa), em função da espessura $50\text{mm} < t < 220\text{mm}$ das paredes (constante para as duas seções), conforme Figura 23.

7. 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 nível 3º grau é tirado da *classificação de contatos imediatos com alienígenas* criada pelo ufologista J. Allen Hynek - veja Figura 24.

   A comunicação entre os humanos e a raça alienígena era feita através de uma sequência de tons que os cientistas acreditavam ser reconhecida pelos

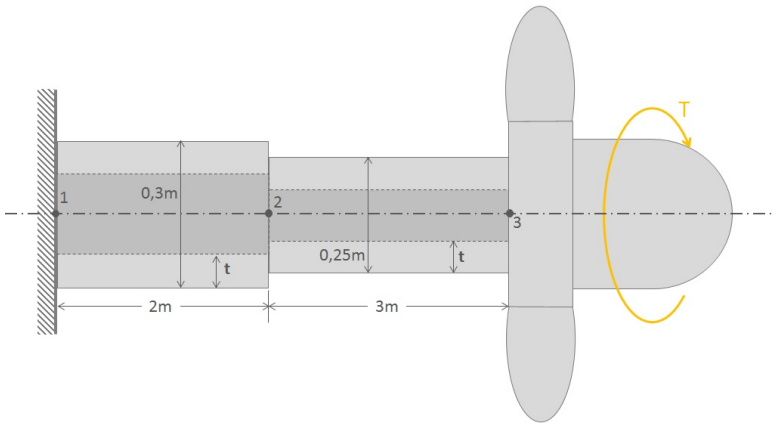


Figura 23: Figura adaptada de Rao [18].

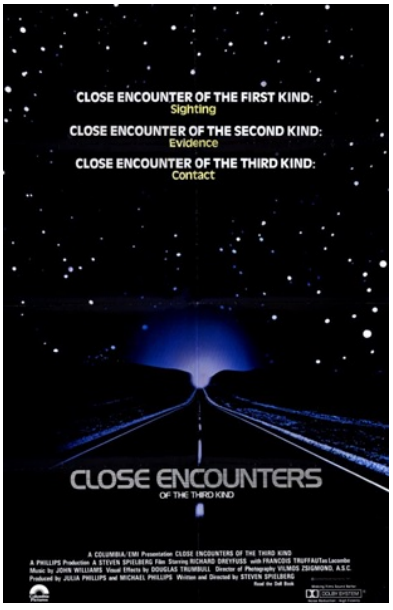


Figura 24: Close Encounters

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

AULA 03 - EQUAÇÕES DIFERENCIAIS

A simulação de um sistema dinâmico consiste na solução de equações diferenciais para condições iniciais de contorno. Em diversos problemas práticos de mecânica, as equações matemáticas que descrevem o seu comportamento são equações diferenciais ordinárias não lineares.

7

SISTEMAS LINEARES - REPRESENTAÇÃO NO ESPAÇO DE ESTADOS

O problema modelado, deve ser agora analisado. Por exemplo, a suspensão de nosso automóvel, como se comportará quando o motorista, sem querer, subir no meio fio? Isto é, qual a resposta de um sistema a uma determinada entrada? Nesse caso, a entrada seria uma função degrau e o sistema a suspensão, que modelamos no Capítulo 1. E a saída? O movimento da massa que representa 1/4 do veículo?

Para analisarmos de forma correta, temos que organizar as ideias. O que é entrada do sistema? Qual a saída que me interessa?

Neste capítulo, teremos uma recordação da representação das equações diferenciais de nosso sistema em espaço de estados. Para modelos simples, obter sua resposta para uma função impulso, degrau, rampa... é simples com ajuda de um software comercial (obviamente, resolver equação diferencial na raça, não é nunca uma tarefa fácil!).

7.1 SLIT - SISTEMAS LINEARES INVARIANTES NO TEMPO

SISTEMA LINEAR a resposta de um sistema linear a uma soma ponderada de sinais de entrada é igual à mesma soma ponderada dos sinais de saída associados a cada um dos respectivos sinais de entrada. Em outras palavras, sistemas são lineares se satisfazem duas propriedades: *homogeneidade* e *aditividade*:

HOMOGENEIDADE quando o sinal de entrada $x(t)$ é multiplicado por um valor k, então o sinal de saída $y(t)$ fica também multiplicado por este mesmo valor k;

ADITIVIDADE quando o sinal de entrada é a soma de dois sinais $x_1(t)$ e $x_2(t)$, que produzem individualmente sinais de saída $y_1(t)$ e $y_2(t)$ respectivamente; então o sinal de saída é a soma dos sinais de saída $y_1(t)$ e $y_2(t)$.

INVARIANTE NO TEMPO é aquele sistema que para uma entrada $x(t)$ o sinal de saída é $y(t)$, independente de quando é aplicada esta entrada. Ou seja, as condições dinâmicas do sistema não mudam com o passar do tempo. Obviamente, nenhum sistema é, na realidade, invariante no tempo, mas pode-se considerar assim muitos sistemas cuja variação é muito lenta.

Importante relembrar (vide nosso primeiro capítulo) que, em Sistemas Dinâmicos, o grande passo é aprender a *modelar*. Algumas hipóteses devem ser consideradas para modelar um problema físico. Por exemplo, nenhum corpo real é rígido ao ser acelerado e o elemento de inércia é um modelo, não um objeto físico. Exemplos de hipóteses simplificadoras estão na Figura 25, extraída de [7].

7.2 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 $t \geqslant t_0$.

1 – Todas as massas são rígidas e têm valores constantes.
2 – As molas são puras e lineares. Não têm perda de energia e não possuem massa. Os seus respectivos coeficientes são constantes.
3 – Os amortecedores são puros e lineares. Não têm efeito de mola e não possuem massa. Os seus respectivos coeficientes são constantes.
4 – Quando a massa pode transladar, o seu movimento é em uma só direção.
5 – Quando a massa pode girar, a sua rotação é em torno de um só eixo.
6 – Todas as condições iniciais são nulas.
7 – Superfícies em contato têm atrito nulo (planos, mancais, etc.).
8 – As variações das grandezas do sistema são relativamente pequenas, a fim de manter o comportamento do sistema praticamente linear.

Figura 25: Hipóteses Gerais de Sistemas Mecânicos. Tabela extraída de [7].

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:

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \qquad \text{equação dos estados} \\ y(t) &= C(t)x(t) + D(t)u(t) \qquad \text{equação de saída} \end{aligned} \tag{6}$$

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 está presente nas equações dinâmicas**. Quando o modelo matemático é obtido usando as leis da física, 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 6 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 detalhes, estude o capítulo 2 do livro texto da disciplina [15]. Existe ainda a representação pela função de transferência, usada em Controle Clássico, assunto a ser tratado futuramente.

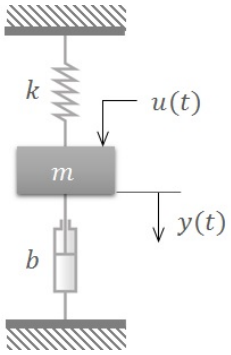


Figura 26: Sistema mecânico massa-mola-amortecedor. Figura adaptada de Ogata [15].

Como exemplo considere o sistema massa-mola-amortecedor da Figura 26, cujo modelo é representado pela seguinte equação diferencial de 2^a ordem:

$$m\ddot{y}(t) + b\dot{y}(t) + ky(t) = u(t) \tag{7}$$

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:

$$\begin{aligned} x_1(t) &= y(t) \\ x_2(t) &= \dot{y}(t) \end{aligned} \tag{8}$$

e a entrada,

$$u_1(t) = u(t) \tag{9}$$

No caso, a entrada é um escalar e não um vetor, ($m = 1$). Colocando na forma de espaço dos estados (Equação 6), por substituição na Equação 7, tem-se

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}(-kx_1 - bx_2) + \frac{1}{m}u_1(t) \end{aligned}$$

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,

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-k}{m} & \frac{-b}{m} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) \tag{10}$$

A equação de saída é,

$$y(t) = x_1(t) \tag{11}$$

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 10 e Equação 11 com Equação 6, 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:

```
» 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.
```

7.3 SIMULAÇÃO DE SISTEMAS DINÂMICOS LINEARES

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

7.3.1 *Função impulso*

A função impulso unitário é definida como,

$$\delta(t) = \begin{cases} 0, & t \neq 0 \\ 1, & t = 0 \end{cases} \tag{12}$$

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);
```

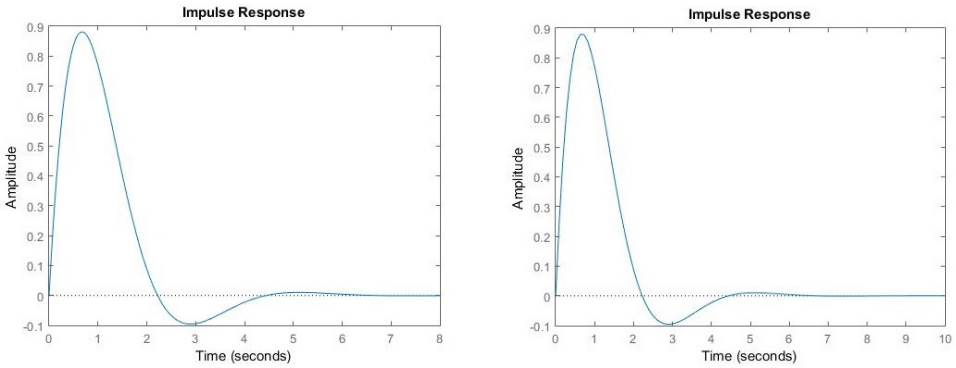
O resultado da simulação é apresentado em uma janela gráfica, como mostra a Figura 27a. Opcionalmente, pode-se fornecer um valor em segundos para o tempo final de simulação. Por exemplo, para simulação de 10s deve-se digitar o comando (Figura 27b),

```
» 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. Verifique a resposta à função impulso do sistema dado como exemplo, modificando variáveis como rigidez ou amortecimento.



(a) *Análise automática.* (b) *Análise 10s.*

Figura 27: Função impulso.

7.3.2 *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 28.

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:

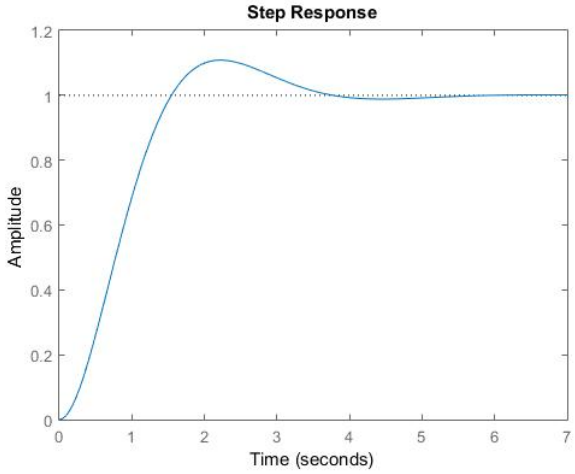


Figura 28: Função step do MATLAB

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

7.3.3 *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 constantes = 0.5
lsim(sys,u,t); % Simulacao
```

O resultado da simulação é apresentado em uma janela gráfica, como mostra a Figura 29.

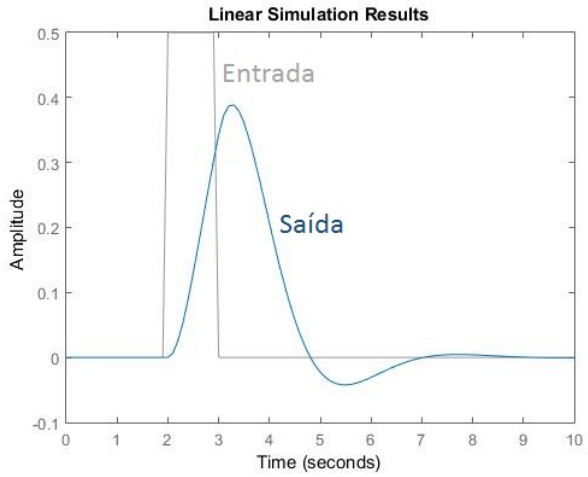


Figura 29: Resposta a um sinal genérico.

A resposta para uma função rampa é de grande importância para o curso. O MATLAB não possui um comando direto, mas pode-se usar a função `lsim`. Ainda para o sistema `sys` definido anteriormente, a resposta para uma função rampa de entrada é vista Figura 30.

```
t = 0:0.1:10; % Vetor de tempo de simulacao
u = zeros(length(t),1); % Vetor de entrada, com mesma dimensao de t
for i=21:length(t) % Atribuicao de valores nao nulos linearmente ...
    crescentes
 u(i) = t(i)-2; % Atribuicao de valores nao nulos
end
lsim(sys,u,t); % Simulacao
```

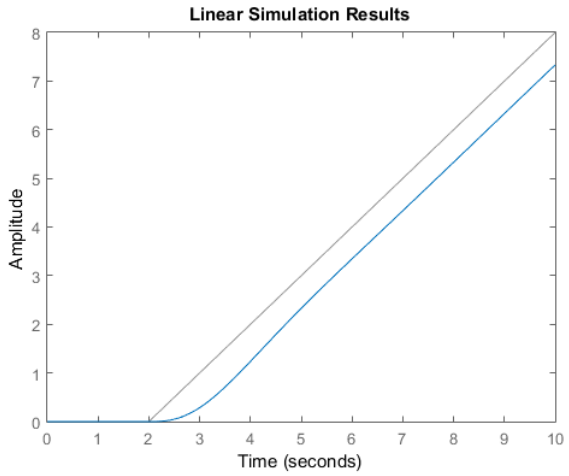


Figura 30: Entrada genérica tipo rampa.

Importante ressaltar que a análise da resposta obtida em comparação à entrada imposta será discutida mais adiante, no estudo de sistemas de primeira e segunda ordem. Aqui nos contentamos a simular sem interpretar a resposta.

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

» `lsim(b,a,u,t);`

quando a equação diferencial tem a forma geral,

$$\frac{d^n}{dt^n}y(t)+a_{n-1}\frac{d^{n-1}}{dt^{n-1}}y(t)+\ldots+a_1\frac{d}{dt}y(t)+a_0y(t)=b_m\frac{d^m}{dt^m}u(t)+\ldots+b_1\frac{d}{dt}u(t)+b_0u(t) \tag{13}$$

onde

b = $[b_m, b_{m-1}, b_{m-2}, \ldots, b_1, b_0]$ é o vetor de coeficientes especificados no lado direito da Equação 13;

a = $[1, a_{n-1}, a_{n-2}, \ldots, a_1, a_0]$ é o vetor de coeficientes do lado esquerdo da Equação 13;

u = é o vetor de instantes conhecidos do sinal $u(t)$ especificados na Equação 13;

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 13.

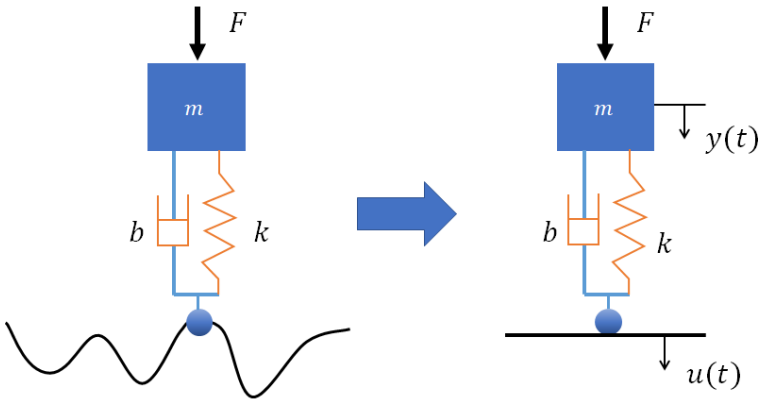


Figura 31: Sistema massa-mola-amortecedor.

Por exemplo, pode-se comprovar que a relação entre a altura $u(t)$ de uma via e o deslocamento $y(t)$ do carro contendo um sistema de absorção de energia massa-mola entre as rodas e o chassis é (Figura 31),

$$m\ddot{y}(t) + \frac{b}{m}\dot{y}(t) + \frac{k}{m}y(t) = g + \frac{b}{m}\dot{u}(t) + \frac{k}{m}u(t) \tag{14}$$

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

Por fim, pode-se converter a equação no formato da Equação 14 em equações no espaço de estados, conforme Equação 6, através do comando `tf2ss`, do inglês *transfer function to state-space*:

« `[A,B,C,D]=tf2ss(b,a)`

A listagem a seguir exemplifica o caso ilustrado na Equação 14 quando a perturbação $u(t)$ na pista é um degrau. Valores de massa, rigidez e amortecimento são, respectivamente, $m = 1000$; $k = 2000$ e $b = 500$, definidos em unidades coerentes. A título de ilustração, o amortecimento é duplicado e quadruplicado.

```
% Resposta do veiculo a obstaculos na pista
%%Parametros
clear all; clc; %close all;
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:
%  d^n/dt^n {y} + a_(n-1) d^n-1/dt^n-1 y + a_1 d/dt {y} + a_0 {y} =
%  b_m d^m/dt^m {u} + b_(m-1) d^m-1/dt^m-1 u + b_1 d/dt {u} + b_0 {u}
% onde:
% b=[b_m,b_{m-1},...,b_1,b_0] vetor de coeficientes (lado direito);
% a=[1, a_{n-1},...,a_1,a_0] vetor de coeficientes (lado esquerdo);
% u=  vetor de instantes conhecidos do sinal u(t);
% t= vetor tempo 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.
%
%O comando MatLab   : y=lsim(b,a,u,t)
%
%  d^2/dt^2 {y} + b/m d/dt {y} + k/m {y} = b/m d/dt {u} + k/m {u}
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
```

Obviamente, essa é somente a introdução do assunto, que será tratado com muito mais profundidade no decorrer do seu curso de Mecatrônica.

7.4 EXERCÍCIOS

1. Adaptado de [20]: Para os sistemas abaixo determine se o sistema é linear ou não linear e invariante ou não invariante no tempo.
   - $y(t) = u(t+1)$;
   - $y(t) = 1/u(t)$;
   - $3\ddot{y}(t) + \dot{y}(t) - y(t) = u(t)$;
   - $y(t) = u(t)\sin t$;
   - $y(t) = u(t) + 2$;
   - $y(t) = u(t)\cos 2\pi t$
2. 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. Faça manualmente as matrizes e compare com o resultado do MatLab (usando a função tf2ss).

     Importante ressaltar aqui que sua resposta pode ser diferente daquela dada pelo MatLab. O MatLab usa um modelo chamado *Controllable Canonical Form*,

enquanto que o método utilizado em nossa análise é conhecido como *Observable Canonical Form*. Mas a função de transferência (i.é, relação entre saída e entrada do sistema, que você aprenderá mais adiante) será sempre a mesma. Tudo isso é para tranquilizá-lo, sua resposta pode estar certa e para mostrar que falta ainda muito conhecimento...

- 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.

3. O sistema mostrado na Figura 32 possui as seguintes características $k_1 = k_2 = 5N/m$, $a = 0,25m$, $b = 0,75m$, $L = 1m$, $M = 2kg$, $m = 1kg$.
   - Derive sua equação de movimento para a rotação em torno de O;
   - Coloque o sistema na forma do espaço dos estados e defina as matrizes A, B, C e D do sistema;
   - Mostre a resposta do sistema para a função impulso $F(t) = 500\delta(t)N$
   - Mostre a resposta do sistema para a função $F(t) = 500\sin(\omega t)N$, com $\omega = 1, 10$ e $100rpm$. Discuta os resultados.

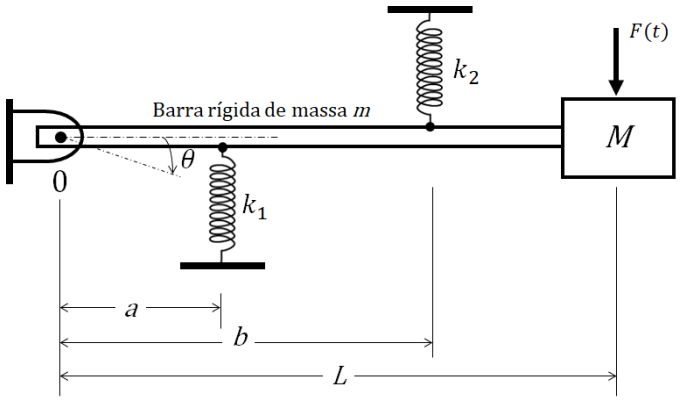


Figura 32: Barra uniforme, rígida, apoiada em duas molas.

4. A Figura 33 mostra um modelo mais completo de 1/4 de veículo, onde m_1 é 1/4 da massa estrutural do veículo, m_2 é a massa da montagem roda-pneu-eixo; k_1 e b_1 são, respectivamente, a rigidez e amortecimento da suspensão, k_2 representa a rigidez dos pneus; $z(t)$ é o deslocamento da superfície da rodovia; o atuador de força, representado por f é controlado por feedback e representa os componentes ativos do sistema de suspensão. Suponha que um automóvel em movimento passe por diferentes obstáculos (elevações) na pista.
   - Desenhe os diagramas de corpo livre e derive as equações diferenciais de movimento;
   - Determine a representação em espaço de estados;
   - Usando MATLAB, defina o sistema na forma de espaço de estados;
   - Analise a resposta do modelo para um degrau na pista, conforme Figura 34. Despreze a parte ativa da suspensão. Dados: $m_1 = 350Kg$,

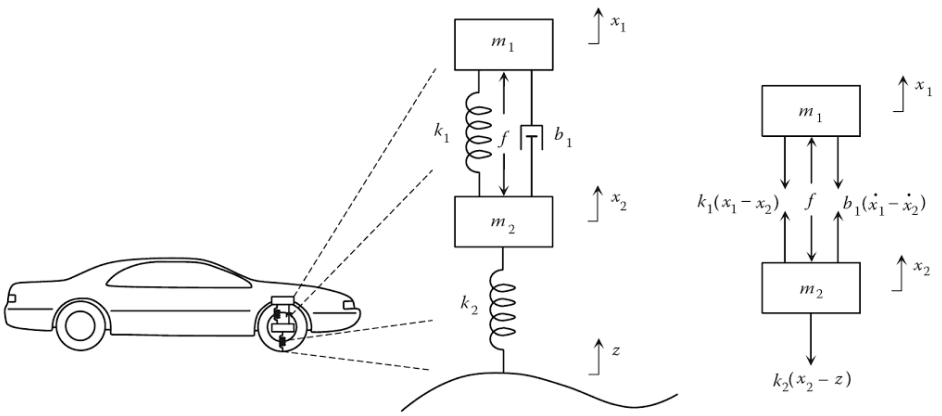


Figura 33: Um quarto de veículo. Figura extraída de [6].

$m_2 = 70Kg$, $b_1 = 2500Ns/m$, $k_1 = 176000N/m$ e $k_2 = 27000N/m$. Duplique e quadruplique o amortecimento, e compare as respostas.

Considere a saída como a amplitude de deslocamento da massa m_1. Caso queira verificar a amplitude de deslocamento de ambas as massas, estude mais sobre o comando:

```
opts = stepDataOptions('InputOffset',[0 0],'StepAmplitude',[0 0.25]).
```

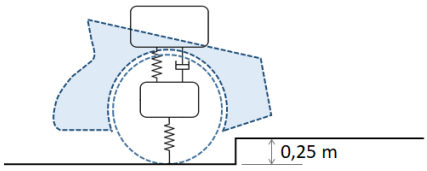


Figura 34: Entrada da pista.

8 SOLUÇÃO DE EQUAÇÕES DIFERENCIAIS

Como já discutido, 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 lineares e 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 utilize a função `ode15s`.

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

8.1 EDO DE PRIMEIRA ORDEM

A função `ode45` foi projetada para lidar com o seguinte problema geral:

$$\dot{y} = f(t, y) \qquad y(t_0) = y_0 \tag{15}$$

onde t é a variável independente, y é um vetor de variáveis dependentes a serem encontradas e $f(t, y)$ é uma função de t e y. O problema matemático é especificado quando o vetor de funções do lado direito da Equação 15, $f(t, y)$, é definido e as condições iniciais, $y = y_0$ no instante t_0, são fornecidas.

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 15:

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

Inicialmente, cria-se a função `ydot`,

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

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]`

A condição inicial dada é que $y = -2$ para $t_0 = 1$ e queremos integrar $1 \leqslant t \leqslant 4$. O seguinte conjunto de comandos mostra explicitamente a solução,

```
tspan = [1 4];                          %vetor intervalo de integracao
y0 = -2;                                %condicao inicial
[tout,yout] = ode45(@ydot,tspan,y0); %resolve o problema
plot(tout,yout)
```

8.2 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 (i.é, reduzir a ODE dada para uma série de equações de primeira ordem). 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,

$$x(1) = y$$
$$x(2) = \dot{y};$$

e

$$\dot{x}(1) = x(2)$$
$$\dot{x}(2) = \ddot{y} = \frac{1}{5}u(t) - \frac{4}{5}x(1) - \frac{7}{5}x(2)$$

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,

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

E, para utilizar a função `ode45`,

```
[t,x]=ode45(@estado_1,[0,6],[0,9])
plot(t,x(:,1))}
```

A matriz `x` tem o comprimento da variável `t`. Cada coluna de `x` é uma variável dependente diferente. Por exemplo, em nosso caso, a primeira coluna representa y a cada instante e a segunda coluna, $\dot{y}$. Para plotar as duas funções $x(1)$ e $x(2)$ versus `t`, utilize a função `plot(t,x)`; no exemplo, será plotado apenas $x(1) = y$ usando o comando `plot(t,x(:,1))`.

8.3 MODELO DE UM PÊNDULO NÃO LINEAR

Esta seção é um resumo do exemplo apresentado em Palm III [16], capítulo 9, páginas 389-392. O exemplo refere-se a um pêndulo de massa m concentrada na extremidade de uma haste de massa desprezível, como mostrado na Figura 35. A equação de movimento do sistema é,

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

Suponha $L = 1m$ e $g = 9,81 m/s^2$. 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 $\dot{\theta}(0) = 0$.

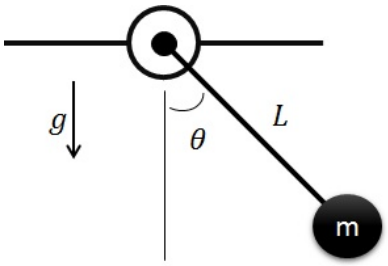


Figura 35: Pêndulo.

8.3.1 *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 $T = 2\pi\sqrt{L/g} = 2,006s$

8.3.2 *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 36.

8.4 SISTEMA DE VÁRIAS EQUAÇÕES NÃO LINEARES ACOPLADAS

Para o seguinte sistema acoplado de equações diferenciais,

$$\begin{aligned} \ddot{y} &= \dot{z}^2 + \tan z \\ \ddot{z} &= \dot{y} + \dot{z} + \cos z \end{aligned} \tag{16}$$

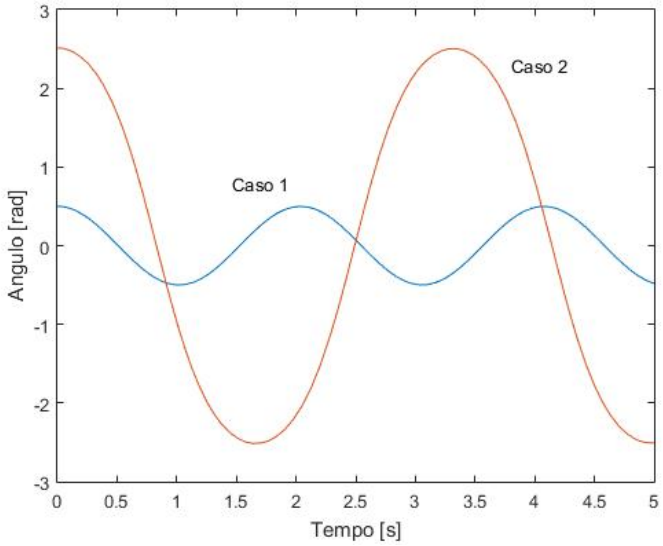


Figura 36: Solução do pêndulo para duas condições iniciais.

define-se as seguintes variáveis,

$$\begin{aligned} x(1) &= y \\ x(2) &= \dot{y} \\ x(3) &= z \\ x(4) &= \dot{z} \end{aligned} \tag{17}$$

de modo que

$$\begin{aligned} \dot{x}(1) &= x(2) \\ \dot{x}(2) &= x(4)^2 + \tan x(3) \\ \dot{x}(3) &= x(4) \\ \dot{x}(4) &= x(2) + x(4) + \cos x(3) \end{aligned} \tag{18}$$

A solução, via MATLAB, é a seguinte (Figura 37),

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

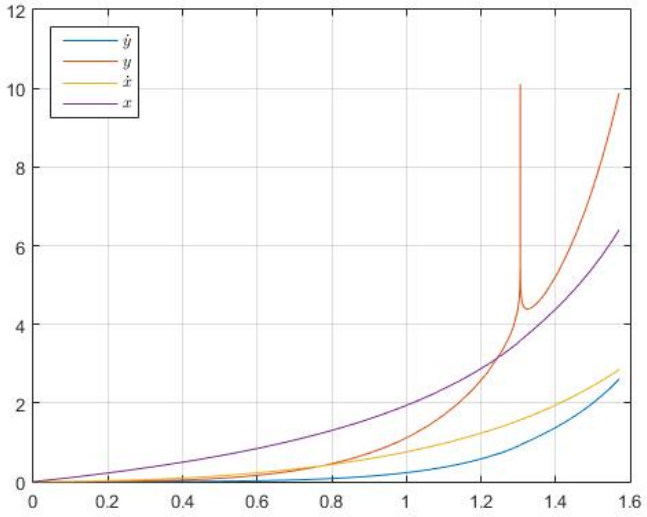


Figura 37: Solução do sistema acoplado de equações diferenciais dado pela Equação 16.

8.5 EXERCÍCIOS

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

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

2. Encontre as equações de movimento para o sistema ilustrado em Figura 38 e simule o modelo no MatLab. Considere os dados: $J_1 = 2000 kgm^2$, $J_2 = 1000 kgm^2$, $k_1 = 310^5 Nm/rad$, $k_2 = 810^4 Nm/rad$, $b_1 = 3100 Ns/rad$, $c_2 = 3210 Ns/rad$, $n = D_2/D_1 = 1,5$.

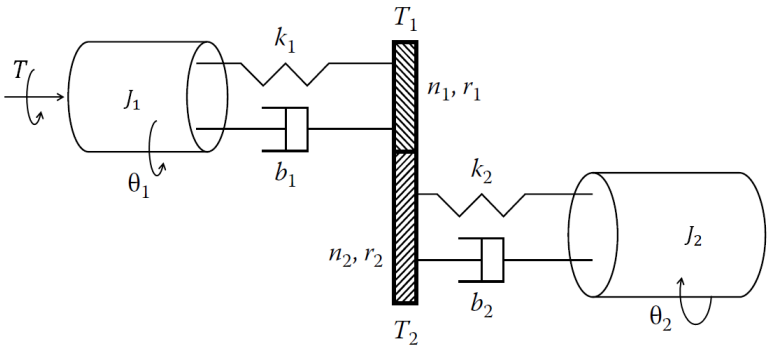


Figura 38: Trem de engrenagens de um sistema de gerador e turbina.

3. Considere em um pêndulo invertido montado em um carrinho motorizado, conforme Figura 39. O sistema de pêndulo invertido é um exemplo comumente encontrado nos livros didáticos do sistema de controle e literatura de pesquisa. Sua popularidade deriva, em parte, do fato de que é instável sem controle, ou seja, o pêndulo simplesmente cairá se o carrinho não for movido para equilibrá-lo. Além disso, a dinâmica do sistema não é linear. O objetivo do sistema de controle é equilibrar o pêndulo invertido aplicando uma força ao carrinho ao qual o pêndulo está preso. Um exemplo do mundo

real que se relaciona diretamente com este sistema de pêndulo invertido é o controle de atitude de um foguete na decolagem.

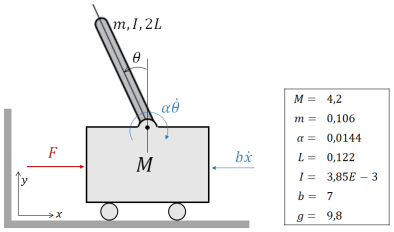


Figura 39: Pêndulo invertido montado em um carrinho motorizado.

Neste caso, consideramos um problema bidimensional onde o pêndulo não se move no plano vertical. Para este sistema, a entrada de controle é a força aplicada F que move o carrinho horizontalmente e as saídas são a posição angular do pêndulo θ e a coordenada de posição horizontal do carrinho x. Desse modo,

- Deduza as seguintes equações de movimento:

$$F - b\dot{x} = (M + m)\ddot{x} - mL\ddot{\theta}\cos\theta + mL\dot{\theta}^2\sin\theta$$
$$-\alpha\dot{\theta} = -m\ddot{x}L\cos\theta + \left(mL^2 + I\right)\ddot{\theta} - mgL\sin\theta$$

- Faça uma simulação e comente os resultados.
- Linearize o problema e compare os resultados.

4. 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 40 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.

5. O sistema de transmissão do *Porshe Panamera S E-Hybrid 2014* é mostrado na Figura 41. 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*.

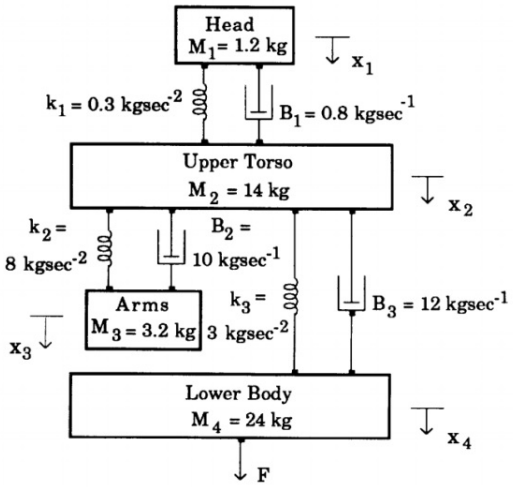


Figura 40: Modelo do corpo humano sentado. Dados médios para um humano adulto. Figura extraída de [18].

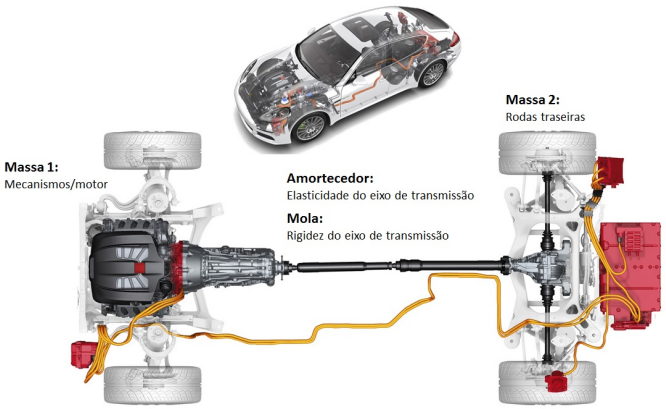


Figura 41: Sistema de transmissão do Porshe Panerama, modelo 2014.

O modelo dinâmico do sistema está ilustrado na Figura 42 e pode ser definido através das seguintes equações diferenciais,

$$\begin{aligned}
\dot{\theta}_1 &= \omega_1 \\
\dot{\theta}_2 &= \omega_2 \\
\dot{\omega}_1 &= \frac{1}{J_1}\left[k\left(\theta_2 - \theta 1\right) + d\left(\omega_2 - \omega_1\right) + \tau_m\right] \\
\dot{\omega}_2 &= \frac{1}{J_2}\left[k\left(\theta_1 - \theta 2\right) + d\omega_1 - \left(d + b\right)\omega_2\right]
\end{aligned} \tag{19}$$

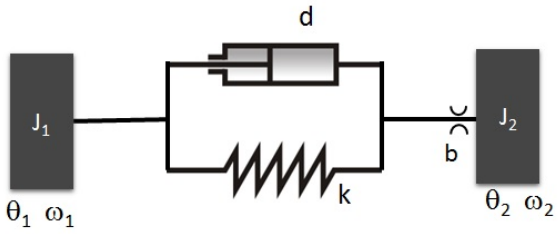


Figura 42: Esquematização do sistema MMAM.

Estude o modelo dinâmico para as variáveis definidas na Tabela 15.

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	$0.2656Nm/rad$
d	Amortecimento torcional do eixo	$3.215\ 10^{-5}Nms/rad$
τ_m	Torque do motor	$10\ 10^{-2}Nm$
b	Atrito viscoso	$1\ 10^{-5}Nms/rad$

Tabela 15: Parâmetros do sistema MMAM.

6. A equação de movimento para um foguete em vôo vertical próximo à Terra, Figura 43, pode ser obtida da segunda lei de Newton,

$$\frac{d}{dt}\left[m(t)v(t)\right] = F_T + F_D + F_G$$

onde $v(t)$ velocidade vertical ($v = 0$ quando $t = 0$).

F_T é a força de propulsão,

$$F_T = c\frac{dm}{dt},$$

onde c é a velocidade de escape. A força de arrasto, F_D é dada por,

$$F_D = -\frac{1}{2}C_D\rho A v^2$$

onde C_D é o coeficiente de arrasto, ρ é a densidade do ar, A é a área da seção transversal do foguete.

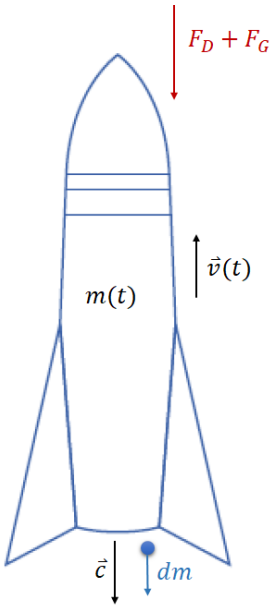


Figura 43: Foguete em lançamento vertical.

A força da gravidade F_G é dada por

$$F_G = -m(t)g$$

A massa $m(t)$, em kg, do foguete varia devido ao consumo de combustível, i.é,

$$m(t) = m_0 - \beta t$$

onde m_0 é a massa inicial. O foguete consome combustível a uma taxa β, em kg/s, por um período de τ s.

A aceleração da gravidade $g(t)$ é dada pela relação,

$$g = g_0 \frac{R^2}{(R+r)^2},$$

onde r é a altitude do foguete acima da superfície da Terra ($x = 0$ quando $t = 0$), R é o raio da Terra, g_0 é a gravidade na superfície da Terra. Como o enunciado refere-se a uma trajetória *próxima à Terra*, considera-se a gravidade g constante.

Considere os dados da Tabela 16, e determine :

- a velocidade máxima alcançada pelo foguete;
- a altitude máxima alcançada pelo foguete;
-

para os seguintes casos:

- desprezando arraste e força de propulsão constante $F_T = 4.74$N;
- considerando arraste e força de propulsão constante $F_T = 4.74$N;
- considerando arraste e força de propulsão de acordo com a listagem abaixo. DICA: use interpolação para achar a força no instante t, com o comando: `interp1(T,F,t)` ;

$\rho = 1.225 kg/m^3$	Densidade do ar
$C_d = 0.4$	Coeficiente de arrasto
$d = 0.0343m$	Diâmetro do foguete
$A = \pi * (d/2)^2 m^2$	Área da seção transversal do foguete
$\beta = 5.810^{-3} kg/s$	Taxa de queima de combustível
$m_0 = 0.1173kg$	Massa inicial
$c = 817m/s$	Velocidade de escape
$\tau = 1.86s$	Tempo de queima do combustível
$g_0 = 9.8m/s^2$	Aceleração da gravidade

Tabela 16: Parâmetros do protótipo de um foguete.

```
T = [  0        0.0310  0.0920  0.1390  0.1920
       0.2090   0.2310  0.2480  0.2920  0.3700
       0.4750   0.6710  0.7020  0.7230  0.8500
       1.0630   1.2110  1.2420  1.3030  1.4680
       1.6560   1.8210  1.8340  1.8470  1.8600];
F = [  0        0.9460  4.8260  9.9360  14.0900
       11.4460  7.3810  6.1510  5.4890  4.9210
       4.4480   4.2580  4.5420  4.1640  4.4480
       4.3530   4.3530  4.0690  4.2580  4.3530
       4.4480   4.4480  2.9330  1.3250  0]
```

Parte IV

AULA 04 - 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.

9 NÚMEROS COMPLEXOS

9.1 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
```

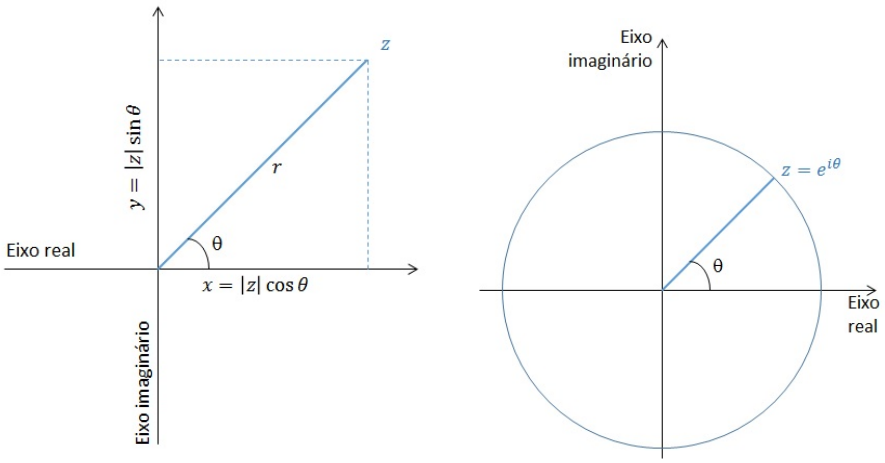
Um número complexo $z \in \mathbb{C}$ pode ser escrito na pode ser definido pelo par ordenado (x, y) de números reais x e y, ou por suas coordenadas polares r, θ,

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

onde x, y são a parte real e imaginária, respectivamente; e r e θ 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 $z \in \mathbb{C}$ feita no plano complexo está ilustrada na Figura 44a.



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

Figura 44: (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 θ (Figura 44b).

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
```

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 $z2$.

9.2 OPERAÇÕES MATEMÁTICAS

Seguem as regras básicas para operações de números complexos. Caso tenha alguma dúvida, recorra ao seu material das aulas de cálculo ou [4].

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:

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

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:

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

Para potência de um número complexo,

$$z^n = r^n\,[\cos(n\theta) + i\sin(n\theta)]$$

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

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

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
```

9.3 FUNÇÕES

Para transformar diretamente o número complexo $z = x + yi$ para polar, tem-se o comando `cart2pol`,

```
» z = 1 + 2i
z =
         1.0000 + 2.0000i
»[theta, r] = cart2pol(real(z), imag(z))
theta =
         1.1071
r =
         2.2361
```

Ou, alternativamente, `pol2cart`, para o caminho inverso,

```
» theta=1.1071; r=2.2361;
»[x, y] = pol2cart(theta, r)
x =
     1.0001
y =
     2.0000
```

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 vcoê quer saber o ângulo em graus deve fazer a conversão,

```
» 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 44b,

```
» 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
```

9.4 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 $z = 3e^{i\pi/3}$. Use as funções do MATLAB para calcular z^2, z^3, $1/z$ and $z+1$.

10 FUNÇÃO DE VARIÁVEL COMPLEXA

10.1 INTRODUÇÃO

Seja s um número complexo qualquer pertencente a um conjunto S de números complexos. Dizemos que s é uma variável complexa se a parte real e/ou imaginária forem variáveis. Particularmente, define-se s como,

$$s = \sigma + i\omega, \tag{20}$$

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

Se, para cada valor de s, o valor de outro número complexo w é determinado, então w é uma função da variável complexa s no conjunto S:

$$w = G(s)$$

A função complexa $w = G(s)$ da variável complexa s é um mapa de pontos do domínio, isto é, do plano complexo $s = \sigma + i\omega$ para pontos da imagem, isto é, para pontos no plano complexo, expressos pela soma das suas componentes real e imaginária,

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

onde $u(\sigma, \omega)$ e $v(\sigma, \omega)$ são quantidades reais.

Por exemplo,

$$\begin{aligned} G(s) &= s^2 - 2s + 3 \\ &= (\sigma + i\omega)^2 - 2(\sigma + i\omega) + 3 \\ &= (\sigma^2 - \omega^2 - 2\sigma + 3) + i(2\sigma\omega - 2\omega) \end{aligned}$$

donde,

$$\begin{aligned} u(\sigma, \omega) &= \sigma^2 - \omega^2 - 2\sigma + 3 \\ v(\sigma, \omega) &= 2\sigma\omega - 2\omega \end{aligned}$$

Sendo $G(s)$ um número complexo, obedece às mesmas definições e propriedades estabelecidas no Capítulo 9. Em particular:

- Valor absoluto de $G(s)$: $|G(s)| = \sqrt{u^2 + v^2}$
- Argumento de $G(s)$: $\theta_G = \tan\left(\frac{v}{u}\right)$

10.2 LIMITE

Uma vizinhança de um ponto s_0 é o conjunto de todos os pontos para os quais:

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

onde ϵ é alguma constante positiva. Portanto, uma vizinhança consiste em todos os pontos de um disco, ou região circular, no plano complexo, inclusive o centro s_0, mas, sem incluir o círculo de contorno.

Seja G uma função definida em todos os pontos de uma vizinhança de um ponto s_0 (não necessariamente no ponto s_0). Dizemos que o limite de G, quando s tende a s_0, é w_0, i.é,

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

se, para qualquer $\epsilon > 0$ existe um número $\delta > 0$ tal que:

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

Se existe o limite de G no ponto s_0 então este limite é único.

Teorema 1 *Sejam*

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

Então,

$$\lim_{s \to s_0} G(s) = \lim_{(\sigma,\omega) \to (\sigma_0,\omega_0)} u(\sigma, \omega) + i \lim_{(\sigma,\omega) \to (\sigma_0,\omega_0)} v(\sigma, \omega) = u_0 + iv_0$$

onde os dois últimos limites são de funções reais de duas variáveis reais.

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

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

Então

$$\lim_{s \to s_0} [G(s) \pm F(s)] = w_0 \pm W_0$$

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

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

10.3 CONTINUIDADE

Uma função F é contínua num ponto s_0 se, e somente se,

$$\lim_{s \to s_0} G(s) = G(s_0)$$

A função f diz-se contínua se for contínua em todos os pontos do seu domínio.

10.4 DERIVADA E AS RELAÇÕES DE RELAÇÕES DE CAUCHY-RIEMAN

Da mesma maneira que se deriva uma função real $g(x)$ em relação à variável real x, pode-se derivar uma função complexa $G(s)$ em relação à variável complexa s,

$$\frac{dG(s)}{ds} = \lim_{\Delta s \to 0} \frac{G(s + \Delta s) - G(s)}{\Delta s}$$

A diferença é que o incremento Δs a partir de um ponto qualquer e arbitrário s pode ocorrer em qualquer uma das infinitas direções possíveis no plano complexo s. Para que a derivada exista no ponto é necessário *que todas as infinitas direções tenham o limite exatamente igual*.

Retornando à definição,

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

onde $s = \sigma + i\omega$. O incremento da variável independente s a partir de s_0 será,

$$\Delta s = s - s_0 = \sigma + i\omega - (\sigma_0 + i\omega_0) = (\sigma - \sigma_0) + i(\omega - \omega_0) = \Delta\sigma + i\Delta\omega$$

O incremento de $G(s)$ será

$$\begin{aligned}\Delta G(s) &= G(s) - G(s_0) = u(\sigma,\omega) + iv(\sigma,\omega) - [u(\sigma_0,\omega_0) + iv(\sigma_0,\omega_0)] \\ &= [u(\sigma,\omega) + iv(\sigma,\omega) - u(\sigma_0,\omega_0)] + i\,[v(\sigma,\omega) + iv(\sigma,\omega) - v(\sigma_0,\omega_0)] \\ &= \Delta u + i\Delta v\end{aligned}$$

Como $u(\sigma,\omega)$ e $v(\sigma,\omega)$ e são funções reais teremos (para incrementos infinitesimais),

$$\begin{aligned}\Delta u &= \left.\frac{\partial u}{\partial\sigma}\right|_{\substack{\sigma=\sigma_0\\ \omega=\omega_0}} \Delta\sigma + \left.\frac{\partial u}{\partial\omega}\right|_{\substack{\sigma=\sigma_0\\ \omega=\omega_0}} \Delta\omega \\ \Delta v &= \left.\frac{\partial v}{\partial\sigma}\right|_{\substack{\sigma=\sigma_0\\ \omega=\omega_0}} \Delta\sigma + \left.\frac{\partial v}{\partial\omega}\right|_{\substack{\sigma=\sigma_0\\ \omega=\omega_0}} \Delta\omega\end{aligned}$$

Como qualquer direção tem que dar o mesmo limite, vamos escolher duas direções independentes: ao longo do eixo real σ, portanto $\Delta\sigma\ \forall$ e $\Delta\omega = 0$,

$$\frac{dG(s)}{ds} = \lim_{\Delta s\to 0}\frac{\Delta G(s)}{\Delta s} = \lim_{\Delta\sigma\to 0}\frac{\Delta u + i\Delta v}{\Delta\sigma} = \left.\frac{\partial u}{\partial\sigma}\right|_{\substack{\sigma=\sigma_0\\ \omega=\omega_0}} + i\left.\frac{\partial v}{\partial\sigma}\right|_{\substack{\sigma=\sigma_0\\ \omega=\omega_0}}$$

e ao longo do eixo imaginário $i\omega$, portanto $\Delta\sigma = 0$ e $\Delta\omega\ \forall$,

$$\frac{dG(s)}{ds} = \lim_{\Delta s\to 0}\frac{\Delta G(s)}{\Delta s} = \lim_{\Delta\omega\to 0}\frac{\Delta u + i\Delta v}{i\Delta\omega} = -i\left.\frac{\partial u}{\partial\omega}\right|_{\substack{\sigma=\sigma_0\\ \omega=\omega_0}} + \left.\frac{\partial v}{\partial\omega}\right|_{\substack{\sigma=\sigma_0\\ \omega=\omega_0}}$$

Igualando as equações acima teremos as condições necessárias e suficientes para que exista a derivada de $G(s)$ no ponto s_0, conhecidas como as *condições de Cauchy-Riemann*. As relações de Cauchy-Rieman, para qualquer ponto s_0, são dadas por,

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

Obedecer às *relações de Cauchy-Rieman* é condição necessária e suficiente para a existência da derivada de uma função em determinado ponto (ver detalhes em [4]).

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

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

Teorema 4 *Sejam u e v funções reais e univalentes das variáveis σ e ω 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 $G'(s)$ da função $G(s) = u(\sigma,\omega) + iv(\sigma,\omega)$ existe, sendo $s_0 = \sigma_0 + i\omega_0$.*

10.5 FUNÇÕES ANALÍTICAS

Uma função G de variável complexa s se diz analítica num ponto s_0, se sua derivada $G'(s)$ existe não só em s_0, como também em todo ponto s de uma vizinhança arbitrariamente pequena em torno de s_0.

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 G e suas derivadas tendendo a infinito é chamado de *polo da função*. Por exemplo, a função

$$G(s) = \frac{1}{s(s^2+4)}$$

tem pontos singulares isolados em $s = 0$, $s = 2i$ e $s = -2i$. 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 em D e 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 $Q(s) \neq 0$.

10.6 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 $\bar{s}$.

10.7 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 NÃO 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,

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

11 TRANSFORMADA DE LAPLACE

11.1 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. Depois de manipular as equações, retorna-se ao domínio do tempo t através da inversa (Figura 45).

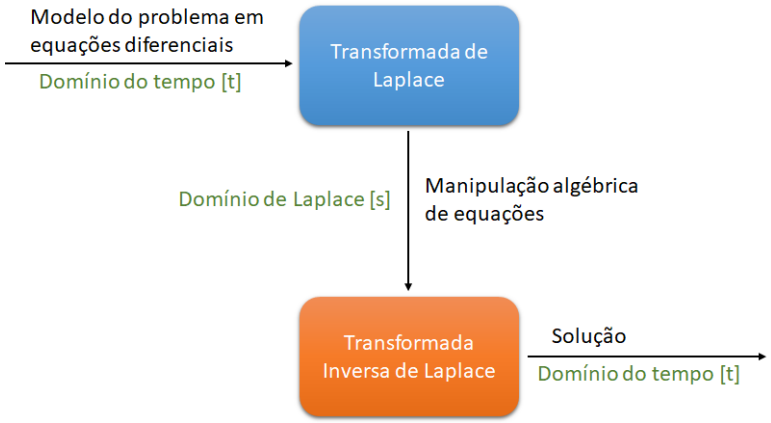


Figura 45: Transformada de Laplace: método útil para resolver equações diferenciais.

O matemático francês Pierre Simon Laplace (1749-1827) descobriu um meio de resolver equações diferenciais que consiste em,

- Multiplicar cada termo da equação por e^{-st};
- Integrar cada termo, em relação ao tempo, de 0 a ∞; s é uma constante, com unidade de 1/tempo.

A transformada de Laplace $\mathcal{L}[y(t)]$ de uma função $y(t)$ é definida como,

$$\mathcal{L}[y(t)] = \int_0^\infty y(t)e^{-st}dt = Y(s) \tag{21}$$

A edição anterior do livro texto [15] tem o Capítulo 2 dedicado ao estudo da Transformada de Laplace. Aqui, apresentamos apenas Tabela 17 e Tabela 18, com, respectivamente, os pares de transformadas de Laplace e as propriedades das transformadas.

$y(t)$	$Y(s)$
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}	$\frac{1}{(s+a)^2}$
$\frac{t^{n-1}}{(n-1)!}t^{n-1}e^{-at}$, $n = 1, 2, 3...$	$\frac{1}{(s+a)^n}$
$t^{n-1}e^{-at}$, $n = 1, 2, 3...$	$\frac{1}{(s+a)^{n+1}}$
$\sin \omega t$	$\frac{\omega}{s^2+\omega^2}$
$\cos \omega t$	$\frac{s}{s^2+\omega^2}$
$\sinh \omega t$	$\frac{\omega}{s^2-\omega^2}$
$\cosh \omega t$	$\frac{s}{s^2-\omega^2}$
$\frac{1}{a}\left(1-e^{-at}\right)$	$\frac{1}{s(s+a)}$
$\frac{1}{b-a}\left(e^{-at}-e^{-bt}\right)$	$\frac{1}{(s+a)(s+b)}$
$\frac{1}{b-a}\left(be^{-bt}-ae^{-at}\right)$	$\frac{s}{(s+a)(s+b)}$
$\frac{1}{ab}\left[1+\frac{1}{a-b}\left(be^{-at}-ae^{-bt}\right)\right]$	$\frac{1}{s(s+a)(s+b)}$
$\frac{1}{a^2}\left(1-e^{-at}-ate^{-at}\right)$	$\frac{1}{s(s+a)^2}$
$\frac{1}{a^2}\left(at-1+e^{-at}\right)$	$\frac{1}{s^2(s+a)}$
$e^{-at}\sin \omega t$	$\frac{\omega}{(s+a)^2+\omega^2}$
$e^{-at}\cos \omega t$	$\frac{s+a}{(s+a)^2+\omega^2}$
$\frac{\omega_n}{\sqrt{1-\zeta^2}}e^{-\zeta\omega_n t}\sin\left(\omega_n\sqrt{1-\zeta^2}t\right)$, $(0<\zeta<1)$	$\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$
$-\frac{1}{\sqrt{1-\zeta^2}}e^{-\zeta\omega_n t}\sin\left(\omega_n\sqrt{1-\zeta^2}t-\phi\right)$, $\phi = \arctan\frac{\sqrt{1-\zeta^2}}{\zeta}$ $(0<\zeta<1, 0<\phi<\pi/2)$	$\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$
$1-\frac{1}{\sqrt{1-\zeta^2}}e^{-\zeta\omega_n t}\sin\left(\omega_n\sqrt{1-\zeta^2}t-\phi\right)$, $\phi = \arctan\frac{\sqrt{1-\zeta^2}}{\zeta}$ $(0<\zeta<1, 0<\phi<\pi/2)$	$\frac{\omega_n^2}{s\left(s^2+2\zeta\omega_n s+\omega_n^2\right)}$
$1-\cos \omega t$	$\frac{\omega^2}{s\left(s^2+\omega^2\right)}$
$\omega t-\sin \omega t$	$\frac{\omega^3}{s^2\left(s^2+\omega^2\right)}$
$\sin \omega t-\omega t\cos \omega t$	$\frac{2\omega^3}{\left(s^2+\omega^2\right)^2}$
$\frac{1}{2\omega}t\sin \omega t$	$\frac{s}{\left(s^2+\omega^2\right)^2}$
$t\cos \omega t$	$\frac{s^2-\omega^2}{\left(s^2+\omega^2\right)^2}$
$\frac{1}{\omega_2^2-\omega_1^2}\left(\cos \omega_1 t-\cos \omega_2 t\right)$, $\left(\omega_1^2\neq\omega_2^2\right)$	$\frac{s}{\left(s^2+\omega_1^2\right)^2+\left(s^2+\omega_2^2\right)^2}$
$\frac{1}{2\omega}\left(\sin \omega t+\omega t\cos \omega t\right)$	$\frac{s^2}{\left(s^2+\omega^2\right)^2}$

Tabela 17: Transfomada de Laplace. Tabela extraída de [15].

Item	Função	Transformada	Comentário
1	$\mathcal{L}\left[ay(t) + bg(t)\right]$	$a\,\mathcal{L}\left[y(t)\right] + b\,\mathcal{L}\left[g(t)\right] = aY(s) + bG(s)$	Linearidade
2	$\mathcal{L}\left[y(t) * g(t)\right]$	$\mathcal{L}\left[Y(t)\right]\mathcal{L}\left[g(t)\right] = y(s)G(s)$	Convolução
3	$\mathcal{L}\left[\int y(t)\,dt\right]$	$\frac{Y(s)}{s} + \frac{1}{s}\left[\int_{-\infty}^{t} y(t)\,dt\right]_{t=0}$	Integral
	$\mathcal{L}\left[\frac{d}{dt}y(t)\right]$	$\frac{d}{dt}\mathcal{L}\left[y(t)\right] = sY(s) - y_0$, $(y_0 = y(0))$	
4	$\mathcal{L}\left[\frac{d^2}{dt^2}y(t)\right]$	$\frac{d^2}{dt^2}\mathcal{L}\left[y(t)\right] = s^2Y(s) - sy_0 - y_0'$	Derivada
	$\mathcal{L}\left[\frac{d^n}{dt^n}y(t)\right]$	$\frac{d^n}{dt^n}\mathcal{L}\left[y(t)\right] = s^nY(s) - s^{n-1}y_0 - \ldots - sy_0^{(n-2)} - y_0^{(n-1)}$	
5	$\mathcal{L}\left[y(\frac{t}{a})\right]$	$aY(as)$	Mudança de escala do tempo
6	$\mathcal{L}\left[e^{-at}y(t)\right]$	$Y(s+a)$	Transformada de uma função multiplicada por uma exponencial é uma função transladada.
7	$\mathcal{L}\left[y(t-a)1(t-a)\right]$	$e^{-as}Y(s)$, $(a \geqslant 0)$	Transformada de uma função transladada fica multiplicada por uma exponencial.
8	$\mathcal{L}\left[t^n y(t)\right]$	$(-1)^n \frac{d^n}{ds^n}Y(s)$	Transformada de uma função multiplicada por t é a derivada de $Y(s)$
9	$\mathcal{L}\left[\frac{y(t)}{t}\right]$	$\int_s^\infty Y(s)\,ds$	Transformada de uma função dividida por t é a integral de $Y(s)$

Tabela 18: Propriedades da Transfomada de Laplace, para $\mathcal{L}\left[y(t)\right] = Y(s)$ e $\mathcal{L}\left[g(t)\right] = G(s)$.

Nota-se, pelo item 3 da Tabela 18, se

$$\left[\int_{-\infty}^{t} y(t)\,dt\right]_{t=0} = 0$$

então integrar no domínio do tempo t equivale a dividir por s no domínio da frequência

Para o item 4 da Tabela 18, os termos $y(0)$, $sy(0)$, $y'(0)$,..., são chamados *resíduos*. Caso $y(t)$ tenha condições iniciais nulas,

$$y(0) = 0, y'(0) = 0, y''(0) = 0, ...$$

então os *resíduos* são todos nulos e derivar no domínio do tempo t equivale a multiplicar por s no domínio da frequência,

$$\mathcal{L}\left[\frac{d}{dt}y(t)\right] = sY(s)$$
$$\mathcal{L}\left[\frac{d^2}{dt^2}y(t)\right] = s^2Y(s)$$
$$\mathcal{L}\left[\frac{d^n}{dt^n}y(t)\right] = s^nY(s)$$

De acordo com o item 5 da Tabela 18, se o eixo da variável t for *encolhido* $(0 < a < 1)$, então a transformada de Laplace de $y(t)$ ficará *esticada* em s. Se, por outro lado, $(a > 1)$, então, o eixo da variável t será *esticado* e a transformada de Laplace de $y(t)$ ficará *encolhida* em s.

Calcular a transformada de Laplace $Y(s)$ de uma função $y(t)$ é bastante simples no Matlab; o comando é `Y=laplace(y,t,s)`. Porém, inicialmente, você precisa especificar que as variáveis t e s são simbólicas. Isso é feito com o comando,

```
» syms s t
```

Em seguida, define-se a função $y(t)$, e, para tornar a expressão mais legível, pode-se usar os comandos `simplify` and `pretty`.

```
Y =

(s - 5)/(s*(s + 2)^2)

   s - 5
----------
         2
s (s + 2)
```

Figura 46: Resultado visto na janela de comandos do MatLab para o exemplo de uso do comando `laplace`.

Por exemplo, a sequência de comandos para encontrar a transformada de Laplace da função,

$$y(t) = -1.25 + 3.5te^{-2t} + 1.25e^{-2t}$$

é (ver janela de comandos na Figura 46),

```
syms t s
y=-1.25+3.5*t*exp(-2*t)+1.25*exp(-2*t);
Y=laplace(y,t,s)
pretty(Y)
```

que corresponde à função $Y(s)$,

$$Y(s) = \frac{s-5}{s(s+2)^2}$$

Alternativamente, pode-se usar diretamente o comando
» `Y=laplace(-1.25+3.5*t*exp(-2*t)+1.25*exp(-2*t)).`

11.2 TRANSFORMADA INVERSA DE LAPLACE

A transformada inversa de Laplace, ou seja, a determinação da função do tempo $y(t)$ a partir da transformada $Y(s)$ pode ser obtida através da integral de inversão,

$$\mathcal{L}^{-1}[Y(s)] = y(t) = \frac{1}{2\pi i}\int_{\sigma_c-i\infty}^{\sigma_c+i\infty} Y(s)e^{st}\,ds, \qquad t > 0$$

onde σ_c, é a abcissa de convergência, valor real superior à parte real de todos os pontos singulares de $Y(s)$. O caminho de integração é paralelo ao eixo imaginário, deslocado da origem de σ_c.

O cálculo da inversão é, aparentemente, complicado. No MatLab, no entanto, o comando `ilaplace` é quase mágico... Também é preciso definir as variáveis simbólicas t e s. Por exemplo, calcula-se a transformada inversa da função anterior $Y(s)$,

$$Y(s) = \frac{s-5}{s(s+2)^2}$$

com os seguintes comandos (veja Figura 47),

```
1  syms t s
   Y=(s-5)/(s*(s+2)^2);
   y=ilaplace(Y);
   y=simplify(y)
   pretty(y)
```

```
y =

(5*exp(-2*t))/4 + (7*t*exp(-2*t))/2 - 5/4

exp(-2 t) 5   t exp(-2 t) 7   5
---------- + ------------ - -
     4             2        4
```

Figura 47: Resultado visto na janela de comandos do MatLab para o exemplo de uso do comando `ilaplace`.

A obtenção de $y(t)$, sem usar diretamente a ferramenta numérica do MatLab, pode ser mais simples do que o uso da integral de inversão já definida e um pouco assustadora. Normalmente, não se utiliza a integral de inversão de Laplace, mas, simplesmente, se consulta tabelas existentes. Para tal, deve-se adequar a função $Y(s)$ para a consulta à tabela decompondo-a em outras funções de s mais simples. Será utilizado o *Método de Expansão em Frações Parciais*.

Importante ressaltar que neste método mais simples de obtenção de $y(t)$ a partir de $Y(s)$ pressupõe-se que a solução da transformada inversa de Laplace é única. Esta condição pode ser violada no caso de admitirmos a presença de *funções nulas* na solução. Se não admitirmos a presença de funções nulas, devido ao seu pouco interesse prático, podemos invocar a unicidade da solução da transformada inversa de Laplace pelo *Teorema de Lech*,

Teorema 5 *Seja* $y(t)$ *contínua por partes em todo intervalo finito* $0 \leqslant t \leqslant N$ *e de ordem exponencial para* $t > N$*, então a transformada inversa de* $Y(s)$ *(obtida pela transformada de Laplace de* $y(t)$*) é única, ou seja*
$\mathcal{L}^{-1}[Y(s)] = y(t)$

Suponha que função $Y(s)$ pode ser expressa na forma racional abaixo, como uma função de dois polinômios em s:

$$Y(s) = \frac{B(s)}{A(s)}$$

de modo que a maior potência de s em $B(s)$ seja menor que a maior potência de s em $A(s)$ (veja em [15] o que fazer quando não for o caso).

A vantagem de utilizar esse método é que termos individuais de $Y(s)$, que resultam dessa expansão na forma de frações parciais, são funções simples de s.

11.2.1 *Expansão em frações parciais quando a transformada apresenta pólos distintos*

Quando os pólos são reais e distintos, $Y(s)$ pode ser facilmente decomposta em frações mais simples,

$$\frac{B(s)}{A(s)} = \frac{a_1}{s+p_1} + \frac{a_2}{s+p_2} + \ldots + \frac{a_n}{s+p_n} \tag{22}$$

onde os coeficientes a_k, $k = 1, \ldots n$, são chamados *resíduos* e podem ser definidos de acordo com a fórmula,

$$a_k = \left[(s+p_k)\frac{B_s}{A_s}\right]_{s=-p_k}$$

A partir daí, o cálculo da transformada inversa de $Y(s)$ é trivial:

$$\begin{aligned} y(t) = \mathcal{L}^{-1}[Y(s)] &= \mathcal{L}^{-1}[a_1 Y_1(s)] + \mathcal{L}^{-1}[a_2 Y_2(s)] + \ldots + \mathcal{L}^{-1}[a_n Y_n(s)] = \\ &= a_1 e^{-p_1 t} + a_2 e^{-p_2 t} + \ldots + a_n e^{-p_n t}, \qquad t \geqslant 0 \end{aligned} \tag{23}$$

11.2.1.1 *Utilizando MATLAB*

O Matlab tem comandos tanto para obter a expansão em frações parciais quanto para obter os pólos e zeros de $B(s)/A(s)$. Considere a seguinte função,

$$\frac{B(s)}{A(s)} = \frac{num}{den} = \frac{b_0 s^n + b_1 s^{n-1} + \ldots + b_n}{s_n + a_1 s^{n-1} + \ldots + a_n} \tag{24}$$

Dessa forma, define-se como
$num = [b_0 \quad b_1 \quad \ldots \quad b_n]$
$den = [1 \quad a_1 \quad \ldots \quad a_n]$
A função `[r,p,k]= residue(num,den)` define o resíduo `r`, os pólos `p` e o termo direto `k`, de modo que a expansão resulta em:

$$\frac{B(s)}{A(s)} = \frac{r(1)}{s-p(1)} + \frac{r(2)}{s-p(2)} + \ldots + \frac{r(n)}{s-p(n)} + k(s) \tag{25}$$

Comparando a Equação 22 e a Equação 25, percebe-se que $p(1) = -p_1$, $p(2) = -p_2$, ..., $p(n) = -p_n$, $r(1) = a_1$, $r(2) = a_2$, ..., $r(n) = a_n$.

Por exemplo, a função (extraído de [15]),

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

```
» num=[ 2 5 3 6];
» den=[ 1 6 11 6];
» [r,p,k]=residue(num,den)

r =
     -6.0000
     -4.0000
      3.0000

p =
     -3.0000
     -2.0000
     -1.0000

k =
      2
```

Dessa forma,

$$\frac{B(s)}{A(s)} = \frac{2s^3+5s^2+3s+6}{s^3+6s^2+11s+6} = \frac{-6}{s+3} + \frac{-4}{s+2} + \frac{3}{s+1} + 2$$

11.2.2 *Expansão em frações parciais quando a transformada apresenta pólos iguais*

Quando a função apresenta $r \leqslant n$ pólos iguais, $Y(s)$ pode ser decomposta em frações mais simples,

$$\frac{B(s)}{A(s)} = \frac{b_r}{(s+p_1)^r} + \frac{b_{r-1}}{(s+p_1)^{r-1}} + ... + \frac{b_1}{s+p_1} + \frac{a_{r+1}}{s+p_{r+1}} + ... \frac{a_n}{s+p_n} \tag{26}$$

onde os coeficientes a_k, $k = r+1, ...n$, são calculados como no caso anterior; e os coeficientes b_r podem ser definidos de acordo com a fórmula,

$$b_r = \left[(s+p_1)^r \frac{B_s}{A_s}\right]_{s=-p_1}$$

$$b_{r-1} = \frac{d}{ds}\left\{\left[(s+p_1)^r \frac{B_s}{A_s}\right]\right\}_{s=-p_1}$$

...

$$b_{r-i} = \frac{1}{i!}\frac{d^i}{ds^i}\left\{\left[(s+p_1)^r \frac{B_s}{A_s}\right]\right\}_{s=-p_1}$$

...

$$b_1 = \frac{1}{(r-1)i!}\frac{d^{r-1}}{ds^{r-1}}\left\{\left[(s+p_1)^r \frac{B_s}{A_s}\right]\right\}_{s=-p_1}$$

A partir daí, o cálculo da transformada inversa de $Y(s)$ é trivial:

$$\begin{aligned} y(t) = &\left[\frac{b_r}{(r-1)!}t^{r-1} + \frac{b_{r-1}}{(r-2)!}t^{r-2} + ... + b_2 t + b_1\right] e^{-p_1 t} + \\ &+ a_{r+1}e^{-p_{r+1}t} + a_{r+2}e^{-p_{r+2}t} + ... + a_n e^{-p_n t} \end{aligned} \tag{27}$$

11.2.2.1 *Utilizando MATLAB*

O Matlab tem comandos tanto para obter a expansão em frações parciais quanto para obter os pólos e zeros de $B(s)/A(s)$ quando a transformada apresenta pólos iguais. Considere a seguinte função,

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

Dessa forma, define-se como

```
» num=[0 1 2 3];
» den=[1 3 3 1];
» [r,p,k]=residue(num,den)

r =
     1.0000
     -0.0000
     2.0000

p =
     -1.0000
     -1.0000
     -1.0000

k =
     [ ]
```

Veja que os pólos, representados pelo vetor p são iguais a $s = -1$. Nesse caso, a seguinte expansão em frações parciais deve ser considerada,

$$\frac{B(s)}{A(s)} = \frac{num}{den} = \frac{1}{s+1} + \frac{0}{(s+1)^2} + \frac{2}{(s+1)^3} \tag{28}$$

Verifique,a través das funções,

```
» [num,den]=residue(r,p,K);
» printsys(num,den,'s')

num/den =
              s^2+2s+3
          ---------------
          s^3+3s^2+3s+1
```

que a equação original $B(s)/A(s)$ é recuperada.

11.3 RESOLVENDO EQUAÇÃO DIFERENCIAL COM LAPLACE

De grande importância em nosso curso é a solução de equação diferencial pela Transformada de Laplace. O método é dividido em três etapas,

1. Transformar um problema *difícil* em uma equação simples através da aplicação da transformada de Laplace (equação *subsidiária*)
2. Resolve-se a equação subsidiária através de manipulações algébricas

3. A solução da equação diferencial em função do tempo é obtida pela transformada inversa de Laplace da equação subsidiária.

Por exemplo, dada a EDO,

$$\frac{d^2}{dt^2}y(t) + 3\frac{d}{dt}y(t) + 2y(t) = e^{-t}, \qquad y(0) = 4, \frac{d}{dt}y(0) = 5,$$

Primeiro passo é obter a transformada de Laplace de $y(t)$,

$$\begin{aligned}
&\mathcal{L}[\ddot{y}] + 3\,\mathcal{L}[\dot{y}] + 2\,\mathcal{L}[y] = \mathcal{L}\left[e^{-t}\right] \\
&\mathcal{L}[y(t)] = Y(s) \\
&\mathcal{L}[\dot{y}(t)] = sY(s) - y(0) = sY(s) - 4 \\
&\mathcal{L}[\ddot{y}(t)] = s^2Y(s) - sy(0) - \dot{y}(0) = s^2Y(s) - 4s - 5 \\
&\mathcal{L}[e^{-t}] = \frac{1}{s+1}
\end{aligned}$$

de forma que,

$$Y(s) = \frac{4s^2 + 21s + 18}{s^3 + 4s^2 + 5s + 2} \tag{29}$$

e, a partir daí, a solução já foi mostrada anteriormente,

```
num=[4 21 18];
den=[1 4 5 2];
[r,p,k]=residue(num,den)
```

resultando em,

$$Y(s) = \frac{-8}{s+2} + \frac{12}{(s+1)} + \frac{1}{(s+1)^2} \tag{30}$$

e a inversa é dada por,

$$y(t) = 12e^{-t} - 8e^{-2t} + te^{-t} \tag{31}$$

Entenda a listagem a seguir utilizando o comando `solve`, onde é feita, inclusive, uma comparação com a resposta obtida com o comando `ode45`. A Figura 48 mostra a resposta no tempo.

```
syms Y t s
y0=4;                       % i.b.c.
dy0=5;                      % i.b.c.
f = exp(-t)                 % Define the right-hand side function
F = laplace(f,t,s)          % Find its Laplace transform
Y1=s*Y-y0;                  % Laplace transform of y'(t) : Y1 = s Y ...
    - y(0)
Y2 = s*Y1 - dy0;            % Laplace transform of y''(t) : Y2 = s ...
    Y1 - y'(0)
% Set the Laplace transform of the left hand side minus the right ...
    hand side
% ... to zero and solve for Y:
Sol_s=solve(Y2+3*Y1+2*Y-F,Y)
Sol_s=simplify(Sol_s)       % simplify the expression
pretty(Sol_s)               % format similar to typeset mathematics
Sol_t=ilaplace(Sol_s,s,t)   % inverse of Laplace Transform ...
    Sol_t=12.*exp(-t) - 8.*exp(-2.*t) + t.*exp(-t);
ezplot(Sol_t,[0 5])         %  plot Sol_t(t) over the domain: 0<t<5
hold on
x_0 = [4,5];
```

```
tspan = [0,5];
[t,x] = ode45(@my_func,tspan,x_0);  % solve with ODE45
plot(t,x(:,1),'*');
legend('laplace','ode45')
title('')
% Function
function xbar = my_func(t,x)
xbar = [x(2); exp(-t)-3*x(2)-2*x(1)];
end
```

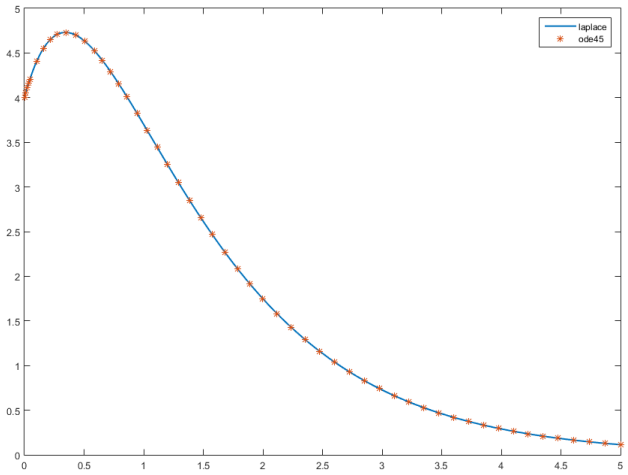


Figura 48: Resposta no domínio do tempo, para soluções por `laplace` e por `ode45`.

O próximo exemplo é, novamente, extraído da edição anterior do livro texto [15]. Como já mencionado, o capítulo 2 do livro está muito bom... Vamos resolver a seguinte equação diferencial através da Transformada de Laplace,

$$\ddot{y}(t) + 2\dot{y}(t) + 10y(t) = t^2, \qquad y(0) = 0, \qquad \dot{y} = 0$$

onde $\dot{y} = d/dt$ e $\ddot{y}(t) = d^2/dt^2$. Primeiro passo é obter a transformada de Laplace de $y(t)$,

$$\mathcal{L}[\ddot{y}] + 2\,\mathcal{L}[\dot{y}] + 10\,\mathcal{L}[y] = \mathcal{L}\left[t^2\right]$$
$$\mathcal{L}[y(t)] = Y(s)$$
$$\mathcal{L}[\dot{y}(t)] = sY(s) - y(0) = sY(s)$$
$$\mathcal{L}[\ddot{y}(t)] = s\,\mathcal{L}[\dot{y}(t)] - \dot{y}(0) = s^2Y(s) - sy(0) - \dot{y}(0) = s^2Y(s)$$
$$\mathcal{L}[t^2] = \frac{2}{s^3}$$

Então,

$$s^2Y(s) + 2sY(s) + 10Y(s) = \frac{2}{s^3}$$

ou,

$$Y(s) = \frac{2}{s^3\left(s^2 + 2s + 10\right)}$$

Próximo passo é a expansão em frações parciais, com ajuda do MATLAB,

```
% Transformada de Laplace de uma equacao diferencial de ordem 5
num=[0  0   0  0  0  2];
den=[1  2  10  0  0  0];
[r,p,k]=residue(num,den)
```

A resposta obtida foi,

```
r =
      0.0060 - 0.0087i
      0.0060 + 0.0087i
     -0.0120 + 0.0000i
     -0.0400 + 0.0000i
      0.2000 + 0.0000i

p =
     -1.0000 + 3.0000i
     -1.0000 - 3.0000i
      0.0000 + 0.0000i
      0.0000 + 0.0000i
      0.0000 + 0.0000i

k =
      []
```

Dessa forma, a resposta no domínio da frequência é,

$$Y(s) = \frac{0,0060 - 0,0087i}{s+1-3i} + \frac{0,0060 + 0,0087i}{s+1+3i} + \frac{-0,012}{s} + \frac{-0,04}{s^2} + \frac{0,2}{s^3}$$

ou,

$$Y(s) = \frac{0,012(s+1) + 0,0522}{(s+1)^2+9} - \frac{0,012}{s} - \frac{0,04}{s^2} + \frac{0,2}{s^3}$$

O último passo é a transformada inversa de Laplace de $Y(s)$,

$$\mathcal{L}^{-1}[Y(s)] = \mathcal{L}^{-1}\left[\frac{0,012(s+1) + 0,0522}{(s+1)^2+9}\right] - 0,012\,\mathcal{L}^{-1}\left[\frac{1}{s}\right] - 0,04\,\mathcal{L}^{-1}\left[\frac{1}{s^2}\right] + 0,2\,\mathcal{L}^{-1}\left[\frac{1}{s^3}\right]$$

Percebe-se que,

$$\mathcal{L}^{-1}\left[\frac{0,012(s+1) + 0,0522}{(s+1)^2+9}\right] = 0,012\,\mathcal{L}^{-1}\left[\frac{(s+1)}{(s+1)^2+3^2}\right] + 0,0174\,\mathcal{L}^{-1}\left[\frac{3}{(s+1)^2+3^2}\right]$$

$$\mathcal{L}^{-1}\left[\frac{0,012(s+1) + 0,0522}{(s+1)^2+9}\right] = 0,012e^{-t}\cos(3t) + 0,0174e^{-t}\sin(3t)$$

$$\mathcal{L}^{-1}\left[\frac{1}{s}\right] = 1$$

$$\mathcal{L}^{-1}\left[\frac{1}{s^2}\right] = t$$

$$\mathcal{L}^{-1}\left[\frac{1}{s^3}\right] = \frac{t^2}{2}$$

Portanto,

$$\mathcal{L}^{-1}[Y(s)] = y(t) = 0,012e^{-t}\cos(3t) + 0,0174e^{-t}\sin(3t) - 0,012 - 0,04t + 0,1t^2$$

11.3.0.1 *Exemplo com função definida por partes no lado direito da equação*

Seja a EDO,

$$\ddot{y} + 3\dot{y} + 2y = f(t), \qquad y(0) = 2, \qquad \dot{y} = 3$$

com a função do lado direito definita conforme,

$$f(t) = 1 \qquad t \leqslant 3$$
$$f(t) = t - 2 \qquad 3 < t < 6$$
$$f(t) = 2 \qquad t \geqslant 6$$

Para resolver, deve-se considerar a função `heaviside(t)`, que corresponde à função de step unitário $u(t)$,

$$H(t) = u(t) = \begin{cases} 0 & t \leqslant 0 \\ 1 & t > 0 \end{cases}$$

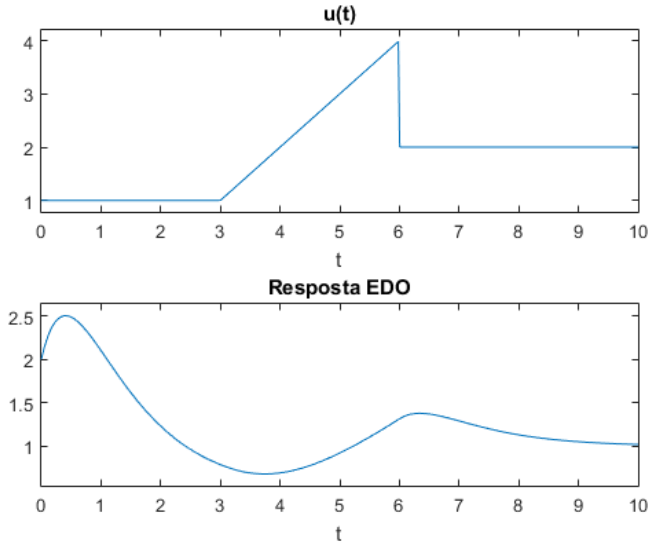


Figura 49: EDO e função linear por partes.

Dessa forma, com o código a seguir, pode-se resolver a EDO ilustrada em Figura 49.

```
clear all; close all; clc
%
syms s t Y
figure(1)
f = 1 + ((t-2)-1)*heaviside(t-3) + (2-(t-2))*heaviside(t-6);
subplot(2,1,1)
ezplot(f,[0,10])
title('u(t)')
y0=2;
dy0=3;
F = laplace(f,t,s);
Y1 = s*Y - y0;
Y2 = s*Y1 - dy0;
Sol_s = solve(Y2 + 3*Y1 + 2*Y - F, Y);
Sol_t = ilaplace(Sol_s,s,t);
subplot(2,1,2)
ezplot(Sol_t,[0,10])
title('Resposta EDO')
```

11.4 EXERCÍCIOS

Use MATLAB somente para CONFERIR suas respostas. Desenvolva as respostas e consulte também o gabarito anexo. Em alguns momentos, o MatLab pode não mostrar a resposta da transformada inversa exatamente como você gostaria. Por exemplo [1], a transformada

$$Y(s) = \frac{10(s+2)}{s(s^2+4s+5)}$$

possui a inversa dada por,

```
» Y=10*(s+2)/(s*(s^2+4*s+5));
» ilaplace(Y)

ans =
        4 - 4*exp(-2*t)*(cos(t) - sin(t)/2)
```

Isto é,

$$\mathcal{L}^{-1}[Y(s)] = y(t) = u(t)\left[4 - 4e^{-2t}\cos(t) + 2e^{-2t}\sin(t)\right]$$

que pode ser reescrita como

$$y(t) = u(t)\left[4 - 4.47e^{-2t}\cos(t - 153,4^o)\right]$$

a partir das relações trigonométricas,

$$x\sin(\theta) + y\cos(\theta) = R\sin(\theta + \beta)$$
$$x\sin(\theta) - y\cos(\theta) = R\cos(\theta + \beta)$$

com $R = \sqrt{(x^2 + y^2)}$ e $\beta = \text{atan}(y/x)$.

O Matlab frequentemente exprime a transformada inversa de Laplace em termos de sinh e cosh. Usando as definições,

$$\sinh(s) = \frac{e^s + e^{-s}}{2} \qquad \cosh(s) = \frac{e^s - e^{-s}}{2}$$

é possível reescrever a expressão hiperbólica como uma função de exponenciais.

Além disso, você pode encontrar a função `heaviside(t)`, que corresponde à função de step unitário $u(t)$.

Agora, aos exercícios...

1. Calcule a transformada de Laplace das seguintes equações diferenciais,
   (a) $\frac{d}{dt}y(t) = 3 - 2t$, $y(0) = 0$
   (b) $\frac{d}{dt}y(t) = e^{-3t}$, $y(0) = 4$
   (c) $\frac{d}{dt}y(t) + y(t) = f(t)$, $y(0) = a$
   (d) $\frac{d}{dt}y(t) + y(t) = e^t$, $y(0) = 1$
   (e) $\frac{d}{dt}y(t) - 5y(t) = 0$, $y(0) = 2$
   (f) $\frac{d}{dt}y(t) - 5y(t) = e^{5t}$, $y(0) = 0$
   (g) $\frac{d^2}{dt^2}y(t) = 1 - t$, $y(0) = 0, \dot{y}(0) = 0$
   (h) $\frac{d^2}{dt^2}y(t) + 16y(t) = 5\delta(t-1)$, $y(0) = 0, \dot{y}(0) = 0$

1 Baseados em http://www.seas.upenn.edu/ ese216/handouts/Chpt13LaplaceTransformsMATLAB.pdf

(i) $\frac{d^2}{dt^2}y(t) + 16y(t) = 16u_3(t) - 16$, $y(0) = 0, \dot{y}(0) = 0, u_3(t) = u(t-3)$

(j) $\frac{d^2}{dt^2}y(t) + 4y(t) = \cos t$, $y(0) = a, \dot{y}(0) = b$

(k) $\frac{d^2}{dt^2}y(t) + y(t) = te^{-t}$, $y(0) = a, \dot{y}(0) = b$

(l) $\frac{d^2}{dt^2}y(t) - y(t) = e^t$, $y(0) = 1, \dot{y}(0) = b$

(m) $\frac{d^2}{dt^2}y(t) - \frac{d}{dt}y(t) - 2y(t) = 4t^2$, $y(0) = 1, \dot{y}(0) = 4$

(n) $y(t) = t + \int_0^t -y(\tau)\sin(-t+\tau)d\tau$

(o) $y(t) = t^2\cos^2 t$

2. Obtenha a transformada inversa de Laplace das expressões abaixo.

NÍVEL 0,

(a) $Y(s) = \frac{1}{s^2+b^2}$

(b) $Y(s) = \frac{a}{(s+a)^2+b^2}$

(c) $Y(s) = \frac{1}{s^n}$

(d) $Y(s) = \frac{1}{(s-a)^n}$

NÍVEL 1,

(e) $Y(s) = \frac{4}{s-2} - \frac{3}{s+5}$

(f) $Y(s) = \frac{s+5}{s^2+9}$

(g) $Y(s) = \frac{8(s+2)-4}{(s+2)^2+25}$

(h) $Y(s) = \frac{4}{s} - \frac{1}{s^2} + \frac{5}{s^3} + \frac{2}{s^4}$

(i) $Y(s) = \frac{10}{(s-5)^2} + \frac{2}{(s-5)^3}$

(j) $Y(s) = \frac{1}{(s-5)^2} + \frac{2}{(s-5)^3}$

(k) $Y(s) = \frac{1}{s^2+6s+13}$

(l) $Y(s) = \frac{6}{s} - \frac{1}{s-8} + \frac{4}{s-3}$

(m) $Y(s) = \frac{19}{s+2} - \frac{1}{3s-5} + \frac{7}{s^5}$

(n) $Y(s) = \frac{3+6s}{s^2+25}$

(o) $Y(s) = \frac{8}{3s^2+12} + \frac{3}{s^2-49}$

(p) $Y(s) = \frac{6s-5}{s^2+7}$

NÍVEL 2,

(q) $Y(s) = \frac{1-3s}{s^2+8s+21}$

(r) $Y(s) = \frac{3s-2}{2s^2-6s-2}$

NÍVEL 3, EXPANSÃO EM FRAÇÕES PARCIAIS,

(s) $Y(s) = \frac{s+7}{s^2-3s-10}$

(t) $Y(s) = \frac{1}{s(s^3+6s^2+11s+6)}$

(u) $Y(s) = \frac{s+1}{s(s^2+4s+4)}$

(v) $Y(s) = \frac{s^2+2s+3}{(s+1)^3}$

(w) $Y(s) = \frac{86s-78}{(s+3)(s-4)(5s-1)}$

(x) $Y(s) = \frac{2-5s}{(s-6)(s^2+11)}$

(y) $Y(s) = \frac{25}{s^3(s^2+4s+5)}$

3. Continue obtendo a transformada inversa de Laplace das expressões dadas (se mantivesse no mesmo item anterior, numeração ultrapassaria o z, e esses são nível 4).
   (a) $Y(s) = s + 1s^3 + s^2 + s$
   (b) $Y(s) = 6s + 3s^2$
   (c) $Y(s) = 5s + 2s^3 + 5s^2 + 8s + 4$
   (d) $Y(s) = \frac{1}{s^2(s^2+\omega^2)}$
   (e) $Y(s) = \frac{5e^{-s}}{s+1}$
   (f) $Y(s) = \frac{10(s+2)(s+4)}{(s+1)(s+3)(s+5)^2}$
   (g) $Y(s) = \frac{s^4+5s^3+6s^2+9s+30}{s^4+6s^3+21s^2+46s+30}$
   (h) $Y(s) = \frac{\omega_n^2}{s(s^2+2s\zeta\omega_n+\omega_n^2)}$, $(0 < \zeta < 1)$

4. Obtenha a transformada inversa de Laplace das transformadas abaixo. Siga a seguinte regra,

$$\mathcal{L}^{-1}\left[e^{-cs}F(s)\right] = u_c(t)\,\mathcal{L}\left[F(s)\right](t-c)$$

   isto é, e^{-cs} é substituído por $u_c(t) = u(t-c)$ e a transformada de Laplace $F(s)$ é retirada de tabelas, mas substituindo t por $t-c$.
   (a) $Y(s) = \frac{e^{-2s}}{s} + \frac{6e^{-3s}}{s}$
   (b) $Y(s) = e^{-3s}\left(\frac{1}{s^2} + \frac{5}{s^3}\right)$
   (c) $Y(s) = \frac{6}{s} + \frac{e^{-s}}{s^2+4}$
   (d) $Y(s) = \frac{e^{-5s}(s+1)}{(s+1)^2+16}$
   (e) $Y(s) = \frac{4e^{-2s}}{s-3} + \frac{e^{-5s}}{s+9}$
   (f) $Y(s) = \frac{e^{-10s}}{(s-3)^2}$
   (g) $Y(s) = \frac{e^{-7s}}{s} + \frac{e^{-11s}}{(s-2)^3}$

5. Resolva as seguintes equações diferenciais,
   (a) $2\ddot{y}(t) + 7\dot{y}(t) + 3y(t) = 0, \qquad y(0) = 3, \qquad \dot{y}(0) = 0$
   (b) $5\ddot{y}(t) + 20\dot{y}(t) + 15y(t) = 30u(t) - 4\dot{u}(t)$, onde $u(t)$ é a função degrau unitário e $y(0) = 5$ e $\dot{y}(0) = 1$.
   (c) $\ddot{y}(t) + \dot{y}(t) + y(t) = g(t)$, em que $y(0) = 1$, $\dot{y}(0) = 0$ e $g(t) = \begin{cases} 0 & 0 \leqslant t < 1 \\ 1 & t \geqslant 1 \end{cases}$
   (d) $\dddot{y}(t) + 6\ddot{y}(t) + 11\dot{y}(t) + 6y(t) = u(t)$, $y(0) = 0$, $\dot{y}(0) = 0$, $\ddot{y}(0) = 0$.

6. A equação a seguir descreve o movimento de um sistema massa-mola, com atrito,

   $3\ddot{y} + 39\dot{y} + 120y = 10u(t)$

   em que $u(t)$ é uma função degrau unitário.
   (a) Plote $y(t)$ para condições iniciais nulas: $y(0) = \dot{y}(0) = 0$
   (b) Plote $y(t)$ para $y(0) = 0$ e $\dot{y}(0) = 10$. Discuta o efeito da velocidade inicial não nula.

7. Dado o modelo da suspensão de duas massas conforme Figura 50, gere o gráfico das respostas $x_1(t)$ e $x_2(t)$, sendo $y(t)$ uma função degrau unitária e condições iniciais nulas. Dados: $m_1 = 250kg$, $m_2 = 40kg$, $k_1 = 15kN/m$, $k_2 = 150kN/m$ e $b_1 = 1917Ns/m$.

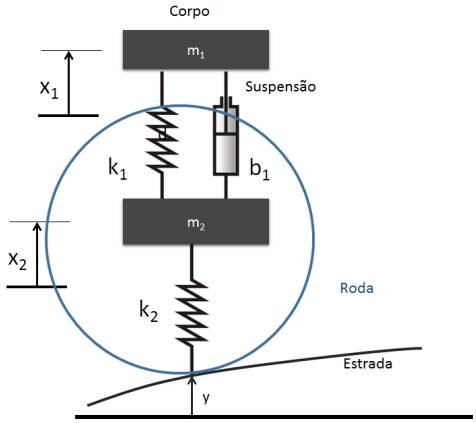
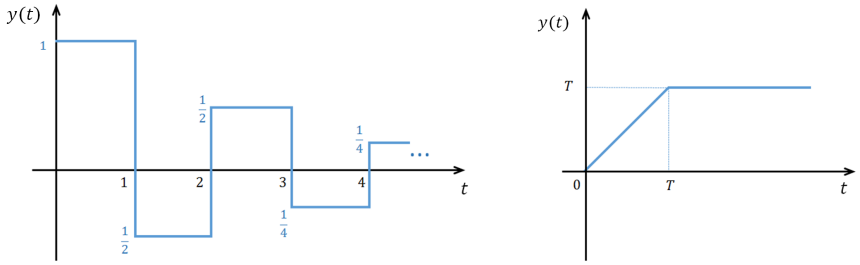


Figura 50: Modelo de suspensão de duas massas. Figura extraída de [16]

8. Calcule a transformada de Laplace de $y(t)$,
   (a) de uma onda quadrada decrescente, representada na Figura 51a;
   (b) da função representada na Figura 51b.



(a) *Onda quadrada descrescente.* (b) *Rampa.*

Figura 51: A interface do caderno.

9. Admitindo condições iniciais nulas, resolva a equação diferencial

$$\dot{y}(t) + y(t) = g(t),$$

onde $g(t)$ é representada na Figura 52.

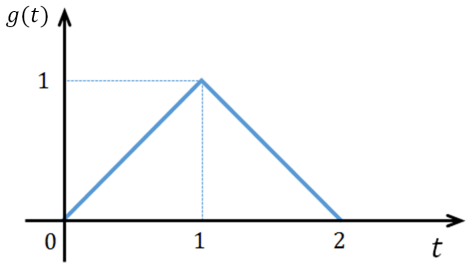


Figura 52: Função crescente-decrescente.

10. O MatLab mostra a seguinte resposta para a transformada inversa de uma função $Y(s)$,

```
ans =
       2 heaviside(t-10) exp(-5/2t+25) sinh(1/2t-5)
```

Reescreva a expressão, usando a definição de `sinh` e `heaviside`.

12 FUNÇÃO DE TRANSFERÊNCIA

A função de transferência de um sistema de equação diferenciais lineares é definida como a relação da transformada de Laplace da saída pela transformada de Laplace da entrada, para condições iniciais nulas (Figura 53),

$$G(s) = \frac{Y(s)}{R(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + ... + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + ... + a_1 s + a_0} = \frac{b_m}{a_n} \frac{(s-z_1)(s-z_2)...(s-z_m)}{(s-p_1)(s-p_2)...(s-p_n)}.$$

onde: z_i são os zeros e p_i são os pólos da função de transferência.

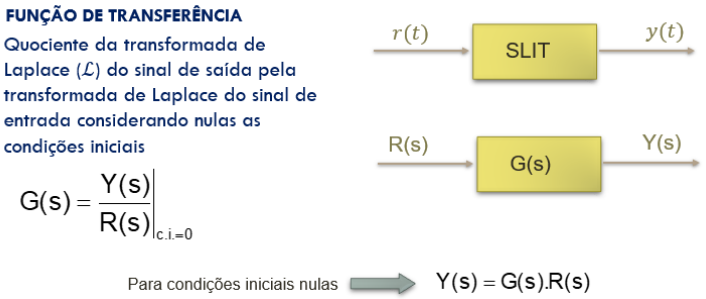


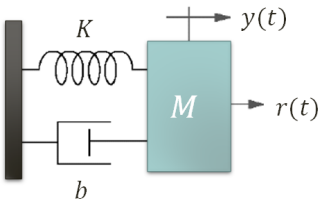
Figura 53: Função de transferência.

A função de transferência é um conceito potente para descrever o comportamento de sistemas do ponto de vista de entrada/saída. Deve-se ressaltar que, para SLITs, a função de transferência caracteriza completamente o sistema do ponto de vista de entrada-saída.

Importantes características da função de transferência são,

- A ordem da função de transferência é a maior potência de s no denominador do polinômio, que é a ordem da equação diferencial equivalente. O sistema é chamado de n-ésima ordem.
- A FT de um sistema é uma propriedade que independende da natureza e da magnitude da entrada;
- A FT não fornece informações a respeito da estrutura física do sistema. A FT de sistemas fisicamente diferentes podem ser idênticas;
- Se a FT de um sistema é conhecida, a resposta do mesmo pode ser analisada para diferentes formas de excitação (entrada), com a finalidade de compreender a natureza e o comportamento do sistema;
- Se a FT pode ser obtida experimentalmente pela introdução de sinais de entrada conhecidos e estudando-se as respostas obtidas.
- Os pólos e zeros tem um papel importante na determinação do comportamento dinâmico do sistema. Podemos visualizar o tipo de comportamento dinâmico associado a cada tipo de pólo, e esse assunto será tratado em detalhes no estudo de sistemas de primeira e segunda ordem.

A título de ilustração, a função de transferência de um sistema mecânico massa-mola-amortecedor é mostrado na Figura 54.



$$M\frac{d^2}{dt^2}y(t) + b\frac{d}{dt}y(t) + K\,y(t) = r(t)$$

$$\mathcal{L}\left[M\frac{d^2}{dt^2}y(t)\right] + \mathcal{L}\left[b\frac{d}{dt}y(t)\right] + \mathcal{L}[K\,y(t)] = \mathcal{L}[r(t)\,]$$

$$M[s^2Y(s) - s\,y(0) - \dot{y}(0)\,] + b[sY(s) - \,y(0)] + KY(s) \;= R(s)$$

$$Ms^2Y(s) + bsY(s) + KY(s) \;= R(s)$$

$$Y(s)[Ms^2 + bs + K] = R(s)$$

$$\boxed{\frac{Y(s)}{R(s)} = \frac{1}{Ms^2 + bs + K}}$$

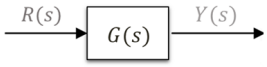


Figura 54: Exemplo da definição de função de transferência.

A função de transferência pode ser definida de várias maneiras, no MATLAB. Por exemplo, através dos comandos (ver Figura 55)

```
» num=[ 0 0 4 16 12]
» den=[ 1 12 44 48 0]
» sys=tf(num,den)
```

```
num =

     0     0     4    16    12


den =

     1    12    44    48     0


sys =

          4 s^2 + 16 s + 12
  ----------------------------
  s^4 + 12 s^3 + 44 s^2 + 48 s

Continuous-time transfer function.
```

Figura 55: Comando MATLAB `tf`.

Alternativamente, pode-se também pode entrar no sistema com pólos, zeros e ganho. Por exemplo, um sistema de zeros duplos localizados em -1 e 1, e pólos duplos localizados em j e $-$j (Figura 56),

```
» z=[-1;-1;1;1]
» p=[-j;-j;j;j]
» K=0.5
» sys=zpk(z,p,K,-1)
```

Para obter zeros e pólos da expressão $B(s)/A(s)$, o comando MATLAB é `tf2zp`. Por exemplo, para expressão,

$$\frac{B(s)}{A(s)} = \frac{4s^2 + 16s + 12}{s^4 + 12s^3 + 44s^2 + 48s}$$

os zeros, pólos e ganho K são obtidos da seguinte forma,

```
z =

    -1
    -1
     1
     1


p =

   0.0000 - 1.0000i
   0.0000 - 1.0000i
   0.0000 + 1.0000i
   0.0000 + 1.0000i


K =

    0.5000


sys =

  0.5 (s+1)^2 (s-1)^2
  -------------------
      (s^2 + 1)^2

Continuous-time zero/pole/gain model.
```

Figura 56: Comando MATLAB `zpk`.

```
» num=[ 0 0 4 16 12];
» den=[ 1 12 44 48 0];
» [z,p,K]=tf2zp(num,den)

num/den =
            -3
            -1


p =
            0
            -6.0000
            -4.0000
            -2.0000


K =
            4
```

Os zeros estão em $s = -3, -1$, e os pólos estão em $s = 0, -6, -4, -2$; o ganho $K = 4$. A partir dos zeros, pólos e ganho, é também possível obter a relação num/den original com as funções `zp2tf` e `printsys`,

```
z=[-1;-3]; p=[0;-2;-4;-6]; K=4;
[num,den]=zp2tf(z,p,K);
printsys(num,den,'s')
```

Repare que, para uso da função `zp2tf` os vetores z e p são coluna e, portanto, os valores estão separados por ponto e vírgula e não por espaço, como no caso do uso da função `tf2zp`.

Para a seguinte função de transferência,

$$G(s) = \frac{s+1}{s^2 - s + 0.5} \tag{32}$$

tem-se os pólos e zeros definidos com os comandos,

```
num=[ 1 1];
den=[ 1 -1 0.5];
[z,p,K]=tf2zp(num,den)
zplane(z,p)
```

Na listagem, o comando `zplane(z,p)` (ou `zplane(num,den)`) irá plotar os pólos (representados por $\circ$) e os zeros (representados por $\times$), conforme Figura 57.

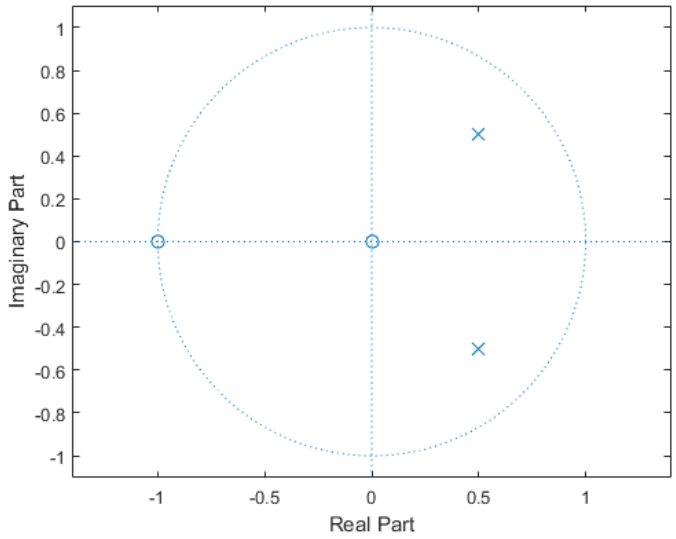


Figura 57: Zeros da função de transferência.

O método analítico para encontrar a resposta no domínio do tempo requer a transformada inversa de Laplace da saída $Y(s)$. Isso já estudamos no Capítulo 11. Aprendemos também que o MATLAB ajuda neste processo, calculando a expansão em frações parciais de $Y(s)$ usando o comando `[r,p,k]=residue(num,den)`, de forma que,

$$G(s) = \sum_i \frac{r(i)}{1 - p(i)} + k \tag{33}$$

Os resíduos são armazenados em `r`, os polos correspondentes são armazenados em `p` e o ganho é armazenado em `k`. Uma vez que a expressão da expansão em frações parciais é conhecida, uma expressão analítica para $y(t)$ pode ser calculada manualmente. Porém, aprendemos no Capítulo 7 um método numérico para encontrar a resposta para uma entrada específica através dos comandos,

`step`

`impulse`
`lsim`
Por exemplo, dada a função de transferência,

$$G(s) = \frac{2}{s+2}$$

sua resposta à função impulso é obtida com os seguintes comandos (Figura 58),

```
num = 2; den = [1 2];
t = 0:3/300:3; % 3 s de simulacao
sys=tf(num,den)
y = impulse(sys,t);
plot(t,y)
```

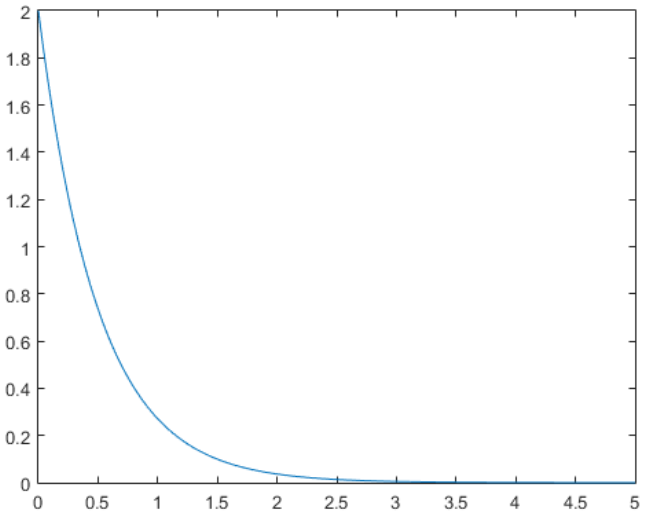


Figura 58: Resposta da função para uma entrada `impulse`.

12.1 EXERCÍCIOS

1. A função de transferência de um termômetro (ou seja, transformada de Laplace da saída pela transformada de Laplace da entrada), é

   $$G(s) = \frac{1}{90s+1}$$

   Responda às seguintes questões,

   (a) Quanto tempo você deverá esperar para medir sua febre de 40°C?

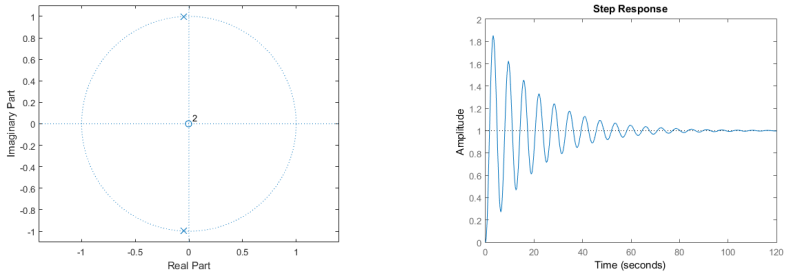
   (b) Em qual instante de tempo o termômetro atingiu 63,2% do real valor da febre?

   (c) Modifique a equação para que o termômetro atinja o valor definido no item anterior duas vezes *mais rápido*.

2. Compare a resposta da função de transferência,

   $$G(s) = \frac{1}{s^2+as+b}$$

   com os seguintes valores de a e b,



(a) *Polos e zeros.* (b) *Resposta temporal.*

Figura 59: Análise da resposta da função de transferência definida com $a = 0.1, b = 1$.

(a) $a = 0.1, b = 1$;

(b) $a = 1, b = 1$;

(c) $a = 2, b = 1$;

(d) $a = 1, b = -1$.

Para comparação, plote a função degrau `step(num,bden)` e `zplane(num,den)`. Discuta a relação entre comportamentos e zeros. Por exemplo, o primeiro item pode ser obtido com a seguinte listagem,

```
        num=[1];
    den=[1, 0.1, 1];
    G = tf(num, den)
    zplane(num,den)
    figure(2)
    step(G)
```

cuja resposta é apresentada na Figura 59.

Parte V

SIMULAÇÃO DE SISTEMAS DINÂMICOS COMPLEXOS

No Simulink, é muito simples simular um modelo matemático que representa um sistema físico através de diagramas de blocos. Existe uma ampla variedade de blocos disponível nas bibliotecas fornecidas para representar vários fenômenos. Uma das principais vantagens de empregar Simulink (e simulação em geral) para a análise de sistemas dinâmicos é que ele nos permite analisar rapidamente a resposta de sistemas complicados que podem ser proibitivamente difíceis de avaliar analiticamente. O Simulink é capaz de aproximar numericamente as soluções para modelos matemáticos que não podemos ou não desejamos resolver *à mão*...

http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=SimulinkModeling

13 DIAGRAMA DE BLOCOS

Diagrama de blocos é uma representação gráfica das funções desempenhadas por cada componente e o fluxo de sinais entre eles. Descreve o interrelacionamento que existe entre os vários componentes.

A Figura 60 ilustra o uso de um diagrama de blocos, neste caso, para modelar a resposta de vôo de uma abelha. O exemplo é extraído de [1]. A dinâmica de vôo de um inseto é incrivelmente complexa, envolvendo uma coordenação cuidadosa dos seus músculos para manter o vôo estável em resposta a estímulos externos. Uma característica conhecida de muitos insetos é sua capacidade de voar contra o vento, fazendo uso do fluxo óptico em seus olhos compostos como um mecanismo de feedback. Grosso modo, a mosca controla sua orientação para que o ponto de contração do campo visual seja centrado em seu campo visual.

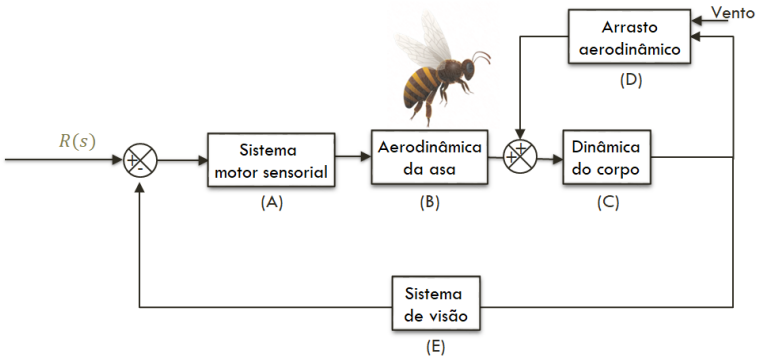


Figura 60: Diagrama de blocos: vôo de uma abelha contra o vento. A parte mecânica consiste da dinâmica de corpo rígido da abelha, o arrasto devido à resistência do ar e as forças geradas nas asas. O movimento causa mudança no campo de visão da abelha,e essa informação é usada para controlar o movimento das asas (através de um sistema motor sensorial), fechando o loop. A Figura é adaptada de [1].

Para entender esse comportamento complexo, podemos decompor a dinâmica geral do sistema em uma série de subsistemas interconectados (ou blocos). Podemos modelar o sistema de navegação de insetos por meio de uma interconexão de cinco blocos: o sistema sensorial motor (A) retira a informação do sistema visual (E) e gera comandos musculares que tentam dirigir a mosca de modo que o ponto de contração seja centrado. Estes comandos musculares são convertidos em forças através do flapping das asas (B) e das forças aerodinâmicas resultantes que são produzidas. As forças das asas são combinadas com o arrasto (D) para produzir uma força líquida no corpo da mosca. A velocidade do vento entra pela aerodinâmica do arrasto. Finalmente, a dinâmica do corpo (C) descreve como o inseto translada e gira em função das forças resultantes aplicadas nele. A posição, a velocidade e a orientação dos insetos são realimentadas para os sistemas aerodinâmico de arrasto e de visão como entradas.

Cada um dos blocos no diagrama pode ser um subsistema complicado. Por exemplo, o sistema visual de uma abelha consiste de cerca de 5500 *olhinhos* por olho, e o sistema sensorial-motor tem cerca de 200k neurônios que são usados para processar informações [9, 21]. Um diagrama de blocos mais detalhado do sistema de controle de vôo de insetos mostraria as interconexões entre esses elementos,

mas aqui usamos um bloco para representar como o movimento da mosca afeta a saída do sistema visual e um segundo bloco para representar como o campo visual é processado pelo cérebro da mosca para gerar comandos musculares. Como enfatizado em [1]:*The choice of the level of detail of the blocks and what elements to separate into different blocks often depends on experience and the questions that one wants to answer using the model. One of the powerful features of block diagrams is their ability to hide information about the details of a system that may not be needed to gain an understanding of the essential dynamics of the system.*.

A operação funcional do sistema pode ser visualizada mais facilmente pelo exame do diagrama de blocos do que pelo exame do próprio sistema físico. Os diagramas de blocos permitem inferir a relação global entre entradas e saídas e, portanto, analisar a estabilidade e o desempenho do sistema.

Um exemplo clássico é a representação do espaço de estados através de diagrama de blocos, conforme ilustra a Figura 61. Como já definido no Capítulo 7, no domínio do tempo, tem-se

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)r(t) \qquad \text{equação dos estados} \\ y(t) &= C(t)x(t) + D(t)r(t) \qquad \text{equação de saída} \end{aligned} \tag{34}$$

Usualmente lidamos com Sistemas Lineares Invariantes no tempo (SLIT). Nesse caso as matrizes $A, B, C e D$ são constantes. Saídas futuras dependem somente do estado presente e entradas futuras.

No domínio da Transformada de Laplace tem-se,

$$\begin{aligned} \mathcal{L}\{\dot{x}(t)\} = &= A\,\mathcal{L}\{x(t)\} + B\,\mathcal{L}\{r(t)\} \\ \mathcal{L}\{y(t)\} &= C\,\mathcal{L}\{x(t)\} + D\,\mathcal{L}\{r(t)\} \end{aligned} \tag{35}$$

ou seja,

$$\begin{aligned} sX(s) &= AX(s) + BR(s) \\ Y(s) &= CX(s) + DR(s) \end{aligned} \tag{36}$$

13.1 PARTES DE UM DIAGRAMA DE BLOCOS

A Figura 62 define partes importantes de um diagrama. Dentre eles,

BLOCO - Todas as variáveis do sistema são representadas por blocos funcionais. A função de transferência dos componentes é normalmente incluída nos blocos correspondentes;

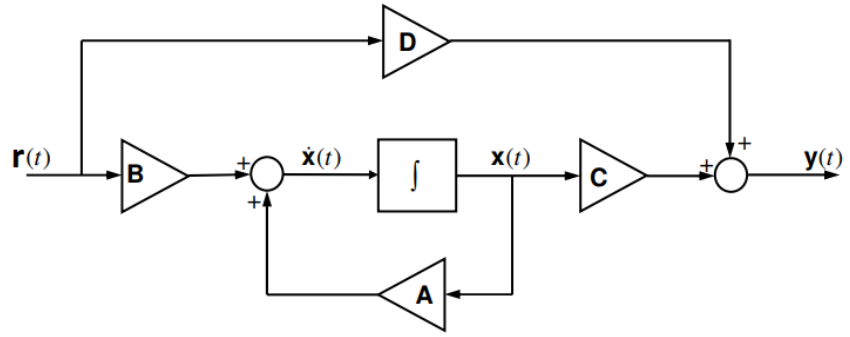
LINHAS COM SETAS - conectam os blocos e indicam a direção do fluxo de sinais. O sinal pode passar somente no sentido indicado pelas setas;

PONTO DE SOMA - indica uma operação de adição ou subtração;

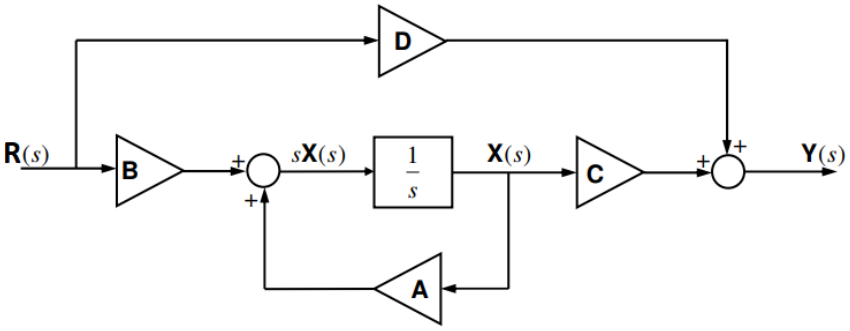
PONTO DE RAMIFICAÇÃO - ponto a partir do qual o sinal proveniente de um bloco vai para outros blocos ou pontos.

13.2 SIMPLIFICAÇÃO DE DIAGRAMA DE BLOCOS

A simplificação objetiva reduzir blocos entrelaçados de subsistemas em um bloco unificado ou uma função de transferência (chamada aqui de TF). Dessa forma, com uma forma compacta global para todo um sistema complexo, com vários subsistemas, podemos analisá-lo através da equação que descreve sua dinâmica.



(a) *Domínio do tempo* t.



(b) *Domínio da Transformada de Laplace* s.

Figura 61: Espaço de estados representado em diagrama de blocos.

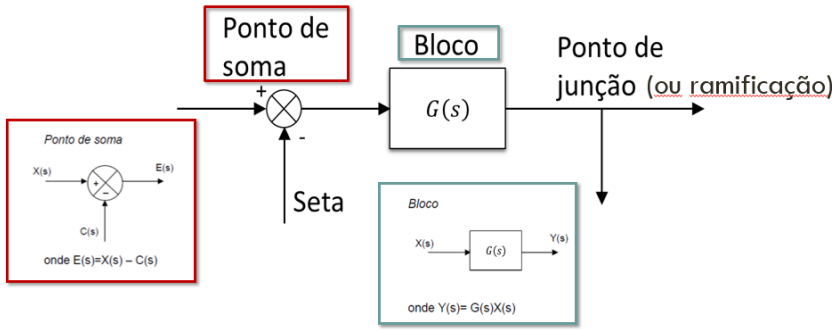


Figura 62: Partes de um diagrama de blocos.

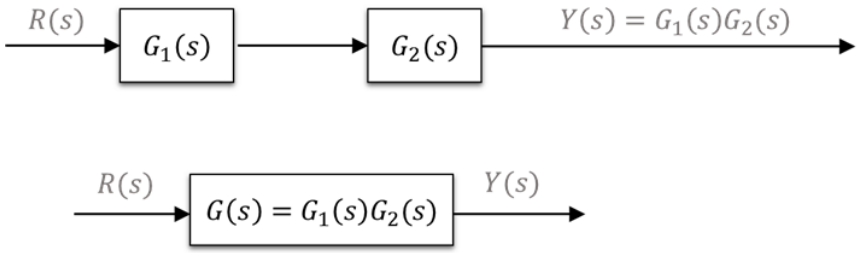


Figura 63: Diagrama de blocos em série. A função de transferência de uma série é o produto da função de transferência dos elementos da série.

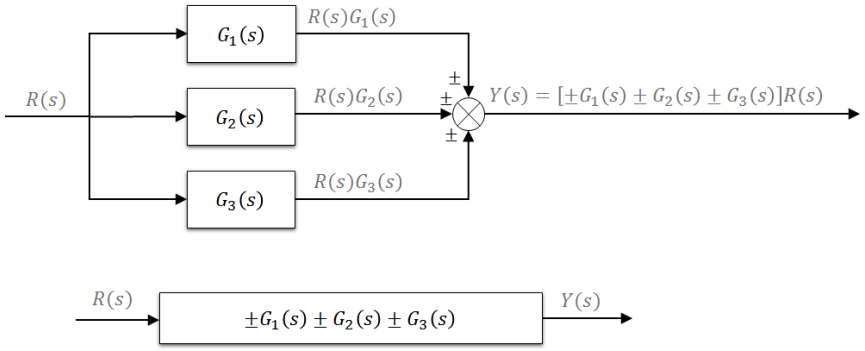


Figura 64: Blocos em paralelo.A função de transferência de blocos em paralelo é a soma da função de transferência desses blocos.

Os blocos podem estar conectados em série (Figura 63) ou em paralelo (Figura 64).

Existe a possibilidade ainda de retroalimentação do sistema Figura 65. A saída $Y(s)$ é realimentada ao somador, em que é comparada à referência de entrada $R(s)$, gerando o erro $E(s)$. Quando a saída é realimentada ao somador para comparação dos sinal de entrada, é necessário converter a forma do sinal de saída à do sinal de entrada $B(s) = H(s)Y(s)$. Essa conversão é realizada por meio do elemento de realimentação, com função de transferência $H(s)$.

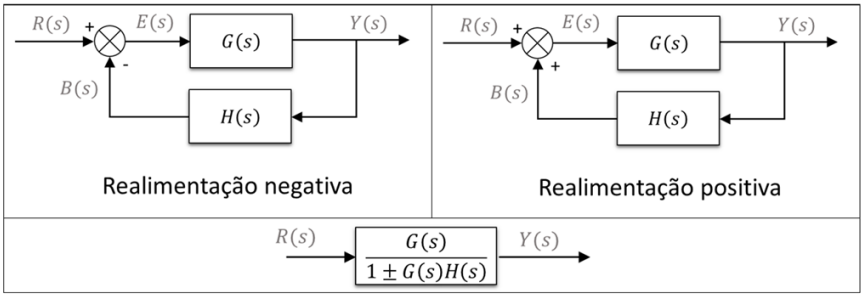


Figura 65: Sistema com retroalimentação.

Sabe-se que a saída é $Y(s)$,

$$\begin{aligned} Y(s) &= G(s)E(s) \\ E(s) &= R(s) \pm B(s) = R(s) \pm H(s)Y(s) \\ Y(s) &= G(s)\,[r(s) - H(s)Y(s)] \end{aligned} \tag{37}$$

de modo que

$$\frac{Y(s)}{R(s)} = \frac{G(s)}{1 \mp G(s)H(s)} \tag{38}$$

Considerando que o sistema com duas entradas $X(s)$ e $P(s)$ é linear, aplica-se o princípio da superposição:*A saída de um sinal formado pela combinação linear de diferentes sinais, é igual à combinação dos sinais de saída gerados por cada sinal separadamente.* A Figura 66 exemplifica a análise de um sistema com duas entradas. A resposta devido à aplicação simultâncea das duas entradas é dada por,

$$Y(s) = Y_1(s) + Y_2(s) \tag{39}$$

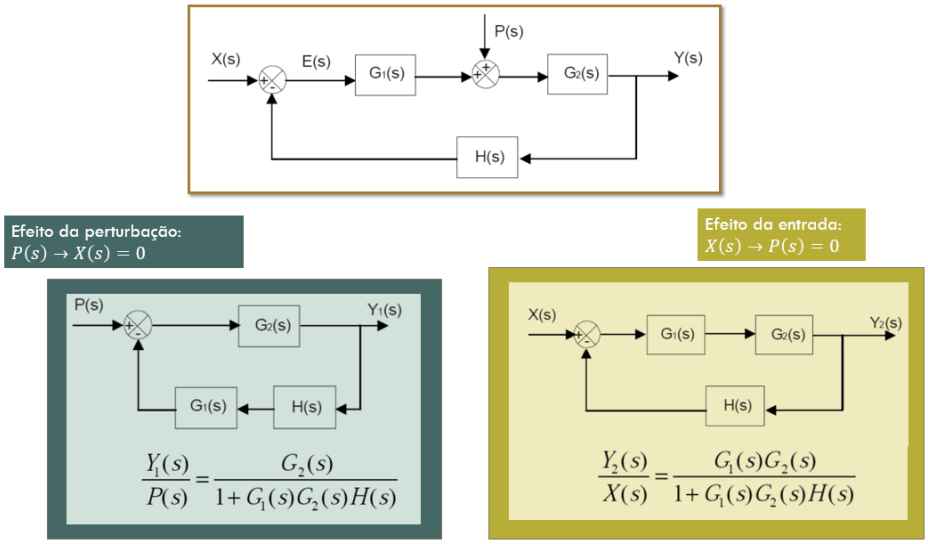


Figura 66: Sistema com perturbação.

13.3 SOLUÇÃO USANDO MATLAB

Os comandos do MatLab
`[num,den]=series(num1,den1,num2,den2)`,
`[num,den]=parallel(num1,den1,num2,den2)` e
`[num,den]=feedback(num1,den1,num2,den2)`, respectivamente, são utilizados para simplificar cada uma das situações ilustradas em Figura 67. São definidos,

$$G_1(s) = \frac{num1}{den1} \qquad G_2(s) = \frac{num2}{den2}$$

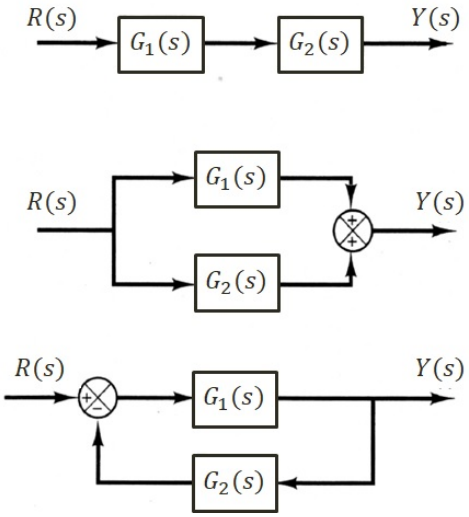


Figura 67: Sistema em série, paralelo e feedback negativo.

Particularmente, utiliza-se `[num,den]=feedback(num1,den1,num2,den2,+1)` quando a retroalimentação for positiva. Por exemplo, para Figura 68, tem-se o seguinte script do MatLab,

```
%%Malha fechada
% feedback positivo
num1=[1]; den1= [1 1];
num2=[1]; den2=[1 2];
sys1=tf(num1,den1); sys2=tf(num2,den2)
FeedPos = feedback(10*sys1,sys2,+1)
% feedback negativo
FeedNeg = feedback(10*sys1,sys2)
```

Veja que, ao contrário do exemplo anterior, primeiro criamos sistemas através do comando `tf` e depois simplificamos os blocos. As duas opções funcionam, você pode escolher a que preferir. Em geral, nós, particularmente, preferimos a segunda, pois organiza os blocos em sub-sistemas.

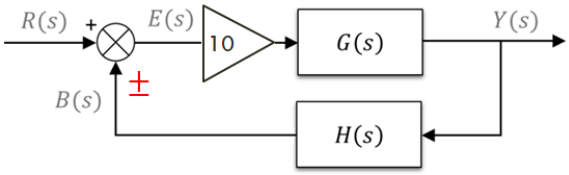


Figura 68: Exemplo de sistema com retroalimentação e uso do MatLab.

Verifique no exemplo da Figura 69 uma situação bastante útil de movimentação de realimentação e soma.

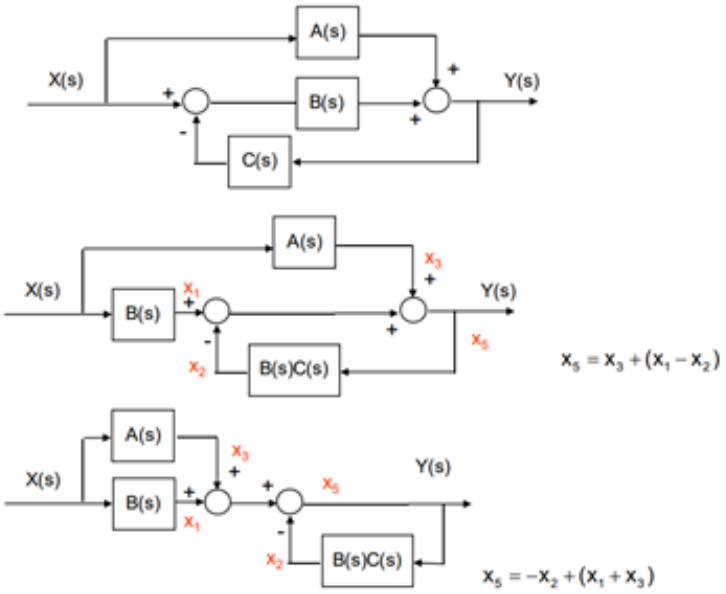


Figura 69: Fonte: https://fenix.tecnico.ulisboa.pt/downloadFile/3779572053485/Cap4%20-%20Diagrama-de-Blocos.pdf

Existem tabelas, em livros e apostilas da Internet, com várias simplificações dos diagramas de blocos. Por exemplo, a tabela a seguir foi retirada de [10], pág. 157.

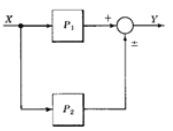
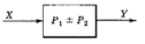
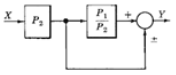
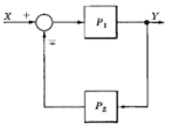
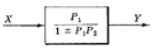
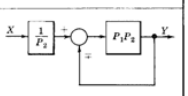
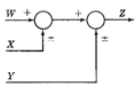
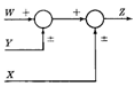
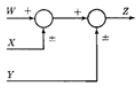
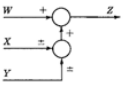
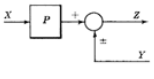
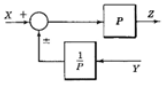
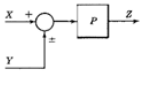
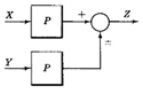
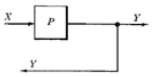
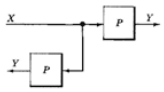
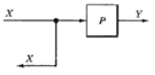
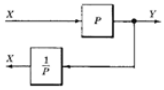
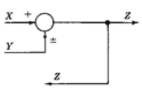
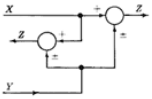
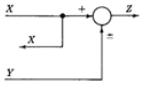
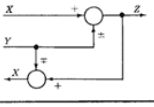
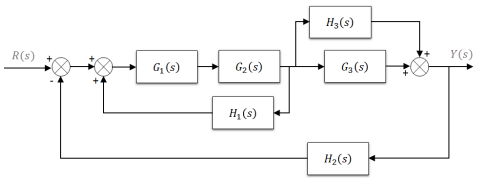
	Transformação	Equação	Diagrama em bloco	Diagrama em bloco equivalente
1	Combinação de blocos em cascata	$Y = (P_1P_2)X$		
2	Combinação de blocos em paralelo; ou eliminação de uma malha direta	$Y = P_1X \pm P_2X$		
3	Remoção de um bloco de um percurso direto	$Y = P_1X \pm P_2X$		
4	Eliminação de uma malha de realimentação	$Y = P_1(X \mp P_2Y)$		
5	Remoção de um bloco de uma malha de realimentação	$Y = P_1(X \mp P_2Y)$		
6a	Reorganizando os pontos de soma	$Z = W \pm X \pm Y$		
6b	Reorganizando os pontos de soma	$Z = W \pm X \pm Y$		
7	Movendo um ponto de soma à frente de um bloco	$Z = PX \pm Y$		
8	Movendo um ponto de soma para além de um bloco	$Z = P[X \pm Y]$		
9	Movendo um ponto de tomada à frente de um bloco	$Y = PX$		
10	Movendo um ponto de tomada para além de um bloco	$Y = PX$		
11	Movendo um ponto de tomada à frente de um ponto de soma	$Z = X \pm Y$		
12	Movendo um ponto de tomada para além de um ponto de soma	$Z = X \pm Y$		

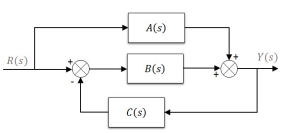
Figura 70: Tabela extraída de [10].

13.4 EXERCÍCIOS

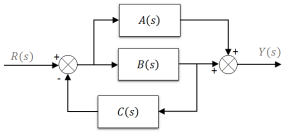
1. Simplifique os diagramas de blocos e confira sua resposta com MatLab.



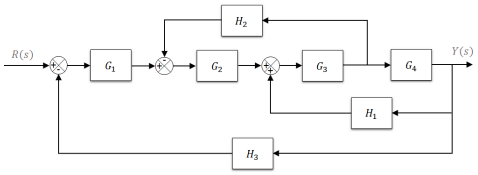
(a)



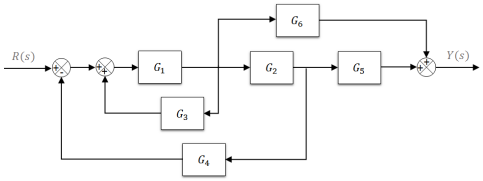
(b)



(c)

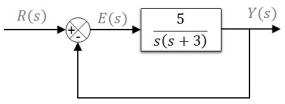


(d)

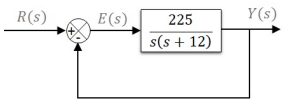


(e)

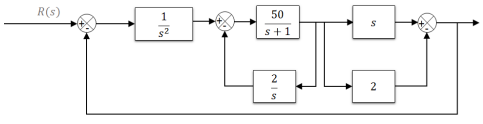
2. Resolva pelo MatLab.



(a)

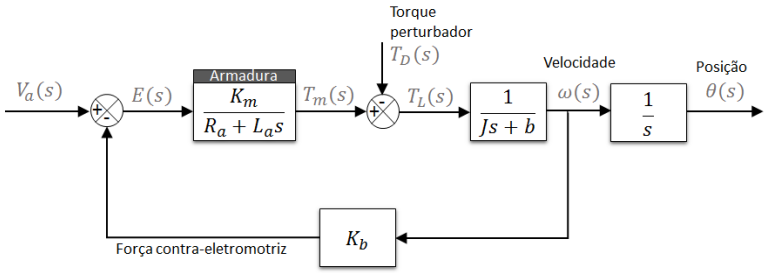


(b)

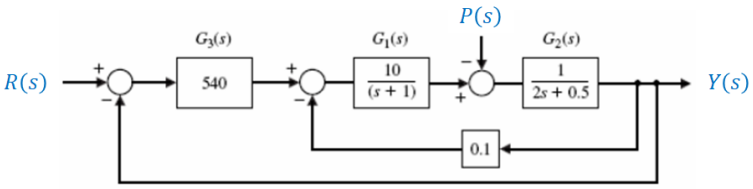


(c)

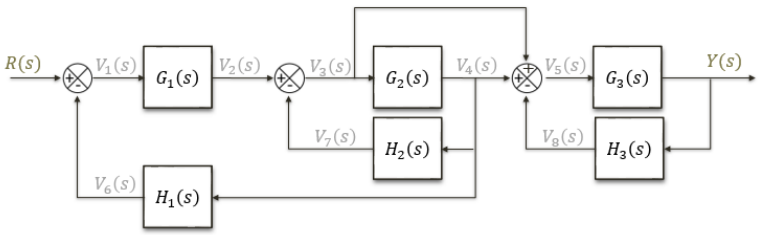
3. Simplifique os diagramas de blocos bastante complexos a seguir sem e com ajuda do MatLab.


Torque
perturbador
$T_D(s)$
Velocidade
Posição
$V_a(s)$
$E(s)$
Armadura
$\frac{K_m}{R_a + L_a s}$
$T_m(s)$
$T_L(s)$
$\frac{1}{Js + b}$
$\omega(s)$
$\frac{1}{s}$
$\theta(s)$
K_b
Força contra-eletromotriz

(a)


$P(s)$
$G_3(s)$
$G_1(s)$
$G_2(s)$
$R(s)$
540
$\frac{10}{(s+1)}$
$\frac{1}{2s+0.5}$
$Y(s)$
0.1

(b)


$R(s)$
$V_1(s)$
$G_1(s)$
$V_2(s)$
$V_3(s)$
$G_2(s)$
$V_4(s)$
$V_5(s)$
$G_3(s)$
$Y(s)$
$V_7(s)$
$H_2(s)$
$V_8(s)$
$H_3(s)$
$V_6(s)$
$H_1(s)$

(c)

14 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.

14.1 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 71). A janela principal do SIMULINK será exibida com as bibliotecas de blocos disponíveis para uso como mostra a Figura 72.

Figura 71: Ícone do SIMULINK.

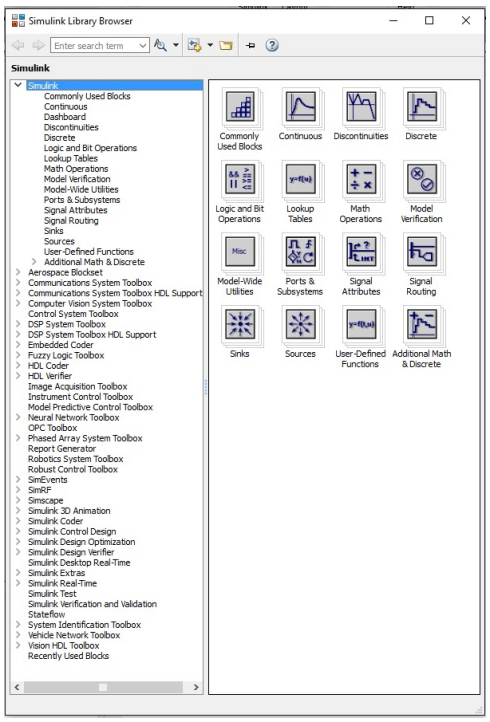


Figura 72: Biblioteca de blocos do SIMULINK.

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 73 a janela (untitled) aberta.

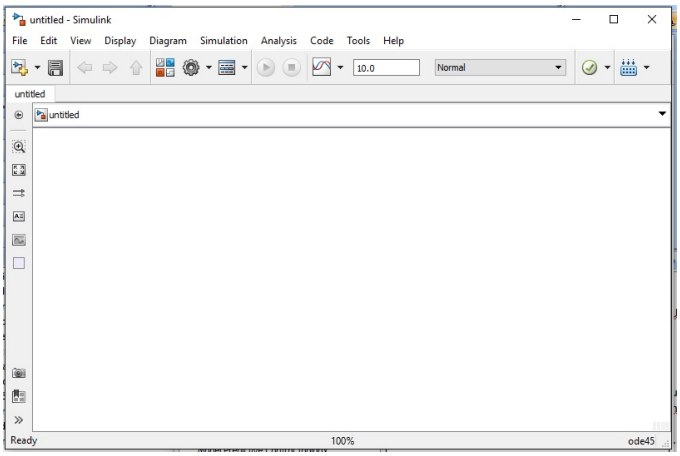


Figura 73: Área de trabalho.

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

14.2 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`.

14.2.1 *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 74 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 [17].

14.2.2 *Blocos*

O modelo do sistema contínuo está mostrado nos blocos da Figura 75. Verifique a opção de introdução de Equações de Estado ou Transformada de Laplace.

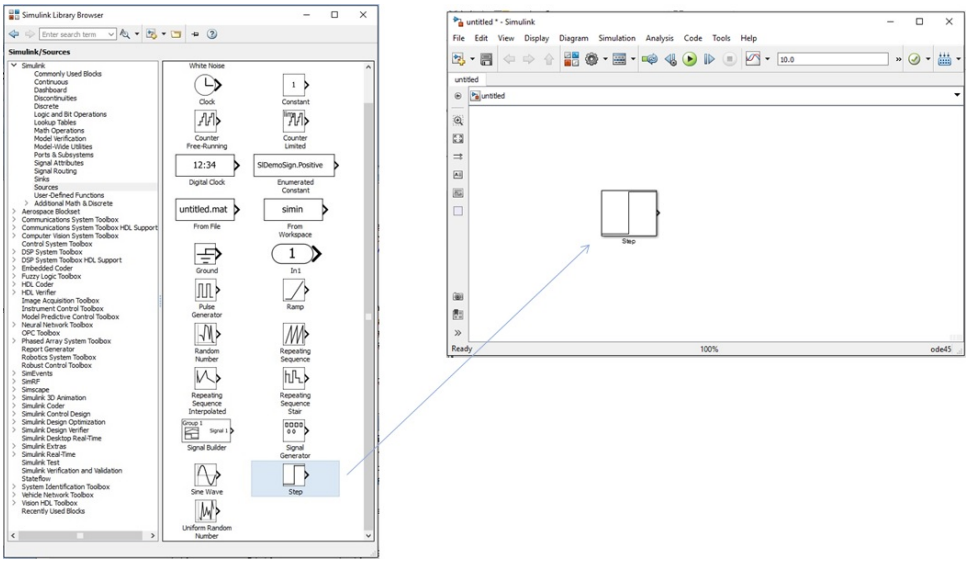


Figura 74: Geração de uma fonte no modelo.

14.2.3 *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.

14.3 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 77 a uma função degrau, conforme representado na Figura 78.

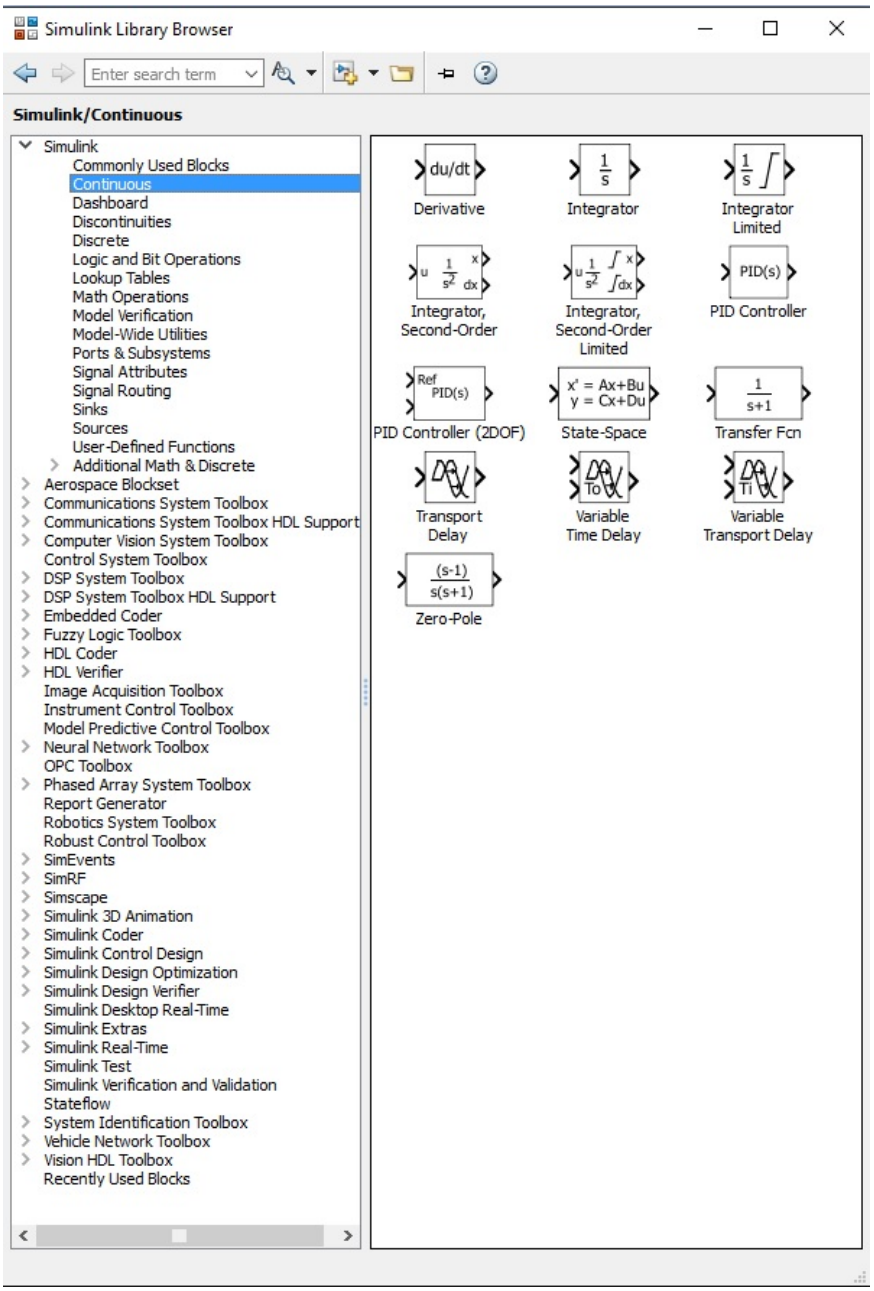


Figura 75: Blocos.

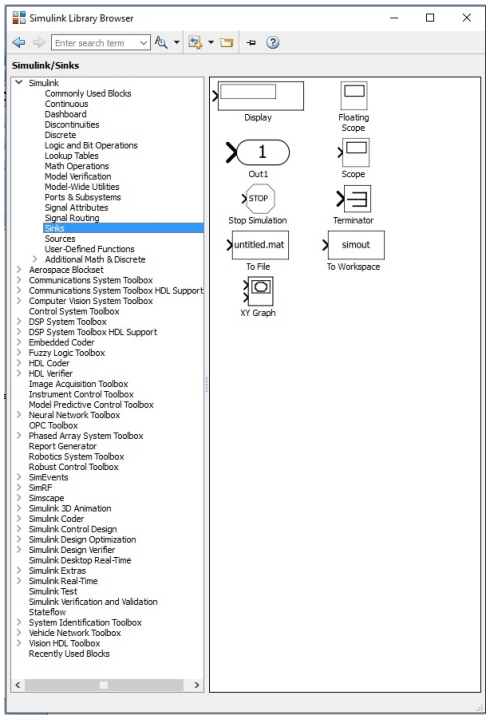


Figura 76: Geração do modo de saída do modelo.

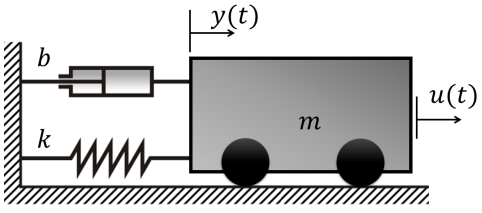


Figura 77: Sistema MMA.

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 8, são,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2/5 & -1/5 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1/5 \end{bmatrix} u(t) \tag{40}$$

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

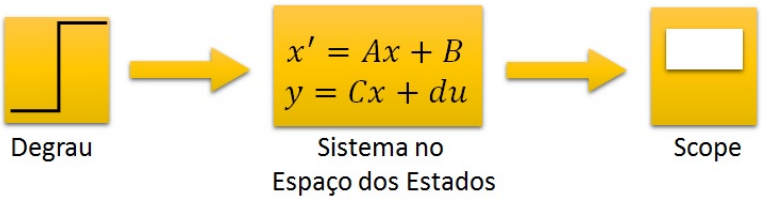


Figura 78: Modelo de sistema dinâmico no SIMULINK.

14.3.1 *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 74;

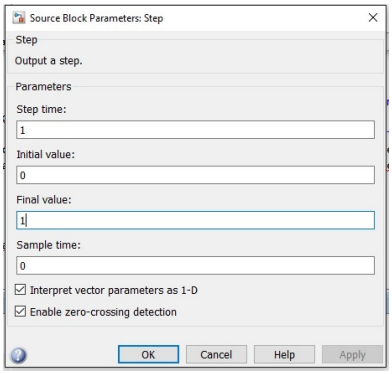


Figura 79: Setup da função `Step`.

   - Dê um duplo clique sobre o signal generator ou clique com o botão direito e selecione os parâmetros de sua função `Step` (Figura 79).

2. Insira os blocos do sistema modelado:
   - Qualquer bloco no simulink pode ser pesquisado na linha de comando (Figura 80). 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*).

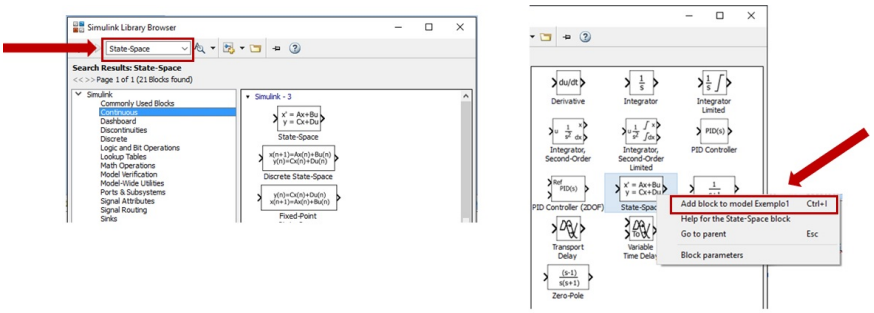


Figura 80: Como adicionar um bloco.

- 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 81).

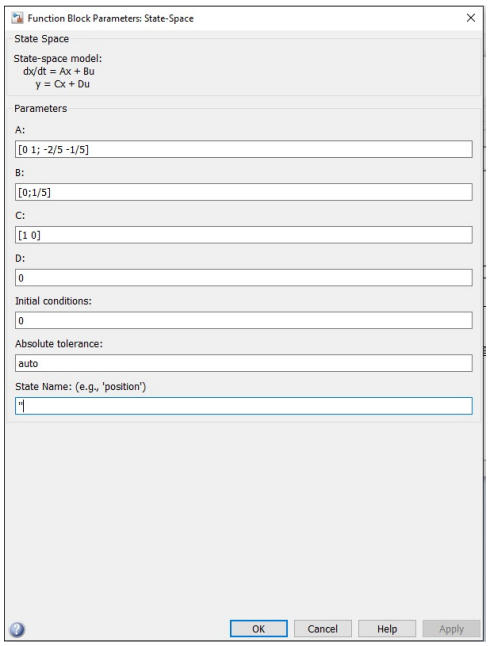


Figura 81: Propriedades conforme Equação 40.

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 82.
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

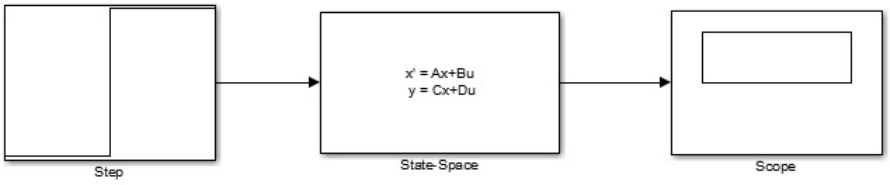


Figura 82: Modelo final da área de trabalho, de acordo com exemplo ilustrado na Figura 78.

e depois em `Model Simulation Parameter` e `Data Import/Export`, para acessar as mais importantes opções de simulação (veja Figura 83).

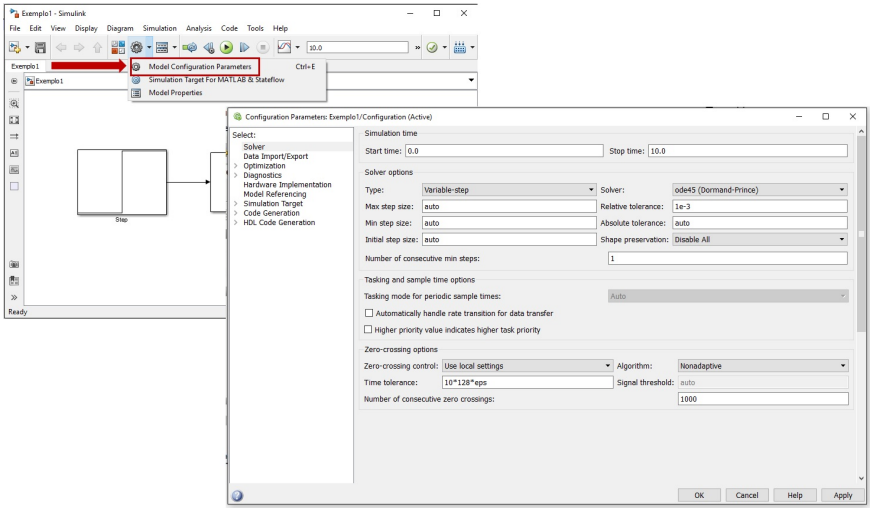


Figura 83: Parâmetros de simulaçao.

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 8) 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 84. 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 85.

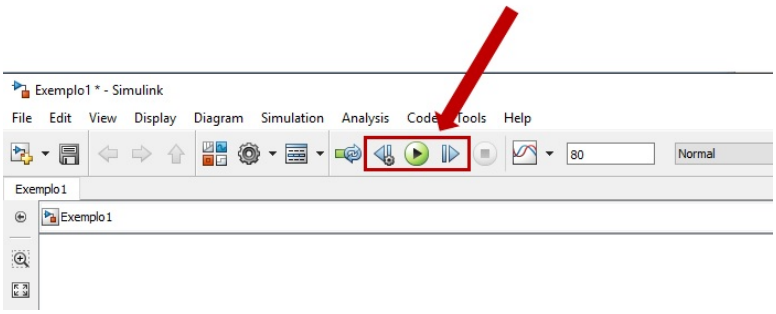


Figura 84: Para rodar o modelo.

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 85. 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

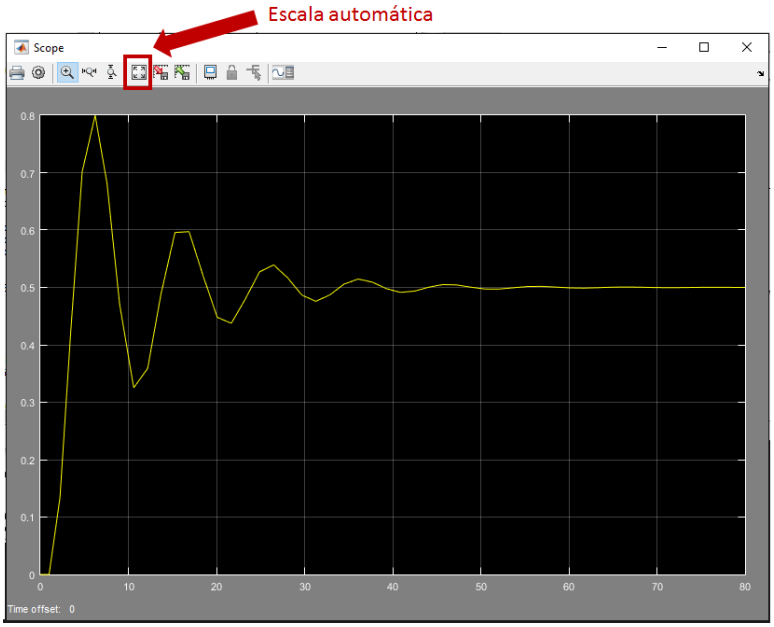


Figura 85: Para rodar o modelo.

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 77 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 parãmetros do bloco, Figura 86, com numerador $[1/5]$ e denominador $\left[11/52/5\right]$. A resposta do modelo deve ser idêntica àquela mostrada na Figura 85.

14.4 EXERCÍCIOS

1. Resolva o sistema do exemplo MMA ilustrado na Figura 77, agora sem entrada no sistema, apenas com deflexão inicial $x_o = 1m$. Dica: use a opção `Integrator` duas vezes para achar velocidade e deslocamento.

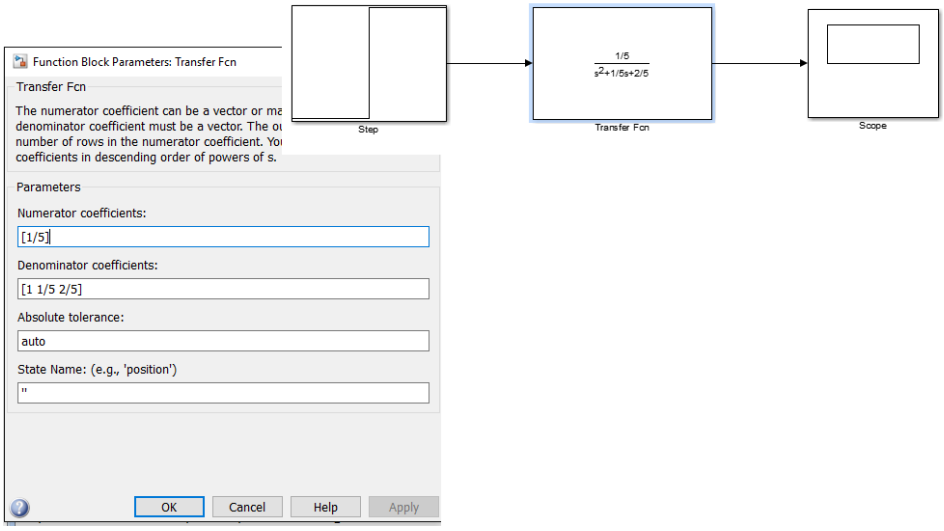


Figura 86: Modelo SIMULINK. Solução por Laplace.

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 $x(0) = -0,1m$ e $v(0) = 0m/s$ 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.

3. Dado o modelo da suspensão de duas massas conforme Figura 50, analisado no Capítulo 11, desenvolva aqui um modelo SIMULINK para obter as plotagens de $x_1(t)$ e $x_2(t)$, sendo $y(t)$ uma função degrau unitária e condições iniciais nulas. Dados: $m_1 = 250kg$, $m_2 = 40kg$, $k_1 = 15kN.m$, $k_2 = 150kN.m$ e $c_1 = 1917N.s/m$.

4.

5. O sistema a ser simulado é um tanque de nível, Figura 87. Uma corrente de entrada alimenta o tanque, cujo valor da vazão pode ser ajustado. Uma corrente de saída tem a sua vazão definida pela altura h de líquido no tanque, através da relação $F = k\sqrt{h}$. Mostre o comportamento da altura h com o tempo. Os parâmetros são: $k = 8m^{5/2}/min$, $A = 0,3m^2$. A altura inicial $h(0) = 3m$.

   A equação de estado é dada por,

$$\frac{dh}{dt} = \frac{1}{A}\left(F_o - k\sqrt{h}\right), \qquad h(0) = 3m$$

Considera-se que existam dois níveis limites para o nível (alarmes) que devem aparecer no gráfico da resposta do simulador:

- $h_{amax} = 4m$ (alarme nível alto)
- $h_{amin} = 2m$ (alarme nível baixo)

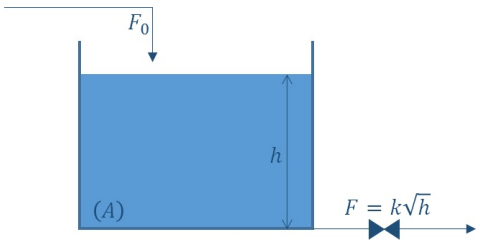


Figura 87: Reservatório.

A vazão de entrada permanece igual a $F_{os} = 13,86m^3/min$ até um tempo de simulação de $t_{deg} = 0,2min$. Neste momento, uma perturbação degrau em $F_o(t)$ aumentará esse valor para $F_{odeg} = F_{os} + \Delta F_o$. Ambas as vazões devem estar presentes em um mesmo gráfico.

A simulação ocorrerá desde o instante $t_0 = 0$ até $t_f = 1min$.

Monte o problema no Simulink. Localize o botão `GED (Graphical Editor)` na janela gráfica.

Parte VI

APOIO E REFERÊNCIAS BIBLIOGRÁFICAS

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

BIBLIOGRAFIA

[1] K. J. Aström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. 2016. URL http://www.cds.caltech.edu/~murray/amwiki/index.php?title=Main_Page.

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

[3] D. K. Chaturvedi. *Modeling and Simulation od systems using MATLAB and Simulink*. CRC Press, 2010.

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

[5] DOU. Inovar-auto, sobre programa de incentivo 'a inovação tecnológica e adensamento da cadeia produtiva de veículos automotores, 2012.

[6] Ramin S. Esfandiari and Bei Lu. *Modeling and Analysis of Dynamic Systems,*. 2014.

[7] Luiz Carlos Felício. *Modelagem da dinâmica de sistemas e estudo da resposta*. RiMa Editora, 2 edition, 2010. ISBN 978-85-7656-169-9. URL http://www2.eesc.usp.br/labdin/luiz/Modelagem%20da%20Dinamica%20de%20Sistemas%20e%20Estudo%20da%20Resposta.pdf.

[8] Avelino Alves Filho. *Elementos Finitos. A Base da Tecnologia CAE*. Editora Érica/Saraiva, 2000.

[9] A. Horridge. The anti-intuitive visual system of the honey bee. horridge a. 63(2):20–35, 2012.

[10] J. J. DiStefano III, A. R. Stubberud, and I. J. Williams. *Sistemas de controle*. 2 edition, 2014.

[11] D. E. Malen. *Fundamentals of automobile body structure design*. SAE International, 2011.

[12] Javad Marzbanrad and Mostafa Pahlavani. Paremaeter determination of a vehicle 5-dof model to simulate occupant deceleration in a frontal crash. *International Journal of Mechanical and Mechatronics Engineering*, 5(7):1284–1289, 2011.

[13] MatLab. Magic squares, 2017. URL https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/exm/chapters/magic.pdf.

[14] Sunday M. Ofochebe, Chigbogu G. Ozoegwu, and Samuel O. Enibe. Performance evaluation of vehicle front structure in crash energy management using lumped mass spring system. *Advanced Modeling and Simulation in Engineering Sciences*, 2(2), 2015.

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

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

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

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

[19] A. R. Romeiro. Desenvolvimento econômico e a questão ambiental: algumas considerações. *Análise Econômica*, 16:141–152, 1991.

[20] W. M. Siebert. *Circuits, Signals and Systems*. MIT Press, 1986. URL https://ocw.mit.edu/courses/mechanical-engineering/2-017j-design-of-electromechanical-robotic-systems-fall-2009/assignments/MIT2_017JF09_p01.pdf.

[21] Mandyam V. Srinivasan. Honeybees as a model for the study of visually guided flight, navigation, and biologically inspired robotics. *Physiol Rev*, 91: 413–460, 2011.