MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

APOSTILA DA DISCIPLINA DE INFORMÁTICA APLICADA

PROFESSORA: MARIA LUISA PERDIGÃO DIZ RAMOS

Belo Horizonte
2020

SUMÁRIO

1 – Usando o Microsoft Office

1.1 Word

1.1.1 – Formatação no Word

EXERCÍCIO 1:

Digitar o texto e formatá-lo conforme apresentação abaixo.

Caminhar é Preciso. **De Eduardo Machado.**

Que é a vida senão uma grande caminhada?

Caminhada, afinal, é uma expressão que se aplica a todas as nossas experiências. Caminhamos no conhecimento, no desenvolvimento físico, no relacionamento humano. Caminhamos na busca de nossos sonhos e projetos. Vivemos uma caminhada que começa dentro de nós mesmos, nas estradas e trilhas da nossa espiritualidade e se espalha a nossa volta.

Há gente que tem pressa demais. Quer chegar tão rápido que mal percebe quanto o caminho é bonito. A paisagem é confortadora e os companheiros agradáveis e amigos.

É preciso encontrar o ritmo ideal.

É preciso perceber, que em determinados trechos do caminho, será preciso andar mais rápido e em outros, caminhar bem lentamente.

Pode ser preciso parar para recuperar forças perdidas. Podemos precisar caminhar certos trechos apoiados em alguém. Podemos precisar ajudar algum companheiro de caminhada.

É preciso ter olhos bem abertos para aprender bem o caminho. Quem sabe, algum dia, teremos que ser guias de outros caminhantes... Por planícies ou estradas íngremes e acidentadas as margens de algum rio ou subindo uma montanha.

Caminhar é preciso, mesmo se houver riscos. É claro que é mais cômodo e seguro ficar sentado a beira do caminho. Mas ficar sentado é estagnar, é não crescer, é perder o caminho e se acomodar.

Quem caminha quer sempre aprender mais. Seu horizonte é o seu sonho, o seu ideal. Aceita desafios. Faz amigos e companheiros de caminhada.

Sabe que o rumo é a felicidade e que caminhamos com os pés no chão e o coração na eternidade.

Bem vindos ao Cefet-MG e ao Curso de Eletrotécnica.

EXERCÍCIO 2:

Digitar o texto e formatá-lo conforme apresentação abaixo.

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Prezado Técnico,

A coordenação de estágio do Cefet-MG possui um Serviço de Encaminhamento Profissional (SEP) que tem como objetivo principal encaminhar os alunos egressos para o mercado de trabalho.

Se for do seu interesse, enquanto ex-aluno da instituição (egresso) você poderá fazer parte de um banco de dados, podendo ter seu nome encaminhado para as empresas que assim o solicitarem, ao longo de um ano, para tornar possível sua colocação profissional.

Para isso preencha o currículo anexo e envie-o a coordenação de estágio, sala 223, campus I do Cefet-MG.

Atenciosamente,

_____

Coordenador de Estágio

1.1.2 – Mala Direta

É possível usar a mala direta quando deseja criar um conjunto de documentos, como uma carta modelo que é enviada a muitos clientes ou uma folha de etiquetas de endereço. Cada carta ou etiqueta tem o mesmo tipo de informações, no entanto o conteúdo é exclusivo. Por exemplo, nas cartas aos seus clientes, cada carta pode ser personalizada para abordar cada cliente pelo nome. As informações exclusivas em cada carta ou etiqueta provêm de entradas em uma fonte de dados.

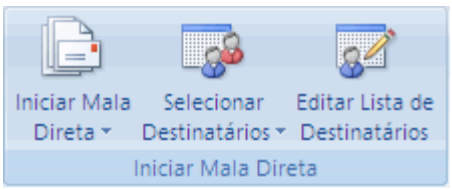
O processo de mala direta inclui as seguintes etapas gerais:

1. Definir o documento principal. O documento principal contém o texto e os gráficos que são os mesmos para cada versão do documento mesclado. Por exemplo, o endereço de retorno ou a saudação em uma carta modelo.
2. Conectar o documento a uma fonte de dados. Uma fonte de dados é um arquivo que contém as informações a serem mescladas em um documento. Por exemplo, os nomes e os endereços dos destinatários de uma carta.
3. Refinar a lista de destinatários ou os itens. O Microsoft Office Word gera uma cópia do documento principal para cada item, ou registro, no seu arquivo de dados. Se o seu arquivo de dados for uma lista de correspondência, esses itens serão provavelmente destinatários da sua correspondência. Se você quiser gerar cópias apenas para determinados itens no seu arquivo de dados, poderá escolher quais itens (registros) incluir.
4. Adicionar espaços reservados, chamados campos de mala direta, ao documento. Ao realizar a mala direta, os campos da mala direta são preenchidos com informações de seu arquivo de dados.
5. Visualizar e completar a mesclagem. É possível visualizar cada cópia do documento antes de imprimir todo o conjunto.

Você pode usar comandos na guia **Correspondências** para executar uma mala direta.

Definir o documento principal

1. Inicie o Word. Um documento em branco.
2. Na guia **Correspondências**, no grupo **Iniciar Mala Direta**, clique em **Iniciar Mala Direta**.

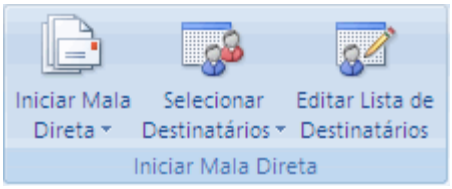


3. Clique no tipo de documento que deseja criar.

   Por exemplo, é possível criar:

- **Um conjunto de envelopes** O endereço de retorno é o mesmo em todos os envelopes, mas o endereço de destino é exclusivo em cada um. Clique em **Envelopes** e, em seguida, especifique suas preferências para o tamanho do envelope e a formatação do texto na guia **Opções de Envelope** da caixa de diálogo **Opções de Envelope**.
- **Um conjunto de etiquetas de endereço** Cada etiqueta mostra o nome e o endereço da pessoa, mas o nome e o endereço em cada etiqueta é exclusivo. Clique em **Etiquetas** e, em seguida, especifique suas preferências para o tipo de etiqueta na caixa de diálogo **Opções de Etiqueta**.
- **Um conjunto de cartas modelo ou emails** O conteúdo básico é o mesmo em todas as cartas ou mensagens, mas cada um contém informações específicas ao destinatário individual, como nome, endereço ou outra informação. Clique em **Cartas** ou **Emails** para criar esses tipos de documentos.
- **Um catálogo ou diretório** O mesmo tipo de informação, como nome e descrição, é mostrado para cada item, mas o nome e a descrição em cada item é exclusivo. Clique em **Diretório** para criar esse tipo de documento.

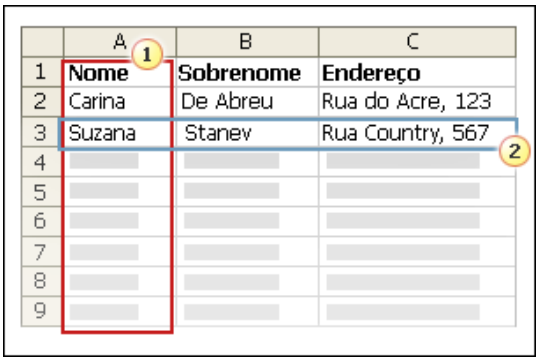
4. Na guia Correspondências, no grupo **Iniciar Mala Direta**, clique em **Selecionar Destinatários**.



   Você poderá usar uma lista já existente ou criar uma nova lista. Para criar uma nova lista clicar em “Digitar Nova Lista”. A partir desse momento você poderá “Personalizar Colunas”, isto é, adicionando, renomeando ou excluindo as já existentes. Após criar as colunas clicar em OK e digitar os dados dos destinatários.

5. Após conectar o seu documento principal a um arquivo de dados, você estará pronto para digitar o texto do documento e adicionar espaços reservados que indicam onde as informações exclusivas aparecerão em cada cópia do documento.

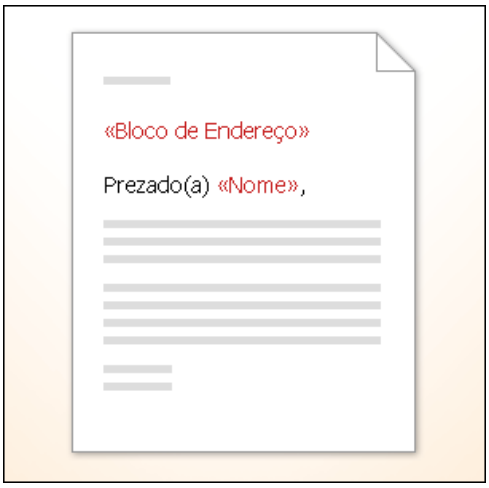
   Os espaços reservados, como endereço e saudação, são chamados de campos de mala direta. Os campos no Word correspondem aos títulos da coluna no arquivo de dados selecionado.



1 As colunas em um arquivo de dados representam as categorias de informações. Os campos que você adiciona ao documento principal são espaços reservados para essas categorias.

2 As linhas em um arquivo de dados representam os registros de informações. O Word gera uma cópia do documento principal para cada registro ao realizar uma mala direta.

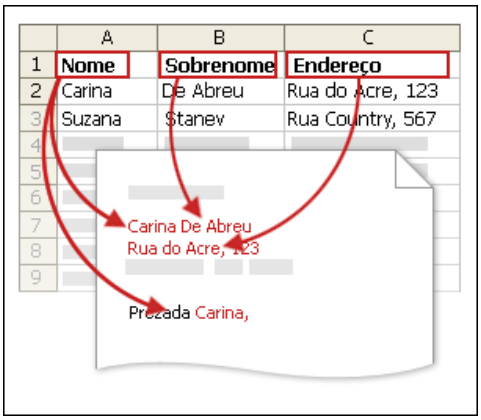
Ao colocar um campo no seu documento principal (ícone “Inserir Campo de Mesclagem”), você indica que deseja uma determinada categoria de informação, como nome ou endereço, para aparecer nesse local.



OBSERVAÇÃO Ao inserir um campo de mala direta no documento principal, o nome do campo é sempre cercado por sinais de divisas (« »). Esses sinais de divisas não aparecem nos documentos mesclados. Eles apenas ajudam a distinguir os campos no documento principal do texto normal.

O que ocorre ao mesclar

Ao mesclar as informações da primeira linha no arquivo de dados, substitui os campos no seu documento principal para criar o primeiro documento mesclado. As informações da segunda linha no arquivo de dados substituem os campos para criar o segundo documento mesclado e assim por diante.



6. Para visualizar o resultado dos dados mesclados basta clicar no ícone “Visualizar Resultados”. Ao lado desse ícone existem botões onde você poderá visualizar desde o primeiro destinatário até o último.

EXERCÍCIO:

1 – Criar a mala direta da página seguinte com a orientação do professor.

2 – Criar a segunda mala direta conforme foi ensinado. Essa mala direta deverá conter pelo menos três alunos cadastrados.

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

OF. Setor de Estágio / «**NUM**» «**DATA**»
Da: Coordenação de Estágio
À «**EMPRESA**»

Prezado Senhor,

De acordo com o regulamento deste Centro, o aluno durante o curso técnico ou após sua conclusão, deverá cumprir o Estágio Supervisionado Obrigatório, com carga horária mínima de 480 horas de atividades em área relacionada com seu curso, podendo ser prorrogado por até dois anos. Esta disciplina poderá ser realizada sob forma de estágio ou emprego.

Apraz-nos, pois, apresentar a V.S.ª, o(a) aluno(a) «**ALUNO**» regularmente matriculado no Curso Técnico de «**CURSO**» a uma oportunidade de estágio/emprego nesta conceituada empresa.

Nos termos do artigo 7º parágrafo 3º da Resolução CNE/CEB 1/2004, Diário Oficial da União, de 04 de fevereiro de 2004 – Seção 1, pág. 21, o estágio não pode exceder a jornada semanal de 40 horas.

Certos da atenção de V.S.ª, traduzimos o nosso sincero agradecimento e reafirmamos nossos protestos de apreço e consideração.

Atenciosamente,

_____

Coordenador de Estágio

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MG
AV. AMAZONAS, 5253 – NOVA SUIÇA – BH / CEP 30480-000
FONE: 3319-7086
CNPJ 17.220.203/0001-96
INSCRIÇÃO ESTADUAL - ISENTO

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

DECLARAÇÃO

Declaramos, para os devidos fins, que **<<NOME_DO_ALUNO>>,** do Curso técnico de **<<NOME_DO_CURSO>>,** está inscrito(a) no Exercício Orientado da Profissão. O(a) aluno(a) tem contrato de estágio assinado para o período de **<<DATA_INÍCIO>> a <<DATA_FIM>>,** na empresa **<<NOME_DA_EMPRESA>>**.

Dados fornecidos pelo(a) aluno(a):

ENDEREÇO: <<ENDEREÇO_DO_ALUNO>>

BAIRRO: <<BAIRRO>>

CIDADE: <<CIDADE>>

CEP: <<CEP>>

TELEFONE: <<TELEFONE>>

DATA DE NASCIMENTO: <<NASCIMENTO>>

Belo Horizonte, 01 de janeiro de 2019.

_____

Coordenador de Estágio

1.1.3 – Índice Analítico

Você cria um índice analítico escolhendo os estilos de título, por exemplo, Título 1, Título 2 e Título 3 , que deseja incluir no índice analítico. O Microsoft Office Word procura títulos que correspondam ao estilo que escolheu, formata e recua o texto da entrada de acordo com o estilo do título e insere o índice analítico no documento.

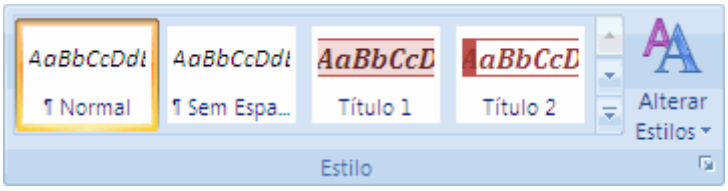
O Microsoft Office Word fornece uma galeria com vários estilos de sumário para você escolher. Marque as entradas do sumário e clique no estilo de sumário que deseja usar a partir das opções da galeria. O Office Word cria automaticamente o sumário a partir dos títulos marcados.

Marcar entradas para um índice analítico

A maneira mais fácil de criar um índice analítico é usar os **estilos de título** internos. Também é possível criar um índice analítico com base nos estilos personalizados que você aplicou. Ou é possível atribuir os níveis do índice analítico às entradas de texto individuais.

Marcar entradas usando estilos de título internos

1. Selecione o título no qual deseja aplicar um estilo de título.
2. Na guia **Página Inicial**, no grupo **Estilos**, clique no estilo desejado.

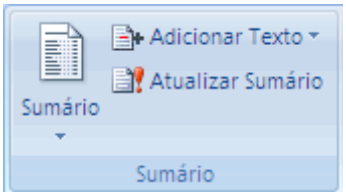


Por exemplo, se você selecionou o texto no qual deseja aplicar o estilo de título principal, clique no estilo chamado **Título 1** na Galeria de estilos rápidos.

Marcar entradas de texto individuais

Se quiser que o índice analítico inclua o texto que não está formatado como título, você poderá usar esse procedimento para marcar entradas de texto individuais.

1. Selecione o texto que deseja inclui no seu índice analítico.
2. Na guia **Referências**, no grupo **Sumário**, clique em **Adicionar Texto**.

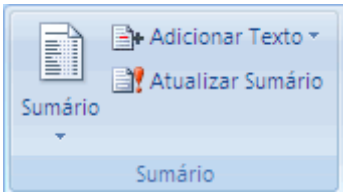


3. Clique no nível que deseja marcar sua seleção, como **Nível 1** para exibição do nível principal no sumário.
4. Repita as etapas 1 a 3 até que tenha rotulado todo o texto que deseja que apareça no índice analítico.

Criar um índice analítico

Após marcar as entradas do seu índice analítico, você está pronto para criá-lo.

1. Clique no local que deseja inserir o índice analítico, normalmente no início de um documento.
2. Na guia **Referências**, no grupo **Sumário**, clique em **Sumário** e, em seguida, clique no estilo de sumário desejado.



OBSERVAÇÃO Para obter mais opções, clique em **Inserir Índice Analítico** para abrir a caixa de diálogo **Índice analítico**.

Criar um índice analítico a partir de estilos personalizados que você aplicou

Use esse procedimento se você já aplicou estilos personalizados aos seus títulos. É possível escolher as configurações de estilo as quais deseja que o Word use ao criar o índice analítico.

1. Clique no local em que deseja inserir o índice analítico.
2. Na guia **Referências**, no grupo **Sumário**, clique em **Sumário** e em **Inserir Sumário**.
3. Clique em **Opções**.
4. Em **Estilos disponíveis**, localize o estilo que aplicou aos títulos no seu documento.
5. Em **Nível do índice**, ao lado do nome do estilo, digite um número de 1 a 9 para indicar o nível o qual deseja que o estilo do título represente.

   OBSERVAÇÃO Se quiser usar apenas estilos personalizados, exclua os números do Nível do índice para os estilos internos, como Título 1.
6. Repita as etapas 4 e 5 para cada estilo de título que deseja incluir no índice analítico.
7. Clique em **OK**.
8. Escolha um índice analítico para se adequar ao tipo de documento:

- **Documento impresso** Se você estiver criando um documento que os leitores irão ler em uma página impressa, crie um índice analítico no qual cada entrada lista o título

e o número da página no qual o título aparece. Os leitores podem ir até a página que quiserem.

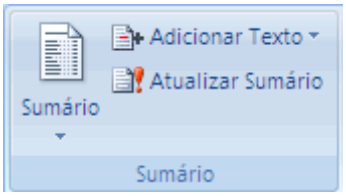
- **Documento online** Para um documento em que os leitores irão ler online no Word, é possível formatar as entradas no índice analítico como hiperlinks, de modo que os leitores possam ir a um título clicando em sua entrada no índice analítico.

9. Para usar um dos designes disponíveis, clique em um deles na caixa **Formatos**.
10. Selecione quaisquer outras opções de índice analítico que desejar.

Atualizar o índice analítico

Se você adicionou ou removeu títulos ou outras entradas do índice analítico no seu documento, é possível atualizar rapidamente o índice analítico.

1. Na guia **Referências**, no grupo **Sumário**, clique em **Atualizar Sumário**.



2. Clique em **Atualizar apenas os números de página** ou **Atualizar o índice inteiro**.

1.1.3.1 – Quebra de Seção

Para realizar a quebra de seção os passos a serem seguidos são:
Layout da página → Quebras → Quebras de Seção / Próxima página.
Inserir → Número de Página → Formatar Número de Página → Iniciar em.

EXERCÍCIO

Criar o índice analítico abaixo (na página 1) após digitar cada título e subtítulo nas páginas indicadas. Usar quebra de página para deixar cada título em uma única página e usar quebra de seção (próxima página) após as páginas 05 (pula para a página 28) e 30 (pula para página 59).

SUMÁRIO

1.1.4 – Funções de Cálculo em Tabelas no Word

A tabela abaixo é usada como exemplo para explicação das funções citadas a seguir:

700,0987	800	100	100 =MIN(LEFT)	800 =MAX (LEFT)
200,9999	0 =AND(A1>700;A2>300)		700 =INT(A1)	3 =PRODUCT(1;3)
500,7777	467,2921 =AVERAGE(A1:A3)		100 =MOD(C1;B1)	1 =TRUE
1401,8763 =SUM(ABOVE)	60 =ABS(-60)		0 =DEFINED(A1/0)	1 =SIGN(A1)
4 =COUNT(ABOVE)	700,1 = ROUND(A1;1)		0 =FALSE	0 =IF(A1>800;SUM(LEFT); MIN(A1:A6))
0 =OR(A1<700; A2>300)	0 =NOT(SUM(A1:A2)<1000)			

As funções de cálculo em tabelas no Word se encontram em “Ferramentas de Tabelas”, “Layout”, “Dados” e “Fórmula”. Para acessar esses Menus é preciso primeiro criar uma tabela e clicar na célula onde deseja inserir a fórmula.

Dicas: O campo **= Formula** pode usar valores retornados pelas funções a seguir. As funções com parênteses vazios podem aceitar qualquer número de argumentos separados por vírgulas (,) ou ponto-e-vírgulas (;), contanto que você use o separador de lista definido como parte das configurações regionais no Painel de controle do Microsoft Windows. Os argumentos podem ser números, fórmulas ou nomes de indicadores. As funções AVERAGE(), COUNT(), MAX(), MIN(), PRODUCT() e SUM() podem aceitar referências a células de tabela como argumentos.

Palavras que podem ser usadas como parâmetros de funções: above (acima), below (abaixo), right (direita) e left (esquerda).

Para indicar intervalos de células utiliza-se “:”, e para indicar lista de células utiliza-se “;”.

Operadores no Word: >, <, >=, <=, =, <>, +, -, /, *.

ATENÇÃO!

Para que o Word reconheça uma função é necessário por o sinal de igual (=) antes de qualquer comando.

As funções de cálculo em tabelas existentes no Word são:

ABS(*x*) - Retorna o valor positivo de um número ou fórmula, independentemente de seu valor real positivo ou negativo.

AND(x;y) - Retorna o valor 1 se as expressões lógicas *x* e *y* forem ambas verdadeiras ou o valor 0 (zero) se uma das expressões for falsa.

AVERAGE( ) - Retorna a média de uma lista de valores.

COUNT( ) - Retorna o número de itens de uma lista.

DEFINED(*x*) - Retorna o valor 1 (verdadeiro) se a expressão *x* for válida ou o valor 0 (falso) se não for possível calculá-la (se for divisão por 0 retorna “!divisão por zero”).

FALSE - Retorna 0 (zero).

IF(EXPRESSÃO; VERDADEIRO; FALSO) – Retorna o 2º parâmetro se expressão for verdadeira; senão retorna o 3º parâmetro.

INT(x) - Retorna os números à esquerda da casa decimal no valor ou fórmula *x*.

MIN( ) - Retorna o menor valor de uma lista.

MAX( ) - Retorna o maior valor de uma lista.

MOD(x;y) - Retorna o resto que resulta da divisão do valor *x* pelo valor *y* um número inteiro de vezes.

NOT(*x*) - Retorna o valor 0 (zero) (falso) se a expressão lógica *x* for verdadeira ou o valor 1 (verdadeiro) se a expressão for falsa.

OR(*x;y*) - Retorna o valor 1 (verdadeiro) se uma das expressões lógicas *x* e *y*, ou as duas, forem verdadeiras, ou o valor 0 (zero) (falso) se as duas expressões forem falsas.

PRODUCT( ) - Retorna o resultado da multiplicação de uma lista de valores. Por exemplo, a função **{** = PRODUCT (1;3;7;9) **}** retorna o valor 189.

ROUND(x;y) - Retorna o valor de *x* arredondado para o número especificado de casas decimais *y*; *x* pode ser um número ou o resultado de uma fórmula.

SIGN(*x*) - Retorna o valor 1 se *x* for um valor positivo ou o valor –1 se x for um valor negativo ou 0 se *x* for 0.

SUM( ) - Retorna a soma de uma lista de valores ou fórmulas.

TRUE - Retorna o valor 1.

EXERCÍCIO

1. O trabalho será realizado em dupla, em sala de aula, durante duas semanas.
2. O tema será definido pelo professor.
3. A criatividade será avaliada.
4. O trabalho deverá contemplar e abordar os seguintes conteúdos trabalhados na disciplina de Laboratório de Informática Aplicada:
    a. Formatação, segundo Anexo A.
    b. Índice Analítico automático contendo, pelo menos, títulos 1 e 2.
    c. Elaboração de uma Mala Direta, com pelo menos 5 atributos, associada ao conteúdo trabalhado. O trabalho tem que conter a Mala Direta original e pelo menos três Malas Diretas preenchidas. Para anexar as Malas Diretas preenchidas no trabalho é só copiar as malas do arquivo em que elas foram geradas.
    d. Criar uma tabela associada ao conteúdo trabalhado. Deverão ser apresentadas na tabela fórmulas usando as funções estudadas do Word. Pelo menos cinco fórmulas deverão ser elaboradas, sendo que pelo menos duas delas deverão ser compostas. As fórmulas compostas deverão conter pelo menos três funções. Todas as fórmulas deverão ser digitadas logo abaixo da tabela, indicando a célula correspondente, ou dentro da própria célula, logo abaixo do resultado.
    e. O trabalho deverá ter no mínimo 12 páginas incluindo capa, sumário, entre outros.
    f. O trabalho deverá conter figuras.

1.2 Excel

1.2.1. Introdução

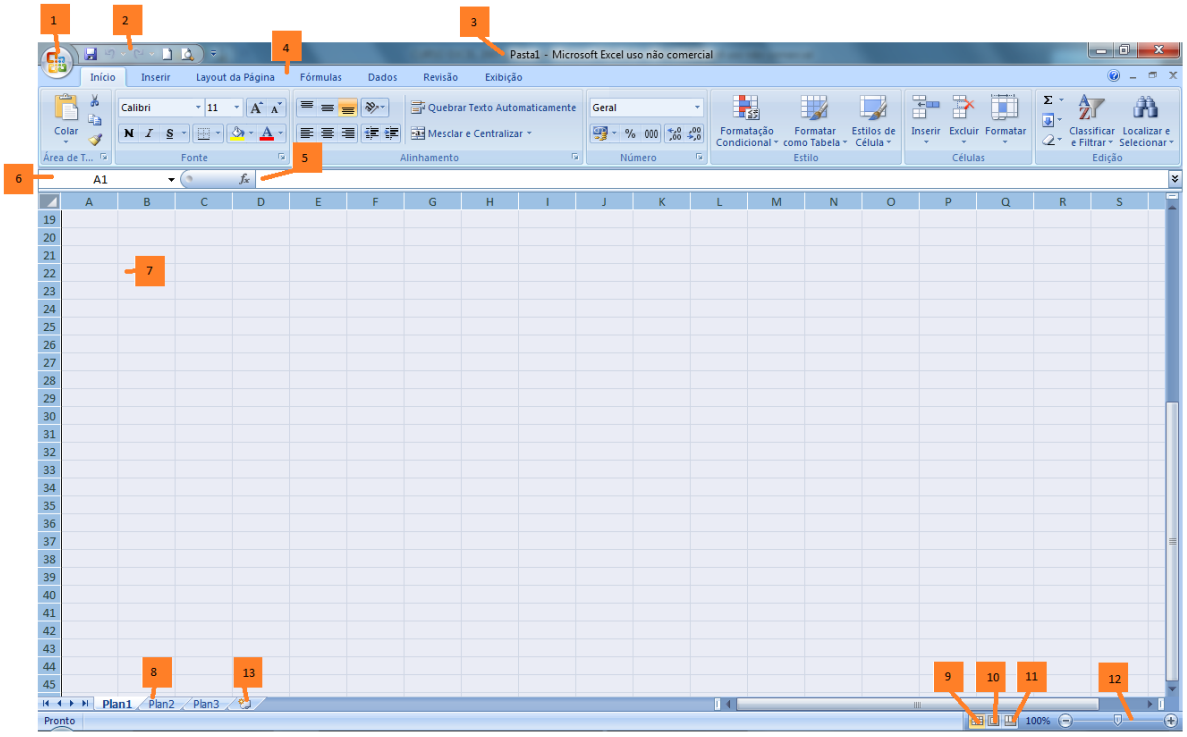
O Excel faz parte do pacote de programas do Microsoft Office. É um programa dedicado a criação de planilhas de cálculos, além de fornecer gráficos, função de banco de dados e outros. Nesta edição possui um novo layout que procura facilitar a localização de seus comandos. Contudo, se essa nova interface é mais fácil de usar para novos usuários, ela requer alguma habituação para aqueles que já estavam acostumados com as versões anteriores.

1.2.2 Layout do Excel

O Excel teve uma grande alteração de seu layout com relação à versão anterior, (2003), tornando-se mais amigável e fácil de visualizar suas funções. Porém, algumas delas se encontram organizadas de forma diferente da versão menos recente, tornando um pouco “difícil” encontrá-las para os usuários acostumados com a versão 2003. Em vista disso, nas figuras abaixo, demonstraremos os novos menus para tornar mais fácil a localização de alguns desses comandos mais usados, e explicaremos resumidamente sua função.

O Layout do Excel está exibido na figura abaixo e numerado de acordo com os itens a seguir:

1 Botão do Office
2 Barra de acesso rápido
3 Título do documento
4 Menus
5 Barra de fórmulas
6 Nome da célula
7 Célula (B22)
8 Planilhas
9 Botão visualização normal
10 Botão visualização da página
11 Pré-visualização de quebra de página
12 Zoom
13 Nova planilha



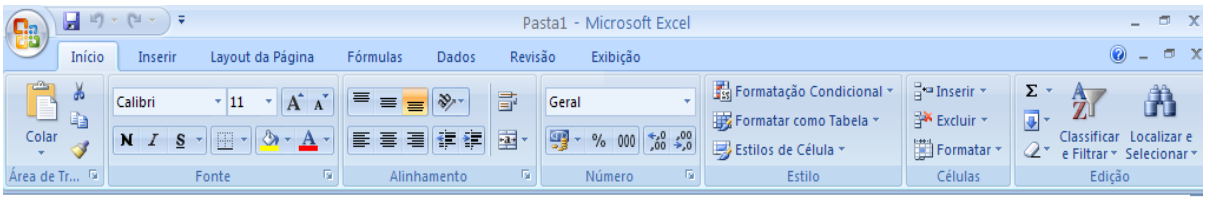
1.2.3 Menus

Existem sete menus básicos que organizam as funções mais usadas do programa. Cada um desses menus agrupam funções semelhantes.

Podemos localizar qualquer comando através desses menus, ou adicioná-los na barra de acesso rápido, como veremos mais adiante. A seguir, encontram-se os menus pré-definidos, onde podemos localizar os seguintes comandos:

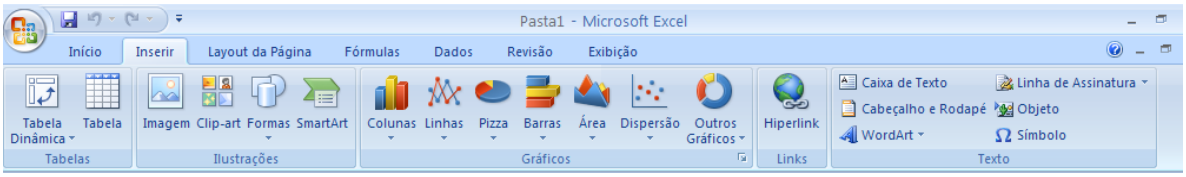
Menu Início

A maior parte dos botões de formatação de texto, tamanho da fonte, tipo de fonte (Arial,...), alinhamento do texto, copiar, recortar e colar encontram-se no menu Início.



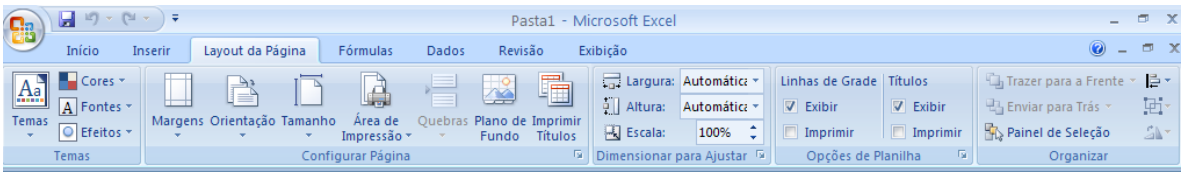
Menu Inserir

No menu Inserir podemos encontrar várias coisas que podemos inserir na planilha, tais como gráficos, tabelas, caixas de texto, símbolos, e outros.



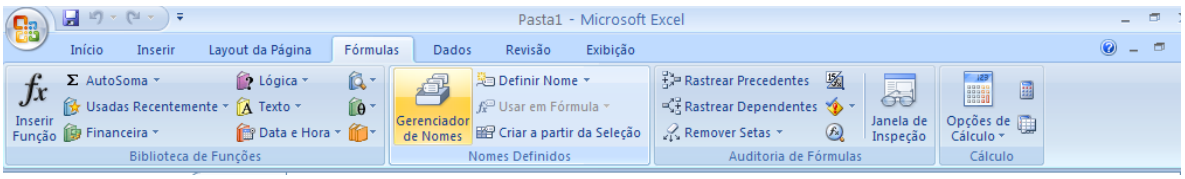
Menu Layout da Página

É no menu Layout da Página que configuramos as margens, orientação e tamanho da folha, quebra de texto, plano de fundo, altura e largura das células, exibição das linhas de grade (“exibir” apenas gera as linhas virtualmente para facilitar visualização na criação da planilha e Imprimir permite que essas linhas sejam impressas numa folha).



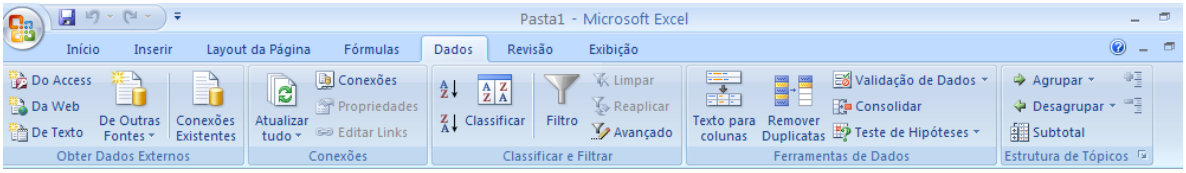
Menu Fórmulas

No menu Fórmulas podemos localizar comandos de gerenciamento dos nomes das células, rastreamento de precedentes e dependentes, Janela de Inspeção e a biblioteca de funções onde estão armazenados comandos de lógica (se, e, ou,...), funções trigonométricas (seno, cosseno,...) e outras.



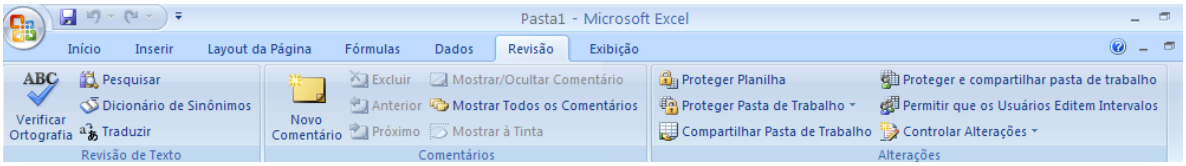
Menu Dados

Menu basicamente usado para criar filtros, classificar em ordem crescente/alfabética ou decrescente, além de estruturas de tópicos para agrupamento de linhas dependentes (como pastas e subpastas no Windows Explorer).



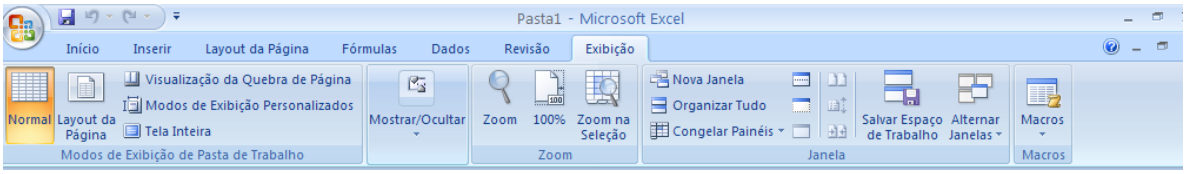
Menu Revisão

Comandos de comentários numa célula e revisão ortográficas estão localizados no menu Revisão.



Menu Exibição

É nesse menu onde se encontram as ferramentas de zoom, exibição de linhas de grade, barra de fórmula e títulos, além do modo de exibição da pasta de trabalho.



1.2.4 Criando planilhas simples

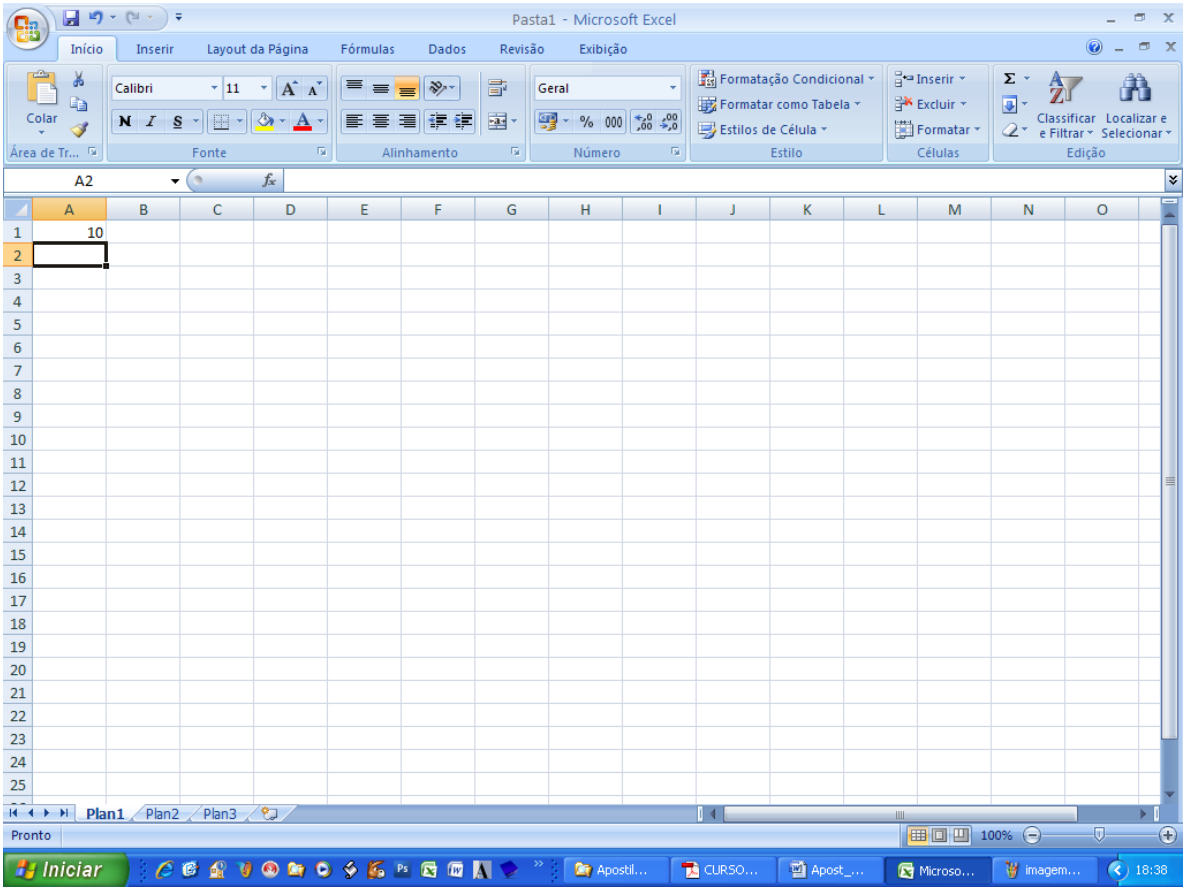
Inserindo dados

Para inserir dados numa planilha basta clicar numa célula e digitar o valor ou texto de entrada, conforme exemplo na figura abaixo.

Exemplo:

Inserindo um valor (10) na célula A1:

Clicar na célula, digitar 10 e aperte a tecla ENTER



Pode-se inserir um texto da mesma forma. Caso o texto ou valor for maior que o espaço pré-definido para a célula, basta redefinir o tamanho desta. Para tanto, veja Formatação de dados e células, a seguir.

Formatação de dados e células

Para redefinir o tamanho das células, basta clicar com o botão direito no número da linha ou nome da coluna e escolher a opção altura da linha/largura da coluna. Outra opção, mais simples, é arrastar com o mouse o limite da linha/coluna. Para formatação automática, basta dar duplo clique na divisória da linha/coluna, na barra de títulos, quando tiver algum texto escrito na célula.

Inserindo e excluindo linhas e colunas

Quando estamos criando uma tabela, podemos nos deparar com a necessidade de inserir ou excluir linhas e/ou colunas. Para tal, basta clicar com o botão direito do mouse na linha ou coluna que se deseja excluir e selecionar a opção Excluir. Para inserir uma nova linha/coluna, selecionamos a linha/coluna em que desejamos adicionar com o botão direito do mouse e clicamos em Inserir. Por exemplo: Se você

quiser inserir uma linha nova entre a linha 1 e 2, clique na linha 2 com o botão direito do mouse e escolha a opção Inserir. A antiga linha 2 passa a ser linha 3, a antiga linha 3 passa a ser linha 4, e assim por diante. O mesmo é válido para colunas.

Fazendo referências de dados para cálculos

Para se referir a uma célula, basta escrever na fórmula o nome desta. Por exemplo, para nos referir a um valor presente na célula A1, na barra de fórmulas não devemos escrever o valor, e sim o nome A1.

Operações básicas de ADIÇÃO, SUBTRAÇÃO, MULTIPLICAÇÃO, DIVISÃO e POTENCIAÇÃO.

Podemos realizar todo tipo de cálculo através dos operadores aritméticos, tais como Adição (+), Subtração (-), Multiplicação (*), Divisão (/), Potenciação(^), e outros. Para tais operações entre células, depois de introduzirmos os valores, basta que façamos referência ao nome da célula que o valor acompanhará sempre que está for referida.

ATENÇÃO!

Para que o Excel reconheça uma função é necessário colocar o sinal de igual (=) antes de qualquer comando.

Conversões de ângulos e operações trigonométricas

O Excel é pré-configurado para identificar o valor de um ângulo em radianos. Para nos referirmos ao número π no Excel, devemos escrever “PI()”.

Usamos o comando GRAUS(num) para converter um valor de ângulo em radiano para grau. Por exemplo: =GRAUS(2 * PI()), a resposta é 360º.

Para conversão inversa, de graus para radianos, usamos o comando RADIANOS(num). Exemplo: =RADIANOS(180), a resposta é 3,141593 que é o valor decimal de π.

Para realizarmos operações trigonométricas, os valores devem estar em radianos. Para tanto podemos usar os comandos combinados de conversão de ângulo de graus para radianos dentro da função trigonométrica, como podemos ver na figura abaixo.

Observamos na barra de fórmulas a diferença de introduzirmos os comandos “=SEN(RADIANOS(90))”, teremos como resultado o valor 1 e “=SEN(90)”, teremos como resultado o valor 0,8939 que não é o resultado correto do seno de π/2.

As demais funções trigonométricas estão listadas abaixo:

=SEN(num);
=ASEN(num);
=COS(num);
=ACOS(num);
=TAN(num);
=ATAN(num);

Podemos realizar operações aritméticas juntamente com trigonométricas, conversões de ângulos, tudo junto numa mesma célula. Abaixo temos um exemplo que combina essas operações:
=(sen(radianos(45))^2)+(cos(radianos(30))^2)-2*45/30, teremos como resposta o valor de -1,75.

1.2.5 Gráficos

O Excel oferece uma ferramenta de criação de gráficos de vários modelos diferentes. Veremos a seguir alguns tipos mais comuns deles.

Tabelas
EXERCÍCIO 1:

Para fazer um gráfico, deve-se primeiro criar uma tabela bem organizada com os dados que se deseja trabalhar.
Existem várias maneiras de criar gráficos. Veja o exemplo abaixo. Inicia-se o processo criando uma tabela com os dados necessários.

TABELA DE ÍNDICE DE MASSA CORPORAL E CIRCUNFERÊNCIA CRANIANA

RESULTADOS DAS MEDIDAS							
NOME	**MASCULINO**	**FEMININO**	**IDADE**	**ALTURA(m)**	**MASSA(kg)**	**IMC**	**CC(cm)**
A		x	21	1,62	52,5	20,00	56,5
B		x	20	1,58	55,9	22,39	54
C		x	23	1,68	62	21,97	58,8
D		x	20	1,71	65	22,23	55
E		x	20	1,62	48,5	18,48	55,3
F		x	20	1,55	65	27,06	57,5
G		x	24	1,66	60	21,77	53
H	x		20	1,70	83,7	28,96	58
I	x		24	1,55	50	20,81	52
J		x	21	1,66	59	21,41	55,5
K		x	20	1,73	57	19,05	54
L		x	22	1,69	53	18,56	53,5
M		x	21	1,66	53,6	19,45	55
N	x		20	1,84	72,7	21,47	61
O		x	22	1,54	46,5	19,61	55
P		x	20	1,56	56	23,01	54
Q		x	20	1,56	60	24,65	53
R		x	21	1,58	66,8	26,76	56
S		x	21	1,70	79	27,34	56
T	x		25	1,76	78,2	25,25	58
U		x	23	1,74	67	22,13	55
V		x	25	1,61	56	21,6	53
W		x	20	1,66	58	21,05	54
X		x	21	1,65	61	22,41	54
Z		x	23	1,65	47	17,26	53

O cálculo do IMC é feito dividindo a massa pela altura ao quadrado (MASSA/(ALTURA)^2), logo a coluna IMC deverá ser preenchida usando essa fórmula.

Em seguida, devem ser selecionados os dados que se deseja inserir no gráfico. Para marcar vários dados ao mesmo tempo, basta clicar na primeira célula superior à esquerda e deslocar arrastando o mouse até a última célula, inferior direita, mantendo o botão esquerdo pressionado. Para selecionar colunas ou linhas não adjacentes tem que pressionar a tecla Ctrl durante a seleção.

Para criar um gráfico que relacione, a cada sujeito, sua idade e peso, selecionamos os dados referidos e escolhe-se um tipo de gráfico adequado para os dados, através do menu Inserir. Existem várias opções de gráficos que podem ser escolhidas, tais como Coluna 2D, Coluna 3D, Cilindro, Cone e Pirâmide. Pode-se ainda verificar outros tipos de gráficos na opção Todos os Tipos de Gráficos.

Quando se cria um gráfico, um novo menu, Ferramentas de Gráfico, é exibido, para se trabalhar com o gráfico. Pode-se alterar o Design do gráfico, alterar os dados selecionados e mesmo o tipo de gráfico no sub menu Design

O sub menu Layout é o mais usado na formatação de um gráfico. Podemos modificar todo o layout do gráfico com relação a eixos, linhas de grade, editar a legenda e títulos, modificar/acrescentar dados ao gráfico e muito mais.

O último sub menu que aparece em Ferramentas de Gráfico é o Formatar. Como o próprio nome induz ao usuário a pensar, é nesse menu que será trabalhado a forma das letras, cores, estilos, e outros.

EXERCÍCIO 2:

Digitar a tabela a seguir no Excel. Seguir as seguintes observações:

1 – As colunas de totais no final da tabela deverão ser preenchidas usando a fórmula SOMA.

2 – A coluna FREQ. deverá ser preenchida usando a fórmula CONT.VALORES e pegando como valores a linha correspondente as colunas de 2ª até sábado (as quais estão marcadas com X).

3 – Todas as datas deverão ser preenchidas usando a fórmula HOJE da função DATA E HORA.

4 – O campo “No. de Lojas” deverá ser preenchido usando a fórmula CONT.VALORES com a coluna Bandeira/Loja.

5 – O campo “No. de Visitas Mês” deverá ser preenchido usando a fórmula CONT.VALORES com as colunas de 2ª à Sábado de todas as Lojas (as quais estão marcadas com X).

6 – O campo “Dias Úteis” deverá ser preenchido usando a fórmula CONT.VALORES com a linha de 2ª à Sábado.

7 – O campo “Média Diária Visitas” deverá ser preenchido usando a fórmula MÉDIA com a coluna FREQ.

8 – Desenhar três gráficos como abaixo:

A – Gráfico de barra do nome da loja versus preço.

B – Gráfico de barra do nome da loja versus tempo de visita.

C – Gráfico de barra do nome da loja versus preço versus tempo de visita.

9 – Copiar cada gráfico para o editor de texto (Word).

10 – Modificar o valor do preço de uma determinada loja para um valor bem mais alto e verificar a atualização automática dos gráficos no Excel e no Word.

PROMOÇÃO E MERCHANDISING

ROTEIRO DIÁRIO DE VISITAS DE PROMOTORES

PROMOTOR	PAULO JOSÉ	**SUPERVISOR**	JOAQUIM MARIA	**ROTEIRO**	REGIÃO SUL
ENDEREÇO	AV. Amazonas - 1000	**TELEFONE CELULAR**	9999-9999	**N° DE LOJAS**	16
CEP	31-130000 Região: BH	**TELEFONE RESIDENCIAL**	3333-3333	**N° Visitas Mês**	35
DATA DE INICIO	25/10/2016 UF: Minas Gerais			**DIAS UTEIS**	6
E-MAIL: paulo@hotmail.com	RG: MG-1.111.111			**MÉDIA VISITAS**	2,2

2ª	3ª	4ª	5ª	6ª	Sáb	CÓDIGO	BANDEIRA/LOJA	ENDEREÇO	BAIRRO	CIDADE	GERENCIADOR	FREQ	TEMPO	PREÇO
x		x		x	x	123	CARREFOUR BH	BR356, 3049	BELVEDERE	BH	1	4	1	10
x		x		x		124	WALMART	AV G. DAVID SARNOFF, 6	C. INDUSTRIAL	CONTAGEM	2	3	2	12
			x			125	VIA BRASIL CONT	AV J. CÉSAR DE OLIVEIRA, 5	ELDORADO	CONTAGEM	1	1	1	42
	x		x			126	CARREFOUR CONT.	BR 381, 3000	CONTAGEM	CONTAGEM	2	2	3	34
x		x		x		127	CARREFOUR PAMP.	AV P. CARLOS LUZ, 4055	ENGENHO	BH	1	3	2	14
	x		x			128	VIA BRASIL PAMP.	AV DOM PEDRO I, 402	ITAPÕA	BH	2	2	1	36
x		x		x	x	129	CARREFOUR DEL REY	AV P. CARLOS LUZ, 3000	CAIÇARA	BH	1	4	2	24
x			x			130	EXTRA BELVEDERE	R MARIA LUIZA SANTIAGO, 10	BELVEDERE	BH	2	2	3	33
			x			131	EXTRA CONTAGEM	AV J. CESAR DE OLIVEIRA	ELDORADO	CONTAGEM	1	1	3	23
x		x		x	x	132	SAMS CLUB	AV G. DAVID SARNOFF, 5	C. INDUSTRIAL	CONTAGEM	2	4	2	11
	x		x			133	SUPER NOSSO 1	AV G. DAVID SARNOFF, 5	BELVEDERE	BH	1	2	1	10
			x			134	SUPER NOSSO 2	AV TANCREDO NEVES, 400	NOVA LIMA	BH	2	1	1	34
	x					135	HIPER SUPER MINAS	AV TANCREDO NEVES, 1000	CASTELO	BH	1	1	1	32
			x			136	SUPER NOSSO	AV HERÁCLITO MOURÃO, 1700	CASTELO	BH	2	1	2	54
	x		x			137	MART PLUS	AV LUIZ PAULO FRANÇO, 251	BELVEDERE	BH	1	2	3	42
	x		x			138	VERDEMAR	AV N. SENHORA DO CARMO, 2	SION	BH	2	2	2	18
							ATUALIZADO	30/04/2015			**TOTAIS**	35	30	429

OBSERVAÇÃO 1:

OBSERVAÇÃO 2:

EXERCÍCIO 3:

Criar a tabela e os gráficos abaixo no Excel de forma que a visualização seja idêntica à apresentada abaixo:

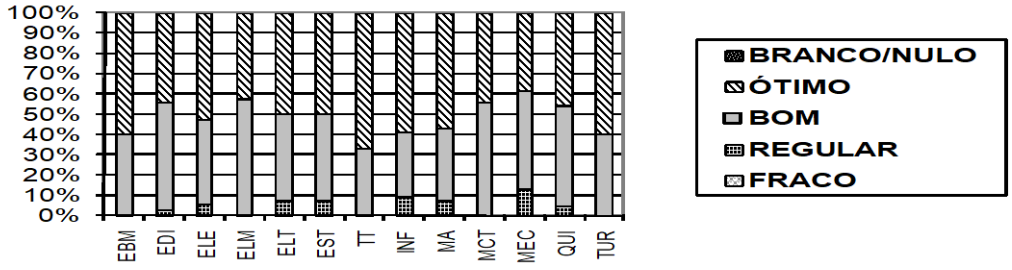
Qual o nível de adequação da carga horária destinada ao estágio?

Carga horária destinada ao Estágio

CURSO	CONCEITO					TOTAL
	FRACO	REGULAR	BOM	ÓTIMO	BRANCO/NULO	
EBM	0	0	2	3	0	5
EDI	0	1	19	16	0	36
ELE	0	2	15	19	0	36
ELM	0	0	4	3	0	7
ELT	0	1	6	7	0	14
EST	0	1	6	7	0	14
TT	0	0	3	6	0	9
INF	0	2	7	13	0	22
MA	0	1	5	8	0	14
MCT	0	0	5	4	0	9
MEC	0	5	19	15	0	39
QUI	0	1	12	11	0	24
TUR	0	0	4	6	0	10
TOTAL	0	14	107	118	0	239

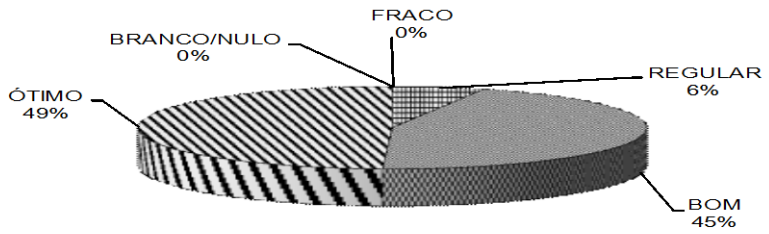
Fonte: "Questionário de Avaliação Individual" - 71º Seminário de Conclusão dos Cursos Técnicos da E.P.T. do CEFET-MG / CPE - Abril/2011

Carga horária destinada ao Estágio



Fonte: "Questionário de Avaliação Individual" - 71º Seminário de Conclusão dos Cursos Técnicos da E.P.T. do CEFET-MG / CPE - Abril/2011

Carga horária destinada ao Estágio



Fonte: "Questionário de Avaliação Individual" - 71º Seminário de Conclusão dos Cursos Técnicos da E.P.T. do CEFET-MG / CPE - Abril/2011

1.2.6 Funções

120	136	123
111	124	234
120	136	100

=ORDEM(111;A1:C3;1) - O número procurado, a matriz onde o número se encontra, 0 para contar em ordem decrescente e qualquer número para ordem crescente. Essa função encontra a ordem em que o número se encontra, na ordem crescente ou decrescente. Na ordem crescente temos: 100, 111, 120, 123, 124... Logo o número 111 está em 2º lugar na ordem crescente da série.

=MENOR(A1:C3;2) – Procura o menor valor citado pelo segundo argumento. No exemplo, procura o segundo menor valor (do menor para o maior).

=MAIOR(A1:C3;2) – Procura o maior valor citado pelo segundo argumento. No exemplo, procura o segundo maior valor (do maior para o menor).

=MODO(A1:C3) – Retorna o valor que mais se repete em um intervalo. Senão tiver valor repetido dá erro. Se houver mais de um valor repetido o mesmo tanto retorna o primeiro encontrado.

=MÉDIA(A1:A3) – Calcula a média de valores.

=MÍNIMO(A1:A3) – Retorna o valor mínimo em uma lista de valores.

=MÁXIMO(A1:A3) – Retorna o valor máximo em uma lista de valores.

=CONT.NÚM(A1:A4;B1:B4) – Conta o número de células que contém números.

=CONT.SE(A1:A3;">111") - Intervalo e condição.

=CONTAR.VAZIO(C1:C7) – Conta o número de células vazias no intervalo.

=CONT.VALORES(B1:B7;124) – Calcula o número de células que não estão vazias e o número de células que contem o argumento.

=E(A1>A2;A2>A3) – Verifica se os argumentos são verdadeiros. Caso sejam retorna verdadeiro, ao contrário retorna falso.

=NÃO(VERDADEIRO) – Inverte verdadeiro em falso e falso em verdadeiro.

=OU(A1=A2;B1=B3) – Verifica se algum argumento é verdadeiro. Caso seja retorna verdadeiro, se todos forem falsos retorna falso.

=SE(B1=B2;B1;B2) – Verifica a condição do primeiro argumento, se for verdadeiro retorna o valor do segundo argumento, se for falso retorna o valor do terceiro argumento.

=SOMASE(A1:A3;">111") – Intervalo e condição.

=TRUNCAR(1234,656;2) – Limita o número para a quantidade de casas solicitadas sem arredondar.

=MED(99; 100; 101; 120) – Retorna o valor central de uma lista de valores (lista colocada em ordem crescente para encontrar o valor). Se o número de valores for par, calcula a média aritmética entre os valores medianos (valores do meio). No exemplo retorna 100,5.

=MOD(numerador; denominador) – Retorna o resto da divisão.

=INT(número) – Pega a parte inteira de um número.

=N(valor) – Converte um valor não numérico em um número. VERDADEIRO em 1 e qualquer outro valor em zero (0).

=QUOCIENTE(numerador; denominador) – Retorna a parte inteira de uma divisão.

2 MATLAB

2.1 Matemática Elementar

Da mesma forma que sua calculadora, o MATLAB é capaz de executar matemática elementar.

(4 x 25) + (6 x 22) + (2 x 99) = 430

Além da adição e multiplicação, o MATLAB dispõe das seguintes operações aritméticas elementares:

Operação	Símbolo	Exemplo
Adição, a + b	+	7 + 3
Subtração, a – b	-	7 – 3
Multiplicação, a * b	*	2.2 * 5.7
Divisão, a / b	/ ou \	42 / 7 = 7 \ 42
Potenciação, a ^ b	^	10^2

A ordem de precedência é a mesma seguida pela matemática: multiplicação e divisão (ambas com a mesma ordem) seguidas pela adição e subtração (ambas com a mesma ordem). Podem-se usar parênteses para alterar essa ordem.

2.2 O espaço de trabalho do MATLAB

Quando está trabalhando na janela de comandos, o MATLAB lembra-se dos comandos que você introduz, assim como dos valores de quaisquer variáveis criados. Dizemos então que esses comandos e variáveis residem no espaço de trabalho do MATLAB e podem ser chamados a qualquer momento que desejar. Por exemplo, digite na janela de comandos:

livro = 10

Para verificar o valor da variável livro, basta que você o peça introduzindo seu nome após o prompt (>>).

Se não conseguir se lembrar do nome de uma variável, pode pedir para o MATLAB apresentar uma lista das variáveis que ele conhece, utilizando o comando ***who***.

Para chamar comandos previamente utilizados, o MATLAB utiliza as teclas de cursor (←, →, ↑, ↓) do seu teclado. Por exemplo, ao pressionar a tecla ↑ uma vez,

chamamos o comando mais recente do prompt do MATLAB. As teclas ← e → são usadas para mover o cursor dentro da linha de comando no prompt do MATLAB.

2.3 Variáveis

Assim como qualquer outra linguagem de programação, o MATLAB tem regras a respeito dos nomes de variáveis. Nomes de variáveis devem ser palavras únicas, sem a inclusão de espaços. As regras para nomes de variáveis são:

- As variáveis são sensíveis a maiúsculas e minúsculas (Livro, livro, LIVRO, liVRo, etc).
- As variáveis podem conter até 63 caracteres (depende da versão do Matlab).
- Os nomes de variáveis devem começar com uma letra, seguida de um número qualquer de letras, algarismos ou sublinhas.

Além dessas regras de nomes de variáveis, o MATLAB tem diversas variáveis especiais. Elas são:

Variável	Valor
ans	Nome de variável padrão usado para resultados.
pi	Razão entre o perímetro da circunferência e seu diâmetro.
eps	Menor número que, somado a 1, cria um número de ponto flutuante maior do que 1 no computador.
flops	Contador do número de operações de ponto flutuante.
inf	Infinito, por exemplo, 1/0.
NaN ou nan	Não-número, por exemplo, 0/0.
i (e) j	I = j = raiz quadrada de –1.
nargin	Número de argumentos de entrada de uma função.
nargout	Número de argumentos de saída de uma função.
realmin	Menor número real positivo utilizável.
realmax	Maior número real positivo utilizável.

As variáveis do espaço de trabalho do MATLAB podem ser excluídas incondicionalmente usando-se o comando ***clear***. Por exemplo:

- clear livro - exclui somente a variável livro.
- clear livro caneta - exclui as variáveis livro e caneta.

- clear - exclui todas as variáveis do espaço de trabalho.

2.4 Outras Características Básicas

Todo texto depois do sinal de porcentagem (%) é considerado um comentário:

```
>> livro = 2    % Número de livros.
livro =
    2
```

Pode-se colocar mais de um comando em uma linha, separando-os por vírgula ou ponto-e-vírgula:

```
>> livro = 2, caneta = 3, lapis = 5;
livro =
    2
caneta =
     3
```

A vírgula diz ao MATLAB para mostrar o resultado; o ponto-e-vírgula suprime a visualização.

```
>>total = livro + caneta + ...   (shift enter)
          lapis
  total =
          10
```

Como se observa anteriormente, uma sucessão de três pontos indica ao MATLAB que o resto do comando aparece na linha seguinte. Isso só funcionará se a sucessão de três pontos estiver entre nomes de variáveis e operações. Isso significa que um nome de variável não pode ser dividido em duas linhas. Da mesma maneira, comentários não podem ser continuados na linha seguinte.

A execução do MATLAB pode ser interrompida a qualquer momento pressionando Ctrl-Q (pressionando as teclas Ctrl e Q simultaneamente). O comando quit termina a execução do MATLAB.

2.4.1 Números Complexos

Uma das características mais poderosas do MATLAB é que não é necessário um manuseio especial para os números complexos. No MATLAB, números complexos podem ser formados de várias maneiras. Alguns exemplos:

```
>>c1 = 1-2i          % o i justaposto indica a parte imaginária
c1 =
```

```
    1.0000  -  2.0000i
>>c1 = 1-2j              % j também funciona
c1 =
    1.0000  -  2.0000i
>>c2 = 3*(2-sqrt(-1)*3)
c2 =
    6.0000  -  9.0000i
>>c3 = sqrt(-2)
c3 =
    0 + 1.4142i
>>c4 = 6+sin(.5)*i
c4 =
    6.0000  +  0.4794i
```

No último exemplo, o valor-padrão do MATLAB i = $\sqrt{-1}$ foi utilizado para formar a parte imaginária. É preciso, nesse caso, multiplicar por i, já que, para o MATLAB, sin(0.5)i não têm sentido. A terminação com os caracteres i ou j, conforme mostrado nos dois primeiros exemplos, só funciona com números simples, não com expressões.

As operações matemáticas com números complexos são escritas da mesma maneira que aquelas com números reais:

```
>>c5=(c1+c2)/c3          % dos dados anteriores
c5 =
        -7.7782 – 4.9497i
>>verifique = i^2        % o quadrado de sqrt(-1) tem de ser igual a -1!
verifique  =
        -1
```

Em geral, as operações com números complexos resultam em números complexos. Nos casos em que há uma parte desprezível, real ou imaginária, remanescente, você pode usar, respectivamente, as funções real e imag para extrair a parte real e a imaginária.

Como exemplo final de aritmética complexa, considere a identidade de Euler (pronuncia-se oiler), que relaciona a forma polar à forma retangular de um número complexo:

$$M\angle\Theta \equiv M * e^{j\Theta} = a + bi$$

Em que a forma polar é dada por um módulo M e um ângulo Θ e a retangular é dada por a + bi. As relações entre essas fórmulas são:

$$M = \sqrt{a2 + b2}$$

$$\Theta = \tan^{-1}(b/a)$$

$$a = M\cos\Theta$$

$$b = M\text{sen}\Theta$$

No MATLAB, a conversão entre polar e retangular utiliza as funções real, imag, abs e angle:

```
>>c1
c1 =
    1.0000  -  2.0000i
>>mod_c1 = abs(c1)
mod_c1 =
    2.2361
>>angulo_c1 = angle(c1)
angulo_c1 =
    -1.1071
>>graus_c1 = angulo_c1*180/pi
graus_c1 =
     -63.4349
>>real_c1 = real(c1)
real_c1 =
     1
>>imag_c1 = imag(c1)
imag_c1 =
    -2
```

A função MATLAB abs calcula o módulo de números complexos ou o valor absoluto de números reais, dependendo do que você utiliza como argumento. Além disso, a função MATLAB angle calcula o ângulo de um número complexo em radianos.

2.5 Funções Matemáticas Elementares

A tabela a seguir apresenta uma lista parcial de funções básicas suportadas por MATLAB. A maior parte dessas funções é usada da mesma forma que você as escreveria matematicamente:

FUNÇÕES MATEMÁTICAS ELEMENTARES	
abs (x)	Valor absoluto ou módulo de um número complexo
acos (x)	Arco cosseno

angle (x)	Ângulo de um número complexo
asin (x)	Arco seno
atan (x)	Arco tangente
ceil (x)	Arredondar para mais infinito (para cima)
cos (x)	Cosseno (ângulos em radianos)
cosd (x)	Cosseno (ângulos em graus)
exp (x) –> Exemplo: exp(1) = e^1	Exponenciação (potência do nº neperiano)
fix (x)	Arredondar para zero (arredonda para baixo)
floor (x)	Arredondar para menos infinito (arredonda para baixo)
gcd (x, y)	Máximo divisor comum dos inteiros x e y.
imag (x)	Parte imaginária de um número complexo
lcm (x, y)	Mínimo múltiplo comum dos inteiros x e y.
log (x)	Logaritmo natural (neperiano – ln)
log2 (x)	Logaritmo na base 2
log10 (x)	Logaritmo na base 10
real (x)	Parte real de um número complexo
rem (x, y)	Resto da divisão de x por y
round (x)	Arredondar para o próximo número inteiro
sign (x)	Função sinal: retorna o sinal de um argumento. Por exemplo: sign(1.2) = 1, sign(-23.4) = -1, sign(0) = 0
sin (x)	Seno (ângulos em radianos)
sind (x)	Seno (ângulos em graus)
sqrt (x)	Raiz quadrada
tan (x)	Tangente (ângulos em radianos)
tand (x)	Tangente (ângulos em graus)

2.5.1 Formatos de Visualização de Números

Quando o MATLAB apresenta resultados numéricos, segue diversas regras. Por definição, se o resultado for um número inteiro, o MATLAB apresenta-o como inteiro. Da mesma forma, quando o resultado é um número real, o MATLAB apresenta-o com, aproximadamente, quatro dígitos à direita do ponto decimal. Se os dígitos significativos do resultado estiverem fora desse limite, o MATLAB apresenta o resultado em notação científica, de maneira semelhante às calculadoras científicas. Você pode anular esse comportamento-padrão especificando um formato numérico diferente por meio do item *Preferences*, do menu *File*, se este estiver disponível, ou introduzindo o comando MATLAB apropriado no prompt. Para x = 50.833, esses formatos numéricos são:

Comando MATLAB	X	Observações
format short	50.833	5 dígitos.
format long	50.83333333333334	16 dígitos.
format short e	5.0833e+01	5 dígitos + expoente.
format long e	50.83333333333334e+01	16 dígitos + expoente.
format short g	50.833	O melhor entre format short e format short e.
format long g	50.8333333333333	O melhor entre format long e format long e.
format hex	40496aaaaaaaaaab	Hexadecimal.
format bank	50.83	2 dígitos decimais.
format +	+	Positivo, negativo ou zero.
format rat	305/6	Aproximação racional.

É importante observar que o MATLAB não altera a representação interna de um número quando optamos por diferentes formatos de apresentação; só a visualização do número é alterada.

2.6 O comando *help*

O comando *help* é a maneira mais simples de se conseguir ajuda caso você saiba exatamente o tópico a respeito do qual você necessita de informações. Ao digitar *help* <tópico>, a tela apresenta a ajuda sobre o tópico, caso ela exista. Por exemplo:

```
>> help sqrt
< Texto da ajuda apresentado>
```

2.7 Operações com Vetores

2.7.1 Vetores Simples

Para criar um vetor no MATLAB, tudo que você precisa fazer é começar com um colchete esquerdo, introduzir os valores desejados separados por espaços (ou vírgulas) e fechar o vetor com um colchete direito. Por exemplo, dado o vetor x, calcular os valores para cada elemento do seno de x e armazená-lo em y.

```
x = [0  0.1*pi  0.2*pi  0.3*pi  0.4*pi  0.5*pi];
```

```
y = sin(x)
y =
         0   0.3090   0.5878   0.8090   0.9511   1.0000
```

Uma vez que utilizamos espaços para separar os valores de um vetor, os números complexos introduzidos como valores de um vetor não podem ter espaços inclusos, a menos que as expressões estejam entre parênteses. Por exemplo, [1 -2i 3 4 5+6i] contém cinco elementos.

2.7.2 Endereçamento Vetorial

No MATLAB, elementos de vetores individuais são acessados usando-se *subscritos*, ou seja, x(1) é o primeiro elemento de x, x(2) é o segundo elemento de x, e assim por diante.

Para acessar um bloco de elementos ao mesmo tempo, o MATLAB utiliza a *notação de dois pontos*: x(1:5), esses são os elementos de x do primeiro ao quinto 1:5 diz para começar com o primeiro e ir até o quinto. Se tivermos x(3:-1:1), esses são o terceiro segundo e primeiro elementos em ordem reversa. 3:-1:1 diz “comece no terceiro, conte regressivamente de um em um e pare no primeiro”.

Se tivermos x([8 2 9 1]) estaremos usando um outro vetor [8 2 9 1] para extrair os elementos do vetor x na ordem que os quisermos! O primeiro elemento tomado é o oitavo, o segundo é o segundo, o terceiro é o nono e o quarto é o primeiro.

2.7.3 Construção de Vetores

Anteriormente, nós introduzimos os valores de x digitando cada um de seus elementos. Isso foi possível porque o vetor x possuía poucos elementos. Usando-se a notação de dois pontos, duas outras maneiras de introduzir os valores de x são:

```
x = (0:0.1:0.5)*pi
x = linspace(0, 0.5*pi, 6)
```

No primeiro caso acima, a notação de dois pontos (0:0.1:0.5) cria um vetor que começa com 0, incrementa de 0.1 e termina com 0.5. Cada elemento nesse vetor é multiplicado por pi para criar os valores desejados em x. No segundo caso, a função MATLAB linspace é usada para criar x. Os argumentos da função são descritos por:

linspace (primeiro_valor, ultimo_valor, número_de_valores)

No caso especial, no qual um espaçamento logarítmico faça-se necessário, o MATLAB apresenta a função logspace:

logspace (0, 2, 3)
logspace (primeiro_expoente, ultimo_expoente, numero_de_valores)

A resposta será: 1 10 100.
Algumas vezes necessita-se de um vetor que não é convenientemente descrito por uma relação de espaçamento linear ou logarítmica. Não há uma maneira uniforme de criar esses vetores. Entretanto, o endereçamento de vetores e a capacidade de combinar expressões podem ajudar a eliminar a necessidade de introduzir elementos individuais de cada vez:

```
>> a=1:5, b=1:2:9
a =
     1   2   3   4   5
b =
     1   3   5   7   9
```

cria dois vetores. Lembre-se de que instruções múltiplas podem aparecer na mesma linha se estiverem separadas por vírgula ou ponto-e-vírgula.

```
>> c=[b a]
c =
     1   3   5   7   9   1   2   3   4   5
```

cria um vetor c composto dos elementos de b seguidos dos de a:

```
>> d=[a(1:2:5) 1 0 1]
d =
     1   3   5   1   0   1
```

cria um vetor d composto do primeiro, do terceiro e do quinto elemento de a, seguidos de três elementos adicionais.

2.7.4 Matemática Vetor-Escalar

Outras operações matemáticas simples entre escalares e vetores seguem a mesma interpretação natural. Adição, subtração, multiplicação e divisão por um escalar simplesmente aplica a operações a todos os elementos do vetor:

```
>> a =
```

```
        1   2   3   4   5
>> a-2
ans =
        -1   0   1   2   3
```

subtrai 2 de cada elemento de a

```
>> 2*a-1
ans =
        1   3   5   7   9
```

multiplica cada elemento em a por 2 e subtrai 1 de cada elemento do resultado.

2.7.5 Matemática Vetor-Vetor

Operações matemáticas entre vetores não são tão simples quanto aquelas entre escalares e vetores. É claro que operações entre vetores de comprimentos diferentes são difíceis de definir e são de valor duvidoso. Entretanto, quando dois vetores têm o mesmo comprimento, adição, subtração, multiplicação e divisão aplicam-se de elemento em elemento. Por exemplo, chama de novo os vetores usados anteriormente:

```
>> a, b
a =
        1   2   3   4   5
b =
        1   3   5   7   9
>> a + b
ans =
        2   5   8   11   14
```

soma os dois vetores elemento por elemento e coloca o resultado na variável padrão ans.

```
>> ans-b
ans =
        1   2   3   4   5
```

subtrai b do resultado mais recente, dando-nos novamente os valores de a.

```
>> 2*a-b
ans =
        1   1   1   1   1
```

multiplica todos os elementos de a por 2, subtrai b deles e coloca o resultado em ans.

A multiplicação e a divisão elemento por elemento funcionam de maneira semelhante, mas usam uma notação ligeiramente não-convencional:

```
>> a.*b
ans =
        1   6   15   28   45
```

Aqui, nós multiplicamos os vetores a e b elemento por elemento usando o símbolo de multiplicação escalar (**.***). O ponto que precede o asterisco, símbolo padrão da multiplicação, diz ao MATLAB para fazer a multiplicação elemento por elemento. A multiplicação sem o ponto significa multiplicação matricial, que será discutida mais tarde.

A divisão de vetores, ou divisão escalar, também requer o uso do símbolo pontuado:

```
>> a./b;
>> b.\a;
```

A potenciação de vetores é definida de diversas maneiras. Assim como na multiplicação e na divisão, **^** está reservado para a potenciação de matrizes e **.^** é usado para significar a potenciação elemento por elemento:

```
>> a.^2;
```

eleva ao quadrado os elementos individuais de a.

```
>> 2.^a;
```

eleva dois a potencia de cada elemento no vetor a.

```
>> b.^a
ans =
        1   9   125   2401   59049
```

eleva os elementos de b aos elementos correspondentes de a.

```
>>b.^(a-3)
ans =
        1.0000   0.3333   1.0000   7.0000   81.0000
```

mostra que operações escalares e de vetores podem ser combinadas.

Exercício:

1 – Teste todas as operações matemáticas feitas com vetor linha em vetor de coluna.

2.7.6 Orientação de Vetores

Nos exemplos acima, os vetores continham uma única linha e múltiplas colunas e por isso são chamados de vetores de linha. Também é possível construir um vetor que seja um vetor de coluna, que terá uma única coluna e múltiplas linhas. Nesse caso, toda a manipulação e a matemática de vetores também se aplica sem qualquer modificação. A única diferença é que os resultados são apresentados em colunas, em vez de em linhas. Para se criar o vetor de coluna basta especificar elemento por elemento e separar os valores com ponto-e-vírgula.

```
>> c = [1; 2; 3; 4; 5]
c =
  1
  2
  3
  4
  5
```

Para criar um vetor de coluna usando a notação de dois pontos início:incremento:fim ou as funções linspace e logspace, deve-se transpor a linha resultante em uma coluna usando-se o operador de transposição (‘) do MATLAB.

```
>> a = 1:3;
>>b = a’
b =
  1
  2
  3
```

Vimos anteriormente, na seção matemática vetor-vetor, que, “quando dois vetores têm o mesmo comprimento, adição, subtração, multiplicação e divisão aplicam-se elemento por elemento”. Agora que sabemos que vetores podem ter uma orientação de linha ou coluna, tal afirmativa tem de ser redefinida: quando dois vetores têm o mesmo comprimento e orientação, adição, subtração, multiplicação e divisão aplicam-se elemento por elemento. Em outras palavras, as operações entre vetores de orientações diferentes não são definidas na base de elemento por elemento.

Se um vetor pode ter uma linha e múltiplas colunas (vetor de linha) ou uma coluna e múltiplas linhas (vetor de coluna), são possíveis intuir facilmente que os vetores

podem, da mesma forma, ter múltiplas linhas e múltiplas colunas. Em particular, essa orientação retangular é útil quando todas as linhas têm o mesmo número de colunas. Vetores com múltiplas linhas e colunas são chamados matrizes. A criação de matrizes segue as mesmas diretrizes que a de vetores de linha e de coluna. Usam-se vírgulas ou espaços para separar os elementos de uma linha específica e pontos-e-vírgulas para separar as linhas individuais:

```
>>g = [1 2 3 4; 5 6 7 8]
g =
  1   2   3   4
  5   6   7   8
```

Além do ponto-e-vírgula, o uso de ***Return*** ou ***Enter*** durante a entrada de uma matriz também informa ao MATLAB para começar uma nova linha. O MATLAB reforça o fato de que todas as linhas devem ter o mesmo número de colunas.

Considerando-se esse formato de vetor retangular, os conceitos de matemática vetor-vetor discutidos anteriormente aplicam-se conquanto o tamanho dos vetores envolvidos na operação, isto é, o número de linhas e colunas, seja idêntico.

Exercício:

1 – Teste todas as operações matemáticas feitas com vetor de linha em matrizes.

Vimos anteriormente, que o comando ***who*** mostra o nome de todas as variáveis criadas pelo usuário. No caso de vetores, é importante também saber o tamanho das variáveis. No MATLAB, o comando ***whos*** fornece essa informação adicional.

A tabela seguinte ilustra as operações com vetores:

OPERAÇÕES DE VETORES DO MATLAB	
Dados ilustrativos: a = [a1 a2 ... an], b = [b1 b2 ... bn], c = {escalar}	
Adição escalar	a+c = [a1+c a2+c ... an+c]
Multiplicação escalar	a*c = [a1*c a2*c ... an*c]
Adição de vetores	a+b = [a1+b1 a2+b2 ... an+bn]
Multiplicação de vetores	a.*b = [a1*b1 a2*b2 ... an*bn]
Divisão direita de vetores	a./b = [a1/b1 a2/b2 ... an/bn]
Divisão esquerda de vetores	a.\b = [a1\b1 a2\b2 ... an\bn]
Potenciação de vetores	a.^c = [a1^c a2^c ... an^c] c.^a = [c^a1 c^a2 ... c^an] a.^b = [a1^b1 a2^b2 ... an^bn]

2.7.7 Operações Matriciais

Uma matriz A mxn é um quadro retangular de mn números reais (ou complexos) dispostos em m linhas horizontais e n colunas verticais.

Dizemos que A é m por n (m x n). Se m = n, dizemos que A é uma matriz quadrada de ordem n, e que os números a11, a22, ..., ann formam a diagonal principal de A. Referimo-nos ao número aij, que está na i-ésima linha e j-ésima coluna de A, como o (i, j)-ésimo elemento (ou coeficiente) de A, e frequentemente escrevemos como: A = [aij]. Vejamos como é representada uma matriz no MATLAB:

```
>> A=[1 2 3;4 5 6;7 8 9]
    A =
          1    2    3
          4    5    6
          7    8    9
```

Os elementos da mesma linha são separados por espaço e as linhas são separadas por ponto-e-vírgula. A matriz A acima é uma matriz quadrada, pois possui o mesmo número de linhas e colunas.

A tabela seguinte ilustra as operações básicas entre matrizes e escalares, matrizes e matrizes. R é a matriz resultado de dimensões de acordo com o resultado da operação realizada.

Operações de Matrizes do MATLAB	
Dados ilustrativos: A = [aij] e B = [bij] são matrizes, por exemplo m x n; c = {escalar}	
Adição de uma matriz por um escalar	A+c = Aij + c ($1 \le i \le m$, $1 \le j \le n$)
Multiplicação de uma matriz por um escalar	A*c = Aij * c ($1 \le i \le m$, $1 \le j \le n$)
Divisão de uma matriz por um escalar	A/c = Aij / c ($1 \le i \le m$, $1 \le j \le n$)
Potenciação de matrizes	A.^c =Aij .^ c ($1 \le i \le m$, $1 \le j \le n$) c.^A = c .^Aij ($1 \le i \le m$, $1 \le j \le n$) A.^B = Aij .^Bij ($1 \le i \le m$, $1 \le j \le n$)
Adição e subtraçao de matrizes (A e B têm que possuir o mesmo número de linhas e de colunas).	A+B = Aij + Bij ($1 \le i \le m$, $1 \le j \le n$) A-B = Aij - Bij ($1 \le i \le m$, $1 \le j \le n$)
Multiplicação de matrizes elemento por	A.*B = Aij * Bij ($1 \le i \le m$, $1 \le j \le n$)

elemento (A e B têm que possuir o mesmo número de linhas e de colunas)	
Multiplicação de matrizes (a multiplicação de A e B é definida somente quando o número de linhas de B é exatamente igual ao número de colunas de A). A (m x p) e B (p x n)	A*B = Ai1B1j + Ai2B2j + ... + AipBpj ($1 \leq i \leq m$, $1 \leq j \leq n$)
Divisão direita de matrizes elemento por elemento (A e B têm que possuir o mesmo número de linhas e de colunas)	A./B = Aij / Bij ($1 \leq i \leq m$, $1 \leq j \leq n$)
Divisão esquerda de matrizes elemento por elemento (A e B têm que possuir o mesmo número de linhas e de colunas)	A.\B = Bij / Aij ($1 \leq i \leq m$, $1 \leq j \leq n$)
Transposta de uma matriz	A' = Aij' = Aji ($1 \leq i \leq m$, $1 \leq j \leq n$)

2.7.8 Manipulação, Comparação e Dimensão de Vetores e Matrizes

Já que as matrizes são de importância fundamental para o MATLAB, há diversas maneiras de manipulá-las no MATLAB. Uma vez formadas as matrizes, o MATLAB provê maneiras poderosas de inserir, extrair e rearranjar subvetores delas identificando os subscritos de interesse. O conhecimento dessas características é fundamental para o uso eficiente do MATLAB. Para ilustrar as características de manipulação de matrizes e vetores de MATLAB, vejamos os seguintes exemplos:

```
>> A=[1 2 3;4 5 6;7 8 9]
   A =
        1    2    3
        4    5    6
        7    8    9
>> A(3,3)=0
   A =
        1    2    3
        4    5    6
        7    8    0
```

muda o elemento na terceira linha e terceira coluna para zero.

```
>>A(2, 6)=1
  A =
        1    2    3    0    0    0
```

```
    4    5    6    0    0    1
    7    8    0    0    0    0
```

coloca um 1 na segunda linha e na sexta coluna. Uma vez que a matriz A não tem seis colunas, o tamanho de A é aumentado conforme necessário e A é preenchida com zeros fazendo com que A continue retangular.

```
>> A=[1 2 3;4 5 6;7 8 9];
>> B=A(3:-1:1,1:3)
   B =
        7    8    9
        4    5    6
        1    2    3
```

cria uma matriz B colocando as linhas de A em ordem inversa.

```
>> B=A(3:-1:1,:)
   B =
      7    8    9
      4    5    6
      1    2    3
```

faz o mesmo que no exemplo anterior. Nesse caso, os dois pontos no final significam tomar todas as colunas. Isto é, : é uma forma reduzida de 1:3 nesse exemplo, já que A tem três colunas.

```
>>C=[A B(:, [1 3 ])]
C =
      1    2    3    7    9
      4    5    6    4    6
      7    8    9    1    3
```

cria C, anexando todas as linhas na primeira e terceira colunas de B, à direita de A.

```
>>B=A(1:2,2:3)
B =
      2    3
      5    6
```

cria B extraindo as duas primeiras linhas e as duas últimas colunas de A.

```
>>C=[1 3]
C =
      1    3
>>B=A(C,C)
```

```
B =
     1   3
     7   9
```

usa o vetor C para selecionar os índices de A, em lugar de especificar estes diretamente usando a notação início:incremento:fim ou início:fim.

Nesse exemplo, B é formada a partir de A., extraindo desta a primeira e a terceira linhas e as colunas de mesmos índices.

```
B=A(:)
B =
     1
     4
     7
     2
     5
     8
     3
     6
     9
```

monta o vetor B armazenando as colunas de A, uma por vez e em ordem, em um único vetor.

```
>>B=B.'
B =
     1   4   7   2   5   8   3   6   9
```

ilustra o uso da operação de transposição pontuada anteriormente.

```
>>B=A
B =
     1   2   3
     4   5   6
     7   8   9
>>B(:,2)=[ ]
B =
     1   3
     4   6
     7   9
```

redefine B excluindo todos os elementos da segunda coluna da matriz original. Quando você atribui a matriz vazia [ ] a alguma coisa, esta é eliminada, fazendo com que a matriz seja reduzida aos elementos remanescentes. Note que você

precisa eliminar linhas ou colunas inteiras, de modo que a matriz resultante permaneça retangular.

```
>>B=B.'
B =
     1   4   7
     3   6   9
```

ilustra a transposição de uma matriz. De forma geral, a iésima linha torna-se a iésima coluna da matriz resultante, de maneira que a matriz que, originalmente, era 3 por 2 torna-se 2 por 3.

```
>>B(2,:) = [ ]
B =
     1   4   7
```

elimina a segunda linha de B.

```
>>A(2,:) = B
A =
     1   2   3
     1   4   7
     7   8   9
```

substitui por B a segunda linha de A.

```
>>B=A(:,[2  2  2  2])
B =
     2   2   2   2
     4   4   4   4
     8   8   8   8
```

cria B copiando quatro vezes a segunda coluna de A.

```
>>A       %mostrando A novamente
A =
     1   2   3
     1   4   7
     7   8   9

>>A(2,2) = [ ]
??? Indexed empty matrix assigment is not allowed.
```

mostra que você só pode eliminar linhas e colunas inteiras de uma matriz. O MATLAB não sabe como reduzir uma matriz da qual foi eliminada apenas uma parte das linhas ou colunas.

```
>>B=A(4, :)
??? Index exceeds matrix dimensions.
```

Nesse exemplo, como não existe a quarta linha de A, o MATLAB não sabe o que fazer, o que é indicado na tela.

```
>>B(1:2,:) = A
??? In an assignment A(matrix, :) = B, the number of columns in A and B must be the same.
```

Esse exemplo mostra que você não pode inserir uma matriz em outra se as dimensões forem diferentes.

```
>>B=[1  4  7];
>>B(3:4,:)=A(2:3,:)
B =
        1    4    7
        0    0    0
        1    4    7
        7    8    9
```

Entretanto você pode copiar a segunda e a terceira colunas de A para uma região de mesma dimensão em B. Uma vez que B não contém linhas com índices iguais a 2, 3 e 4, estas são criadas. Além disso, como a segunda linha de B não foi definida, ela é preenchida com zeros.

```
>>G(1:6)=A(:,2:3)
G =
        2    4    8    3    7    9
```

cria um vetor linha G extraindo todos os elementos da segunda e da terceira colunas de A. Note que as dimensões das matrizes que aparecem em ambos os lados do sinal de igualdade são diferentes.

Quando o termo do lado direito de um comando de atribuição é um escalar e o termo do lado esquerdo é uma matriz, o escalar é expandido. Por exemplo:

```
>>A(2,:)=0
A =
        1    2    3
        0    0    0
        7    8    9
```

substitui a segunda linha de A por zeros. O zero que aparece do lado direito, que é único, é expandido de modo a preencher todos os elementos referentes aos índices especificados à esquerda. Esse exemplo é equivalente a:

```
>>A(2,:)=[0  0  0]
A =
        1    2    3
        0    0    0
        7    8    9
```

Em alguns casos, é mais conveniente fazer referência aos elementos de uma matriz usando um único índice. Quando apenas um índice é usado no MATLAB, este índice é contado percorrendo as colunas de cima para baixo, começando pela primeira coluna. Por exemplo:

```
>>D=[1  2  3  4; 5  6  7  8; 9  10  11  12]   % novos dados
D =
        1    2    3    4
        5    6    7    8
        9  10  11  12
>>D(2)       % segundo elemento
ans =
        5
>>D(5)       %quinto elemento (contando 3 na 1ª coluna e 2 na 2ª)
ans =
        6
>>D(end)   % último elemento da matriz
ans =
        12
>>D(4:7)     % do quarto ao sétimo elemento
ans =
        2    6    10    3
```

Além de fazer referência aos elementos das matrizes usando seus índices, também podemos usar vetores lógicos, resultantes de operações lógicas (que serão descritas de forma detalhada mais tarde), desde que o tamanho do vetor lógico seja igual ao tamanho da matriz à qual ele faz referência. Neste caso, os elementos que contêm o valor Verdadeiro (1) são mantidos, enquanto os elementos iguais a Falso (0) são descartados.

```
>>x = -3:3   % criando alguns dados
x =
```

```
        -3   -2   -1   0   1   2   3
>>abs(x) > 1
ans =
        1   1   0   0   0   1   1
```

retorna um vetor lógico que tem elementos iguais a um nas posições correspondentes às componentes de x que são maiores que 1.

```
>>y = x(abs(x)>1)
y = -3   -2   2   3
```

cria y extraindo de x aqueles componentes com valor absoluto maior que 1. Entretanto, deve-se notar que o comando:

```
>>y=x([1  1  0  0  0  1  1])
??? Index into matrix is negative or zero. See release notes on changes to logical índices.
```

contém um erro mesmo que abs(x) > 1 e [1 1 0 0 0 1 1] pareçam, à primeira vista, gerar o mesmo vetor. Isto se deve ao fato de o vetor [1 1 0 0 0 1 1] ser, no segundo exemplo, um vetor numérico e não lógico. Assim, o MATLAB tenta encontrar os elementos cujos índices pertencem ao vetor [1 1 0 0 0 1 1] e o erro ocorre porque não há elementos com índice 0. Naturalmente, o MATLAB possui uma função, denominada logical, para converter vetores numéricos em vetores lógicos.

```
>>y=x(logical([1  1  0  0  0  1  1]))
y =
        -3   -2   2   3
```

Agora obtivemos, mais uma vez, o resultado desejado. Vetores lógicos são um tipo especial de vetores de precisão dupla no MATLAB. Até aqui, só consideremos vetores numéricos. Ao especificar índices de vetores usando vetores numéricos, extraímos aqueles elementos que possuem os índices indicados. Por outro lado, ao especificar índices de vetores usando vetores lógicos, obtidos como resultado de operações lógicas ou usando o comando logical, extraímos os elementos associados ao termo Verdadeiro ($\neq 0$).

Tal como acontece com vetores, os vetores lógicos também podem ser aplicados a matrizes.

```
>>B=[5  -3; 2  -4]
B =
        5   -3
        2   -4
>>x = abs(B)>2
```

```
x =
     1   1
     0   1
```

Da mesma forma, a extração de elementos usando vetores lógicos também se aplica a matrizes.

```
>>y=B(x)
y =
     5
    -3
    -4
```

Entretanto, os resultados são transformados em vetores colunas, uma vez que não é possível definir uma matriz com apenas três elementos.

Muitas vezes, deseja-se saber quais são os índices dos elementos de uma matriz ou vetor que satisfazem alguma expressão. No MATLAB, isso é obtido usando-se a função *find*, que fornece os índices para os quais uma expressão é verdadeira:

```
>>x = -3:3
x =
     -3   -2   -1   0   1   2   3
>>k = find(abs(x)>1)
k =
     1   2   6   7
```

encontra os indices para os quais abs(x) >1.

```
>>y = x(k)
y =
     -3   -2   2   3
```

cria y usando os índices fornecidos por x.

A função find também pode ser usada com matrizes:

```
>> A=[1 2 3;4 5 6;7 8 9]
 A =
        1   2   3
        4   5   6
        7   8   9
>>[l, c] = find(A>5)
l =
     3
     3
     2
```

```
        3
c =
        1
        2
        3
        3
```

Nesse caso, os indices armazenados nos vetores I e j correspondem respectivamente, às linhas e colunas dos elementos que satisfazem a expressão. Assim, A(i(1), j(1)) é o primeiro elemento de A para o qual A>5 (a busca é feita de coluna em coluna), e assim por diante.

Note que, nos casos em que uma função do MATLAB retorna duas ou mais variáveis, estas aparecem entre colchetes, do lado esquerdo do sinal de igualdade. Esta sintaxe é diferente daquela usada na manipulação de matrizes e vetores mencionada anteriormente, segundo a qual [i, j], do lado direito do sinal de igualdade, produz uma nova matriz anexando j à direita de i.

Em certas situações, pode ser conveniente comparar dois vetores ou duas matrizes. Por exemplo:

```
>> A=[1 2 3;4 5 6;7 8 9]'     % Transposta da matriz anterior
 A =
          1    4    7
          2    5    8
          3    6    9
>>B=A.*(-1).^A                 % uma matriz próxima de A
B =
        -1    4   -7
         2   -5    8
        -3    6   -9
>>C=1:9                        % um vetor com os mesmos valores de A
C =
        1   2   3   4   5   6   7   8   9
>>isequal(A, C)
ans =
        0
>>isequal(A, B)
ans =
        0
>>isequal(A, A)
```

```
ans =
        1
>>isequal(C, C')
ans =
        0
```

A função isequal retorna Verdadeiro (1) quando dois vetores ou matrizes têm as mesmas dimensões e elementos idênticos. Caso contrário, retorna Falso (0).

A função ismember identifica elementos que são idênticos em dois vetores ou em duas matrizes:

```
>>ismember(A, B)              % observe que o resultado é um vetor coluna
ans =

        0    1    0
        1    0    1
        0    1    0

>>ismember(A, C)
ans =
        1    1    1
        1    1    1
        1    1    1
```

ismember retorna Verdadeiro(1) para aqueles elementos de A que também aparecem no vetor fornecido como seu segundo argumento. Os dois argumentos não precisam ter a mesma dimensão:

```
>>x=0:2:20              %um vetor com 11 elementos
X =
        0    2    4    6    8    10    12    14    16    18    20
>>ismember(x, A)
ans =
        0    1    1    1    1    0     0     0     0     0     0
```

Esse é um vetor com a mesma dimensão de x, contendo Verdadeiro (1) nas posições correspondentes aos elementos comuns.

```
>>ismember (A, x)
ans =
        0    1    0
        1    0    1
```

```
    0     1     0
```

Esse é um vetor com o mesmo número de elementos de A, no qual Verdadeiro (1) aparece nas posições em que há elementos comuns. Assim ismember compara seu primeiro argumento com o segundo e retorna um vetor com o mesmo número de elementos do primeiro.

Transformar a matriz A e B em vetores coluna:

```
>>A = A(:)
>>B = B(:)
```

Outros conjuntos de funções do MATLAB incluem (na versão 6.5 do Matlab, as funções union e intersect funcionam para vetores):

```
>>union(A, B)      %a união de todos os elementos, em ordem crescente
ans =
     -9
     -7
     -5
     -3
     -1
      1
      2
      3
      4
      5
      6
      7
      8
      9

>>intersect(A, B)      % a interseção das duas matrizes, em ordem crescente
ans =
      2
      4
      6
      8

>>setdiff(A, B)      % os valores de A que não estão em B, em ordem crescente
```

```
ans =
     1
     3
     5
     7
     9

>>setxor(A, B)        % os valores que não estão na interseção das duas matrizes,
                      % em ordem crescente
ans =
    -9
    -7
    -5
    -3
    -1
     1
     3
     5
     7
     9
```

Essas funções estão resumidas na seguinte tabela:

Comparações entre Vetores	
isequal (A, B)	Variável lógica: Verdadeira se A e B idênticos.
ismember (A, B)	Variável lógica: Verdadeira quando os elementos de A são também elementos de B.
intersect (A, B)	Valores na interseção de A e B.
setdiff (A, B)	Valores de A que não estão em B.
setxor (A, B)	Valores que não estão na interseção de A e B.
union (A, B)	Valores na união de A e B.

Nos casos em que o tamanho de uma matriz, embora desconhecido, é necessário para algum tipo de manipulação, o MATLAB fornece duas funções úteis, *size* e *length*:

```
>>A=[1  2  3  4; 5  6  7  8]
```

```
A =
        1   2   3   4
        5   6   7   8
  >>s=size(A)
  s =
        2   4
```

Quando apenas um argumento de saída é definido, a função size retorna um vetor linha cujo primeiro elemento é o número de linhas e cujo segundo elemento é o número de colunas.

```
  >>[l, c] = size(A)
  l =
        2
  c =
        4
```

Quando há dois argumentos de saída, a função size retorna o número de linhas na primeira variável e o número de colunas na segunda.

```
  >>l = size(A, 1)          % número de linhas
  l  =
        2
  >>c=size(A, 2)            % número de colunas
  c  =
        4
```

Chamada com dois argumentos, a função size retorna o número de linhas ou o número de colunas.

```
  >>length(A)
  ans =
        4
```

Retorna o número de linhas ou o número de colunas (o que for maior).

```
  >>B = pi:0.01:2*pi;
  size(B)
  ans =
        1   315
```

Mostra que B é um vetor linha e:

```
  >>length(B)
  ans =
        315
```

retorna o comprimento do vetor.

```
>>size([ ])
ans =
        0    0
```

Mostra que, de fato, uma matriz vazia tem dimensão zero, como realmente deve ser.

Esses conceitos estão resumidos na tabela a seguir:

Dimensões de Vetores e Matrizes	
whos	Mostra as variáveis que existem no espaço de trabalho e suas dimensões.
s=size(A)	Retorna um vetor linha s, cujo primeiro elemento é o número de linhas de A e cujo segundo elemento é o número de colunas de A.
[l, c]=size(A)	Retorna dois escalares l e c contendo, respectivamente, o número de linhas e o número de colunas de A.
l=size(A, 1)	Retorna o número de linhas de A na variável l.
c=size(A, 2)	Retorna o número de colunas de A na variável c.
n=length(A)	Retorna max(size(A)) na variável n quando A não é vazia.

2.7.9 Sistemas de Equações Lineares

O MATLAB foi escrito para simplificar os cálculos matriciais e de álgebra linear que aparecem em muitas aplicações. Um dos problemas mais comuns de álgebra linear consiste na solução de um sistema de equações lineares. Considere, por exemplo, o seguinte sistema de equações:

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{vmatrix} \begin{vmatrix} x1 \\ x2 \\ x3 \end{vmatrix} = \begin{vmatrix} 366 \\ 804 \\ 351 \end{vmatrix}$$

$$A \cdot x = b$$

no qual o símbolo matemático de multiplicação (**.**) é usado no sentido matricial, em oposição ao sentido vetorial definido anteriormente. No MATLAB, esta multiplicação matricial é representada por meio do asterisco *****. As equações anteriores definem o produto entre a matriz A e o vetor x como sendo igual ao vetor b. A existência de soluções para a equação anterior é um problema fundamental da álgebra linear. Além disso, quando uma solução existe, há diversas maneiras de encontra-la, como o processo de eliminação de Gauss, a

fatoração LU ou o uso direto da matriz inversa de A. Mostraremos como o MATLAB pode ser usado para resolver problemas como o exemplo apresentado.

Para resolver esse problema, é necessário fornecer A e b.

```
>>A=[1  2  3;4  5  6;7  8  0]
A =
        1    2    3
        4    5    6
        7    8    0
>>b=[366; 804; 351]
b =
        366
        804
        351
```

Como foi discutido anteriormente, a forma pela qual a matriz A foi definida acima mostra as duas maneiras usadas pelo MATLAB para distinguir as linhas. O ponto-e-vírgula entre os números 3 e 4 simboliza o começo de uma nova linha, assim como a mudança de linhas entre os números 6 e 7. b é um vetor coluna porque cada ponto-e-vírgula representa o começo de uma nova linha.

Para aqueles que têm algum conhecimento de álgebra linear, é fácil mostrar que esse problema tem solução única se o determinante da matriz A for diferente de zero:

```
>>det(A)
ans =
        27
```

sendo isso verdade, há duas formas de usar o MATLAB para determinar a solução de A . x = b, sendo uma delas mais recomendável. A maneira menos indicada, mas mais direta, consiste em calcular $x = A^{-1}$. b explicitamente:

```
>>inv(A)*b
x =
25.0000
22.0000
99.0000
```

Aqui, inv(A) é uma função do MATLAB que calcula a inversa da matriz A e o operador *, não estando precedido por um ponto, representa a multiplicação de matrizes. A maneira mais recomendável de resolver o sistema consiste em usar o operador matricial de divisão à esquerda.

```
>>x=A\b
```

```
x =
        25.0000
        22.0000
        99.0000
```

Nesse caso, o MATLAB utiliza a fatoração LU para resolver o sistema, sendo a solução representada pela divisão à esquerda de A por b. O operador de divisão à esquerda, \, não é precedido por um ponto, uma vez que a operação é matricial, e não apenas entre os elementos com os mesmos índices, de dois vetores.

Há diversas razões pelas quais essa segunda maneira de resolver um sistema linear é a mais indicada. A mais óbvia delas é que esse segundo método requer menos multiplicações e divisões e, consequentemente, é mais rápido. Além disso, a solução é geralmente mais precisa se o problema for grande.

Em ambos os casos, se o MATLAB não puder achar uma solução ou não puder acha-la com a precisão desejada, uma mensagem de erro é exibida na tela.

Se você for um bom estudante de álgebra linear, deve saber que, quando o número de variáveis e o número de equações são diferentes, normalmente não há uma solução única para o sistema. Entretanto, introduzindo algumas restrições adicionais, pode-se encontrar uma solução com alguma utilidade prática.

No caso de haver, mesmo depois de retiradas todas as equações redundantes de um sistema, mais equações do que incógnitas, isto é, no caso de o sistema ser sobredeterminado, os operadores de divisão do MATLAB, / e \ podem ser usados para encontrar a solução que minimiza o quadrado da norma do resíduo definido por A . x – b. Essa solução é muito útil na prática, sendo conhecida como solução de quadrados mínimos. Considere, por exemplo, o seguinte sistema:

```
>>A=[1  2  3;4  5  6;7  8  0; 2  5  8]    % 4 equações e 3 incógnitas.
A =
        1    2    3
        4    5    6
        7    8    0
        2    5    8
>>b=[366; 804; 351; 514]            % um novo vetor do lado direito
b =
        366
        804
        351
        514
>>x = A\b     % obtendo a solução de quadrados mínimos
```

```
x =
        247.9818
       -173.1091
        114.9273
>>res = A*x-b     % calculando o resíduo de norma mínima
res =
       -119.4545
         11.9455
          0.0000
         35.8364
```

Por outro lado, quando há menos equações do que incógnitas, isto é, no caso de o sistema ser indeterminado, o número de soluções é infinito. Dentre essas soluções, duas são facilmente obtidas com o MATLAB. Quando o operador de divisão é empregado, obtém-se o vetor solução x que tem o maior número de elementos nulos. Por outro lado, usando x=pinv(A) * b obtém-se uma solução tal que o comprimento ou norma de x é o menor possível. Essa solução, baseada na pseudoinversa, também tem grande valor prático, sendo denominada solução de norma mínima. Observe o exemplo abaixo:

```
>>A=A'              % definindo 3 equações e 4 incógnitas
A =
        1   4   7   2
        2   5   8   5
        3   6   0   8
>>b=b(1:3)          % um novo vetor b
b =
        366
        804
        351
>>x=A\b             % encontrando a solução com o maior número de zeros.
X =
                0
       -165.9000
         99.0000
        168.3000
>>xn = pinv(A)*b    % encontrando a solução de norma mínima
xn =
         30.8182
```

```
        -168.9818
          99.0000
         159.0545
>>norm(x)          % calculando a norma Euclidiana de x.
ans =
        256.2200
>>norm(xn)         %verificando que xn tem norma menor que x.
ans =
        254.1731
```

Exemplo

1 – Escreva os comandos no Matlab para resolver o seguinte problema: Pedro comprou dois chicletes, três balas e um pirulito por R$5,00; Joaquim comprou três chicletes, duas balas e dois pirulitos por R$6,00 e Maria comprou um chicletes, duas balas e dois pirulitos por R$4,00. Descubra o preço de cada chicletes, cada bala e cada pirulito. Considerar chicletes como variável C, bala como variável B e pirulito como variável P.

Solução:

```
>> A = [2 3 1; 3 2 2; 1 2 2]
>> B = [5; 6; 4]
>> X = A \ B          % Primeira opção
>> X = inv (A) * B    % Segunda opção
```

Resposta:

Chicletes: R$1,00
Bala: R$0,75
Pirulito: R$0,75

Exercícios

1 – Resolva os sistemas abaixo:

A – $2x - y = 15$
$x + 3y = 25$

B – $x + y + 2z = -1$
$4x + y + 4z = -2$
$2x - y + 2z = -4$

2 – Resolva os problemas abaixo:

A – Uma escola de ensino médio tem 107 alunos nas 1ª e 2ª séries, 74 nas 2ª e 3ª séries e 91 nas 1ª e 3ª séries. Qual o total de alunos dessa escola?

B – Num concurso público, foram realizadas três provas, com 10 questões em cada uma. Cada questão valia um ponto, mas os pesos x, y e z das provas, nessa ordem, eram diferentes. O primeiro classificado no concurso, que acertou 8 questões na primeria prova; 9, na segunda, e 10, na terceira, obteve, no final, um total de 93 pontos. O segundo classificado acertou, nessa mesma ordem, 9, 9 e 7 questões, totalizando 80 pontos. O terceiro classificado acertou 8, 8 e 7 questões, respectivamente, atingindo a soma de 75 pontos no final. Calcule os pesos x, y e z.

2.7.10 Matrizes Especiais

O MATLAB contém diversas matrizes especiais. Algumas delas são de uso geral, enquanto outras são matrizes voltadas para aplicações especializadas. As matrizes de uso geral incluem:

```
>>a = [1  2  3;4  5  6]
>>b = find (a >= 10)
b =
        Empty matrix: 0-by-1
```

Essa é a matriz vazia. O MATLAB retorna a matriz vazia como resposta a uma operação que não tem resultado. No exemplo acima, a matriz a não possui elementos maiores que 10. Uma matriz vazia tem tamanho igual a zero, mas o nome da variável a ela associado existe no espaço de trabalho do MATLAB.

```
>>zeros(3)              % uma matriz 3x3 de zeros
ans =
        0   0   0
        0   0   0
        0   0   0
>>ones(2 ,4)           % uma matriz 2x4 com elementos iguais a 1.
ans =
        1   1   1   1
        1   1   1   1
>>zeros(3)+pi
ans =
```

```
        3.1416    3.1416    3.1416
        3.1416    3.1416    3.1416
        3.1416    3.1416    3.1416
```

um exemplo no qual é criada uma matriz 3 por 3 com todos os elementos iguais a pi.

```
>>rand(3, 1)
ans =
        0.9501
        0.2311
        0.6068
```

uma matriz 3 por 1 de números aleatórios uniformemente distribuídos entre zero e um.

```
>>randn(2)
ans =
       -0.4326    0.1253
       -1.6656    0.2877
```

uma matriz 2 por 2 de números aleatórios que seguem a distribuição normal, com média zero e variância igual a um.

```
>>eye(3)
ans =
        1    0    0
        0    1    0
        0    0    1
```

a matriz identidade 3 por 3.

```
>>eye(3,2)
ans =
        1    0
        0    1
        0    0
```

a matriz identidade 3 por 2.

Além de poder especificar o tamanho da matriz explicitamente, você também pode usar a função size para criar uma matriz com a mesma dimensão que outra:

```
>>A = [1  2  3;4  5  6];
>>ones(size(A))
ans =
        1    1    1
        1    1    1
```

uma matriz com todos os elementos iguais a um e a mesma dimensão de A.

2.8 Arquivos de Instrução

Em problemas mais simples, é mais rápido e eficiente introduzir seus pedidos no prompt do MATLAB. Entretanto, à medida que aumenta o número de comandos e nos casos em que você desejar mudar o valor de uma ou mais variáveis ou reexecutar separadamente alguns comandos, pode tornar-se tedioso introduzir os comandos no prompt. O MATLAB apresenta uma solução lógica para esse problema. Ele lhe permite colocar os comandos MATLAB em um arquivo de texto simples e depois informar ao MATLAB para abrir esse arquivo e executar os comandos exatamente como se você os tivesse introduzido manualmente no prompt do MATLAB. Esses arquivos são chamados arquivos de instrução ou arquivos M. O termo instrução representa o fato de o MATLAB simplesmente seguir as instruções contidas no arquivo. O termo arquivo M refere-se ao fato de que os nomes dos arquivos de instrução têm de terminar com a extensão 'm', por exemplo, teste.m.

Para criar um arquivo M, selecione *New* no menu *File* e selecione *M-file*. Esse procedimento abre uma janela de editor de texto a partir da qual colocará os comandos MATLAB.

Depois de armazenar esse arquivo M com o nome exemplo.m em seu disco, o MATLAB executará os comandos contidos em exemplo.m depois que você digitar exemplo no prompt do MATLAB.

```
>>exemplo
```

Quando o MATLAB interpreta esse comando, prioriza as variáveis MATLAB atuais e os comandos preexistentes em relação aos nomes dos arquivos M. Portanto, se a palavra exemplo não for uma variável atual MATLAB, nem um comando MATLAB (o que realmente não é), o MATLAB abre o arquivo exemplo.m (se ele puder localizá-lo) e executa os comandos lá encontrados exatamente como se eles tivessem sido introduzidos no prompt da janela de comandos. Como resultado, os comandos dentro do arquivo M têm acesso a todas as variáveis no espaço de trabalho. Normalmente, os comandos lidos do arquivo M não são visualizados à medida que são executados. O comando *echo on* informa ao MATLAB para mostrar ou ecoar os comandos na janela de comandos à medida que eles são lidos e executados. Você com certeza já pode adivinhar o que o comando *echo off* faz.

Para abrir o arquivo M é só usar o item *Open M-file* no menu *File* para se fazer alguma alteração caso seja necessário.

Em virtude da grande utilidade dos arquivos de comandos, o MATLAB possui diversas funções que são particularmente apropriadas para o uso em arquivos M. Essas funções são:

Funções dos Arquivos M	
disp (ans)	Mostra os resultados sem identificar os nomes das variáveis.
echo	Controla a exibição dos comandos dos arquivos M na janela de *comandos*. Existe *echo on* e *echo off*.
input	Solicita ao usuário que forneça algum dado de entrada.
keyboard	Transfere temporariamente o controle para o teclado (digite *dbcont* para sair).
pause	Suspende a execução até que o usuário pressione alguma tecla.
pause(n)	Suspende a execução por n segundos.
waitforbuttonpress	Suspende a execução até que o usuário pressione uma tecla ou um botão do mouse.

Quando um comando do MATLAB não é seguido de um ponto-e-vírgula, os resultados gerados pelo comando são exibidos na janela de comandos e o nome da variável é identificado. Para obter uma saída de dados mais bonita, convém, muitas vezes, suprimir o nome da variável. Isso pode ser feito no MATLAB por meio do comando *disp*. Veja exemplo abaixo:

```
>>con_inicial = 90              %forma tradicional de mostrar um resultado
con_inicial =
        90
>>disp(con_inicial)    %mostrando o resultado sem o nome da variável
        90
```

Para evitar que você tenha de editar várias vezes um mesmo arquivo de comandos quando é preciso repetir os cálculos para diversos casos diferentes, o comando *input* permite que você solicite um parâmetro durante a execução de um arquivo de comandos. Por exemplo, considere o arquivo exemplo.m:

```
% Arquivo exemplo.m relativo a cálculo de banho ácido
con_inicial = 90
con_min = 50
perda = input('Porcentagem de água adicionada a cada imersão >')/100
n = floor(log2(con_inicial/con_min)/log2(1+perda))
```

Executando esse arquivo de comandos, obtemos:

```
>>exemplo
Porcentagem de água adicionada a cada imersão >5
perda =
   0.0500
n =
   12
```

Em resposta ao pedido para que a quantidade de água adicionada fosse definida, o número 5 foi fornecido e, em seguida, a tecla *Return* ou *Enter* foi pressionada. Os comandos subsequentes foram calculados usando-se perda=5/100. Note que a função *input* pode ser misturada com outras operações, tal como as demais funções comuns. Ao invés de fornecermos um número como entrada, podemos fornecer uma expressão, por exemplo, no lugar do 5 podemos entrar com round(sqrt(13))+3. Teremos como resultado perda = 0.0700 e n = 8.

O MATLAB apresenta diversos comandos de gerenciamento de arquivos que lhe permitem listar nomes de arquivos, visualizar e excluir arquivos M, mostrar e mudar o diretório atual. A tabela a seguir apresenta um resumo desses comandos:

Comando	**Descrição**
what	Retorna uma listagem de todos os arquivos M do diretório atual.
dir	Lista todos os arquivos do diretório atual.
ls	O mesmo que dir.
type exemplo	Mostra o arquivo M exemplo.m na janela de comandos.
delete exemplo	Exclui o arquivo M exemplo.m.
cd caminho	Muda o diretório dado por caminho.
chdir caminho	O mesmo que cd caminho.
cd	Mostra o diretório de trabalho atual.
chdir	O mesmo que cd.
pwd	O mesmo que cd.
which exemplo	Mostra o caminho para o diretório exemplo.m.

2.8.1 Armazenamento e Recuperação de Dados

Além de guardar variáveis, o MATLAB pode armazenar e recuperar dados de arquivos de seu computador. O item Save Workspace as... do menu File abre uma caixa de diálogo para permitir que você grave em um arquivo todas as variáveis atuais. De forma semelhante, o item de menu Load Workspace as... do menu File abre uma caixa de diálogo para permitir que você recupere variáveis de uma área de trabalho previamente armazenada. O armazenamento de variáveis não as exclui do espaço de trabalho do MATLAB. A recuperação de variáveis de mesmo nome daquelas que já se encontram no espaço de trabalho altera seus valores para aqueles contidos no arquivo.

Se os comandos do menu File não satisfizerem suas necessidades, o MATLAB possui dois comandos, save e load, que apresentam maior flexibilidade. Em particular, o comando save permite que você armazene uma ou mais variáveis no formato de arquivo de sua escolha. Por exemplo:

```
>> save
```

armazena todas as variáveis no formato binário do MATLAB no arquivo matlab.mat.

```
>>save teste1
```

armazena todas as variáveis no formato binário no MATLAB no arquivo teste1.

```
>>save teste1 a b
```

armazena as variáveis a e b, em formato binário, no arquivo teste1.mat.

```
>>save teste1 a b –ascii
```

armazena as variáveis a e b no formato ASCII de 8 dígitos no arquivo teste1.

O comando load usa a mesma sintaxe, com a diferença óbvia de que ele é empregado para recuperar variáveis no espaço de trabalho do MATLAB.

2.9 A Tomada de Decisões: Controle de Fluxo

O controle de fluxo é um recurso extremamente poderoso, pois permite que cálculos feitos anteriormente influenciem operações futuras. O MATLAB apresenta quatro estruturas de tomada de decisão ou controle de fluxo. Elas são: loops for, loops while, estruturas if-else-end e estruturas switch-case. Como essas estruturas frequentemente envolvem diversos comandos MATLAB, elas em geral aparecem em arquivos M, em vez de serem introduzidas diretamente no prompt do MATLAB. Exemplos dessas estruturas são mostrados nos Anexos de B até H.

2.9.1 Estruturas switch-case

Quando sequências de comandos devem ser condicionalmente executadas, com base no uso repetido de um teste de igualdade com um argumento comum, uma estrutura switch-case pode ser usada. Essa estrutura tem a forma:

```
switch expressão
    case teste_expressão1
            comandos1...
    case {teste_expr2, teste_expr3, teste_expr4}
            comandos2...
    otherwise
            comandos3...
end
```

No exemplo a seguir o valor de x é 10 e a variável unidade é inicializada com ‘m’ (metros). Temos três situações de conversão: polegadas, pés e metros, todas elas convertendo o valor de x para centímetros. Caso nenhuma das três opções satisfaça, então a mensagem ‘unidade desconhecida’ é emitida e o valor de y será nan (nenhum valor).

Exemplo: o programa converte metros para centímetro.

```
  x = 10;                  %valor de x
 unidades = 'm';           %unidade de teste metros
 switch unidades           %converte x para centímetros
       case 'pol'            %case polegadas
          y = x*2.54
       case 'p'              %case pés
          y = x*2.54*12
       case 'm'              %case metros
          y=x/100
       otherwise             %nenhum caso anterior
          disp(['Unidade desconhecida:' unidades])
          y=nan              %nenhum valor correspondente
 end
```

Executando esse exemplo, obtemos o valor final de y=0.1000.

Outro exemplo input:

```
x = input (‘Entre com o nome desejado (a, b, c):’, ‘s’)
switch x
    case ‘a’
        y = 20
    case ‘b’
        y = 30
    case ‘c’
        y = 40
end
```

Outro exemplo da estrutura switch-case (**Código retirado e adaptado do trabalho dos alunos Gabriel Wedson Mendonça de Souza e Rafael Nathan Pena de Souza - 2019**)

```
clc
clear all
opcao =input('Você quer saber "quais numeros acertou" ou "posicao do numero
no sorteio" ou se foi o "grande vencedor" ou deseja "sair" = ','s')
while(strcmp(opcao, 'sair') ~= 1)
    a = input(' Entre com os numeros que escolheu: ')
    b = input(' Entre com os números sorteados: ')
    switch opcao
      case 'grande vencedor'
          y = isequal(a,b)
          if y == 1
            disp ('$$$VOCÊ FOI O GRANDE VENCEDOR$$$')
            else disp (':( NÃO FOI DESTA VEZ ):')
          end
       case 'quais numeros acertou'
          y = ismember(a,b)
          r = a(y)
       case 'posicao do numero no sorteio'
          y = find(a==b)
       otherwise
          disp(['Opção desconhecida:' opcao])
          y = nan
    end
```

```
    opcao =input('Você quer saber "quais numeros acertou" ou "posicao do
numero no sorteio" ou se foi o "grande vencedor" ou deseja "sair" = ','s')
end
```

2.9.2 Estruturas if-else-end

Em diversas situações, as sequências de comandos têm de ser executadas condicionalmente, com base em um teste relacional. Nas linguagens de programação, essa lógica é implementada por meio de alguma variação da estrutura if-else-end. A estrutura if-else-end mais simples é:

```
if expressão
        comandos
end
```

Os comandos entre as instruções if e end são executados se todos os elementos na expressão forem Verdadeiro (diferentes de zero). Por exemplo:

```
>> macas=10;        %numero de maçãs
>> custo=macas*25;   %custo das maçãs
>> if macas>5        %dê um desconto de 20%
        custo=(1-20/100)*custo;
   end
>> custo
    custo =
              200
```

Nos casos em que há duas alternativas, a estrutura if-else-end fica:

```
if expressão
        comandos executados se Verdadeiro
else
        comandos executados se Falso
end
```

Aqui, o primeiro grupo de comandos é executado se a expressão for Verdadeiro; o segundo grupo de comandos é executado se a expressão for Falso.

Quando houver três ou mais alternativas, a estrutura if-else-end toma a forma:

```
if expressão1
        comandos executados se expressão1 for Verdadeiro
elseif expressão2
        comandos executados se expressão2 for Verdadeiro
```

```
elseif expressão3
        comandos executados se expressão3 for Verdadeiro
elseif expressão4
        comandos executados se expressão4 for Verdadeiro
elseif ...
   :
   :
else
        comandos executados se nenhuma outra expressão for Verdadeiro
end
```

Nessa última forma, somente os comandos associados à primeira expressão Verdadeiro encontrada são executados; as expressões relacionais seguintes não são testadas e o resto da estrutura if-else-end é saltada. Além disso, não é necessário que o comando final else esteja presente.

Agora que já sabemos como tomar decisões com as estruturas if-else-end, é possível mostrar uma maneira permissível de se saltar ou quebrar os loops for e while, mas não as estruturas if-else-end:

```
>> EPS=1;
for num=1:1000
        EPS=EPS/2;
        if(1+EPS) <=1
                EPS=EPS*2
                break
        end
 end
EPS =
         2.2204e-016
>> num
   num =
           53
```

Esse exemplo demonstra outra maneira de calcular eps. Nesse caso, faz-se com que o loop for seja executado um grande número de vezes. A estrutura if-else-end verifica então se EPS ficou suficientemente pequeno. Em caso afirmativo, EPS é multiplicado por dois e o comando break força o loop for a terminar prematuramente, isto é, em num = 53 nesse caso.

2.9.3 Loops while

Ao contrário do loop for, que executa um grupo de comandos um número fixo de vezes, o loop while executa um grupo de comandos um número indefinido de vezes.

A forma geral do loop while é:

```
while expressão
        comandos
end
```

Os comandos entre as instruções while e end são executados enquanto todos os elementos em expressão forem Verdadeiro. Por exemplo:

```
>> num=0;EPS=1;
>> while (1+EPS)>1
        EPS=EPS/2;
        num=num+1;
    end
>> num
   num =
         53
>> EPS=2*EPS
   EPS =
        2.2204e-016
```

Esse exemplo mostra uma maneira de calcular o valor especial MATLAB eps. Nesse caso, nós usamos EPS em maiúsculas para que o valor MATLAB não fosse substituído. Nesse exemplo, EPS começa em 1. Enquanto (1+EPS)>1 for Verdadeiro (diferente de zero), os comandos dentro do loop while serão executados. Uma vez que EPS é continuamente dividido por dois, às vezes ele se torna tão pequeno que, somando-se EPS a 1, não será maior do que 1. (Lembre-se de que isso acontece porque o computador usa um número fixo de dígitos para representar os números. O MATLAB usa dezesseis dígitos, de forma eps deve ser aproximadamente igual a 10^{-16}). Nessa altura, (1+EPS) > 1 torna-se Falso (zero) e o loop while termina. Finalmente, multiplica-se EPS por 2 porque a última divisão por 2 o tornou pequeno demais.

Reescrever o primeiro comando FOR do item 3.9.2 usando o comando WHILE.

```
>> n = 1;
    while  n <= 10
```

```
        x(n) = sin(n*pi/10);
        n = n + 1;
    end
>> x
x =
    Columns 1 through 6
        0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
    Columns 7 through 10
        0.8090    0.5878    0.3090    0.0000
```

Exemplo usando as estruturas switch-case, if-else-end e while (**Código retirado e adaptado do trabalho dos alunos Ana Cecília Rocha de Azevedo e Arthur Nunes Roberto Leite - 2019**).

```
clear all, clc
a=input('Digite o número de passagens desejadas: ')
while (a ~= 0)
  b=input('De uma a três refeições, quantas deseja ter durante o voo? ')
  if (a >= 1) & (a <= 250) & (b >= 1) & (b <= 3)
    disp('Indique a classe desejada')
    f=input('bussines, executiva, economica: ', 's')
    switch f
      case 'bussines'
        valor=a*b*900
        if valor < 50000
          disp('Neste valor, pagamento somente em cartão de débito. Os bilhetes eletrônicos poderão ser emitidos após a confirmação do pagamento. Agradecemos pela preferência!')
        else disp('Neste valor, o pagamento pode ser feito em cartão de crédito. Os bilhetes eletrônicos poderão ser emitidos após a confirmação do pagamento. Agradecemos pela preferência!')
        end
      case 'executiva'
        valor=a*b*600
        if valor < 30000
          disp('Neste valor, pagamento somente em cartão de débito. Os bilhetes eletrônicos poderão ser emitidos após a confirmação do pagamento. Agradecemos pela preferência!')
```

```
        else disp('Neste valor, o pagamento pode ser feito em cartão de crédito. Os bilhetes eletrônicos poderão ser emitidos após a confirmação do pagamento. Agradecemos pela preferência!')
       end
     case 'economica'
       valor=a*b*300
       if valor < 10000
        disp('Neste valor, pagamento somente em cartão de débito. Os bilhetes eletrônicos poderão ser emitidos após a confirmação do pagamento. Agradecemos pela preferência!')
        else disp('Neste valor, o pagamento pode ser feito em cartão de crédito. Os bilhetes eletrônicos poderão ser emitidos após a confirmação do pagamento. Agradecemos pela preferência!')
       end
     otherwise
       disp(['Classe não existente: ' f])
    end
  else disp ('Digite um número para as refeições e/ou passagens corretamente')
 end
 a=input('Digite o número de passagens desejadas: ')
end
```

2.9.4 Loops for

Os loops for possibilitam que uma série de comandos seja repetida por um número de vezes fixo e predefinido. A forma geral do loop for é

```
for x = conjunto
        comandos
end
```

Os comandos entre as instruções for e end são executados uma vez para cada coluna do conjunto. A cada iteração x é atribuído para a próxima coluna do conjunto, isto é, durante o i-ésimo ciclo do loop, temos que x = conjunto (:, i). Por exemplo:

```
>> for n=1:10
          x(n) = sin(n*pi/10);
     end
>> x
```

```
x =
    Columns 1 through 6
        0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
    Columns 7 through 10
        0.8090    0.5878    0.3090    0.0000
```

Em palavras, a primeira instrução diz para i igual a 1 a 10, calcule todas as instruções até a próxima instrução de end. No primeiro ciclo do for, n=1, no segundo, n=2 e assim por diante, até n=10. Depois do ciclo para n=10, o loop for termina e os comandos após a instrução end são executados, como é o caso da apresentação dos resultados em x. Outros aspectos importantes dos loops for são:

- Um loop for não pode ser terminado reatribuindo-se a variável de loop n dentro do loop for:

```
>> for n=1:10
        x(n) = sin(n*pi/10);
        n = 10;
    end
>> x
x =
    Columns 1 through 6
        0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
    Columns 7 through 10
        0.8090    0.5878    0.3090    0.0000
```

- A instrução 1:10 é uma instrução padrão de criação de conjuntos no MATLAB.
- Os loops for podem ser aninhados conforme desejado:

```
>> for n=1:3
        for m=3:-1:1
            A(n,m)=n^2+m^2;
        end
        disp(n)
    end
     1
     2
     3
>> A
A =
     2     5    10
```

```
     5     8    13
    10    13    18
```

- Os loops for devem ser evitados sempre que houver uma forma equivalente, por meio de conjuntos ou matrizes, para resolver um dado problema. Por exemplo, o primeiro exemplo pode ser reescrito como:

```
>>n=1:10;
>>x=sin(n*pi/10)
x =
    Columns 1 through 6
            0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
    Columns 7 through 10
            0.8090    0.5878    0.3090    0.0000
```

Embora ambas as formas conduzam ao mesmo resultado, a Segunda técnica tem execução mais rápida, é mais intuitiva e requer menos digitação.

Exemplo usando a estrutura for (**Código retirado do trabalho dos alunos Tiago Dutra Vidigal e Vinícius Rodrigues Magalhães Lopes - 2019**).

```
f = input ('Entre com o valor da força:')
x = input ('Entre com o valor do cosseno em graus:')
t = input('Entre com os valores de trabalho para calcular as distancias requeridas:')
for n = 1:length(t)
   D(n) = t(n)/(f*cosd(x));
end
D
plot (t,D)
title ('Distância em função do trabalho')
xlabel ('Trabalho')
ylabel ('Distância')
```

2.9.5 Exercícios

1 – Escreva um programa que pede para o usuário digitar um vetor linha com algarismos decimais. O programa deverá verificar se o que foi digitado é realmente um vetor linha e se os números são algarismos decimais, caso não seja, exibir mensagem de erro e verificar se o usuário deseja rodar o programa novamente. Caso afirmativo, separar os algarimos pares dos algarismos ímpares. Usando a função disp, exibir uma mensagem informando o número de algarismos

pares que foram digitados. O programa deverá ficar em loop até o usuário digitar “fim”.

2 – Repita o exercício anterior para uma matriz.

3 – Escreva um programa que pede para o usuário digitar um vetor linha com letras. O programa deverá verificar se o que foi digitado é realmente um vetor linha e se são letras do alfabeto, caso não seja, exibir mensagem de erro e verificar se o usuário deseja rodar o programa novamente. Caso afirmativo, separar as vogais das consoantes. Usando a função disp, exibir uma mensagem informando o número de consoantes e o número de vogais que foram digitadas. O programa deverá ficar em loop até o usuário digitar “fim”.

4 – Escreva um programa que pede para o usuário digitar um vetor linha com algarismos decimais. O programa deverá verificar se o que foi digitado é realmente um vetor linha e se os números são algarismos decimais, caso não seja, exibir mensagem de erro e verificar se o usuário deseja permanecer no programa. Caso afirmativo, usando loop for somar os números maiores que 4. Usar a função disp, para exibir uma mensagem com o resultado da soma. O programa deverá ficar em loop até o usuário digitar “parar”.

2.10 Gráficos

Gráficos constituem um recurso visual poderoso para interpretação de dados. Usando-se vetores, o MATLAB segue esse mesmo procedimento para a plotagem. Consideremos então a tarefa de plotar a função senoidal y = sin(x) para $0<=x<=2\pi$.

```
>>x = linspace(0, 2*pi, 30);
```

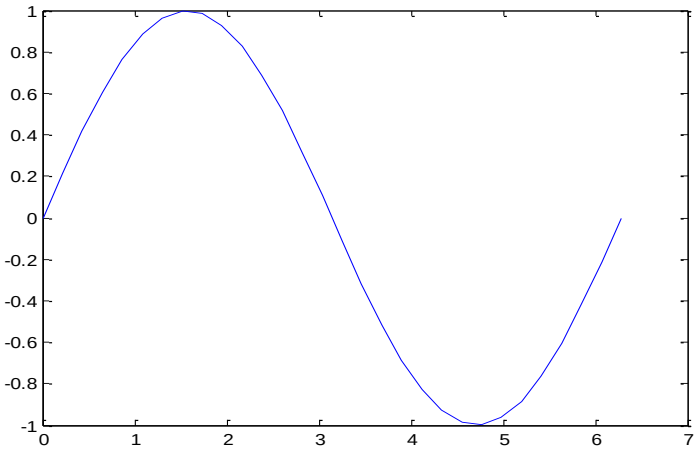
Cria 30 pontos entre 0 e 2π.

```
>>y = sin(x);
```

Calcula o seno dos pontos em x.

```
>>figure (1);
>>plot(x, y);
```

Gera o gráfico apresentado abaixo.



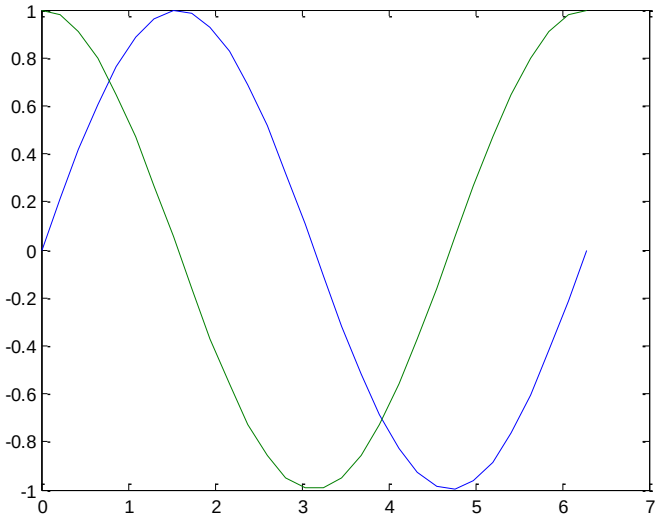
A função plot do MATLAB é extremamente poderosa. Ela automaticamente escolhe os limites dos eixos, marca os pontos individuais e desenha linhas retas entre eles. As opções no comando plot permitem-lhe plotar vetores múltiplos de dados nos mesmos eixos, usar tipos diferentes de linhas, tais como pontilhada ou tracejada, marcar somente os pontos de dados sem interliga-los, usar cores distintas para as diferentes curvas. Além disso, é possível colocar nomes nos eixos, um título na parte superior, desenhar uma grade nas marcas mais grossas, e daí por diante. Para ilustrar alguns desses recursos, consideremos os seguintes exemplos:

>>z = cos(x);

>>figure (2);

>>plot(x, y, x, z)

Obtém-se um gráfico do seno e do cosseno nos mesmos eixos.



>>figure (3);

>>plot(x, y, x, y, ’+’)

obtém-se o gráfico do seno duas vezes: a primeira com linhas interconectando os pontos de dados e a segunda marcando os pontos com o símbolo +.

>>figure (4);

>>plot(y, z)

um gráfico do seno versus o cosseno (forma uma circunferência).

>>figure (5);

>>plot(x, y, x, 2*y.*z, ‘--‘)

obtém-se uma ilustração da identidade 2senθcosθ = sen2θ. O gráfico de 2senθcosθ é plotado usando-se linha tracejada.

>>grid

coloca uma grade nas marcas grossas do gráfico atual.

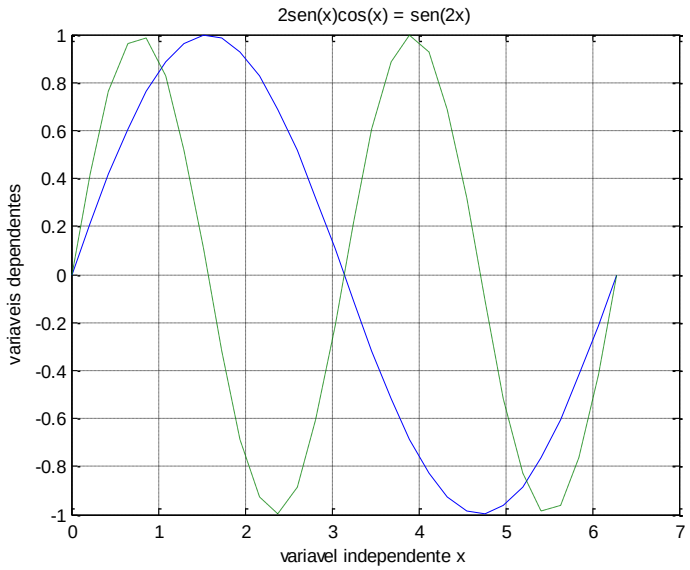
>>xlabel(‘variável independente x’)

>>ylabel(‘variaveis dependentes’)

coloca um nome no eixo dos x e dos y do gráfico atual.

>>title(‘2sen(x)cos(x) = sen(2x)’)

coloca um título no gráfico atual, mostrado abaixo:



>>figure (6);

>>plot3(y,z,x), grid

cria um gráfico tridimensional com grade.

2.10.1 Estilos de Linha, Marcadores e Cores

Você pode especificar as cores e os tipos de linha que quiser dando ao comando plot um terceiro argumento depois de cada par de vetores de dados. O argumento

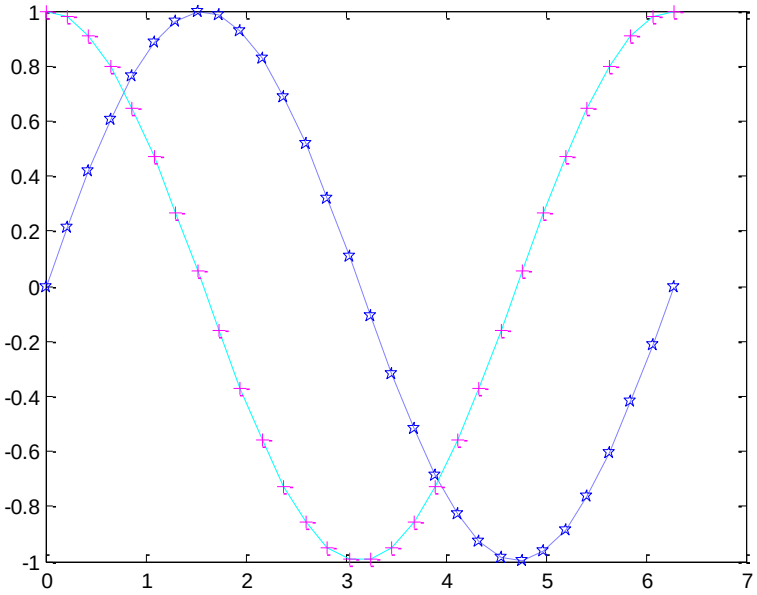
opcional adicional é uma string contendo um ou mais caracteres, de acordo com a tabela abaixo:

Símbolo	Cor	Símbolo	Marcador	Símbolo	Tipo de linha
b	azul	.	ponto	-	linha contínua
g	verde	o	círculo	:	linha pontilhada
r	vermelho	x	X	-.	traços e pontos
c	ciano	+	+	--	linha tracejada
m	magenta	*	estrela		
y	amarelo	s	quadrado		
k	preto	d	losango		
w	branco	∨	triângulo para baixo		
		∧	triângulo para cima		
		<	triângulo p/ a esquerda		
		>	triângulo p/ a direita		
		p	pentagrama		
		h	hexagrama		

Se você não especificar uma cor, o MATLAB começa com o azul e segue ciclicamente pelas sete primeiras cores da tabela para cada nova curva. O tipo de linha habitualmente usado é o contínuo, a menos que você especifique um tipo de linha diferente. Não há marcador padrão; se nenhum marcador é especificado, eles não são desenhados. Se algum marcador é escolhido, o símbolo correspondente é colocado em cada ponto dos dados, mas estes não são conectados, a não ser que um tipo de linha também seja selecionado.

Se uma cor, um marcador e um estilo de linha são especificados, a cor se aplicará tanto aos marcadores como à linha. Para conseguir um gráfico com uma cor diferente para os marcadores, faça dois gráficos para os mesmos dados alterando a especificação das curvas. Eis um exemplo em que se usa diferentes estilos de linha, cores e marcadores de pontos:

```
>>figure (7);
>> plot(x,y,'b:p',x,z,'c-',x,z,'m+')
```



2.10.2 Estilos de Gráficos

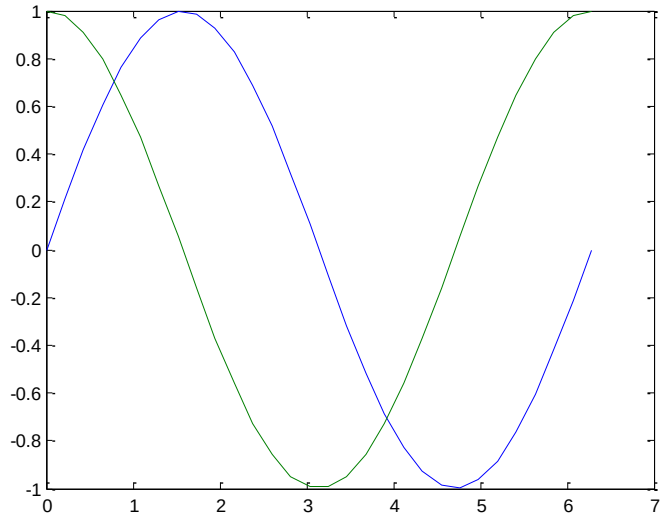
O comando colordef permite que você selecione um estilo genérico para seus gráficos. O estilo-padrão é colordef white. Esse estilo usa um fundo branco para os eixos, um fundo cinza-claro para figuras, preto para legendas e azul, verde-escuro e vermelho para as três primeiras curvas traçadas. Se você prefere um fundo preto, use colordef black. Esse estilo usa um fundo preto para os eixos, um fundo cinza-escuro para figuras, branco para legendas e amarelo, magenta e ciano para os três primeiros gráficos traçados. Se você optar por colordef none, o MATLAB usará o padrão da versão 4. Esse estilo usa o preto como fundo para os eixos e figuras, o branco para legendas e amarelo, magenta e ciano para os três primeiros gráficos traçados.

2.10.3 Grades, Eixos, Legendas e Títulos

O comando grid on adiciona linhas de grade ao gráfico atual nas posições dos eixos em que há marcadores. O comando grid off remove a grade. O comando grid sem argumentos adiciona e retira alternadamente a grade. O MATLAB inicia sempre com grid off para gráficos bidimensionais. Normalmente, os eixos bidimensionais estão inscritos em um retângulo contínuo, chamado caixa dos eixos. Essa caixa pode ser removida com o comando box off; o comando box on, por sua vez, restaura a caixa. O comando box alterna o estado da caixa dos eixos

entre on e off. Podemos atribuir nomes aos eixos horizontais e verticais com os comandos xlabel e ylabel, respectivamente. O comando title adiciona uma linha de texto ao topo do gráfico. Vamos usar o gráfico do seno e do cosseno novamente, como exemplo:

```
>>x=linspace(0, 2*pi, 30);
>>y=sin(x);
>>z=cos(x);
>>figure (8);
>>plot(x,y,x,z)
```



Agora, retiremos a caixa dos eixos e adicionemos um título e nome aos eixos:

```
>>box off        %remove a caixa dos eixos quando colocado após o comando plot
>>xlabel (‘Variável independente X’)    % legenda do eixo horizontal
>>ylabel (‘Variáveis dependentes Y e Z’)   % legenda do eixo vertical
>>title (‘Curvas do seno e cosseno’)   % título do gráfico
```

É possível adicionar ao gráfico qualquer texto em qualquer posição usando-se o comando text. O formato é text(x,y, ‘texto desejado’), com (x, y) representando as coordenadas, nas unidades próprias do gráfico em questão, do ponto central do lado esquerdo da string de texto. Para adicionar um texto identificando a curva do seno na posição (2.5, 0.7) usa-se:

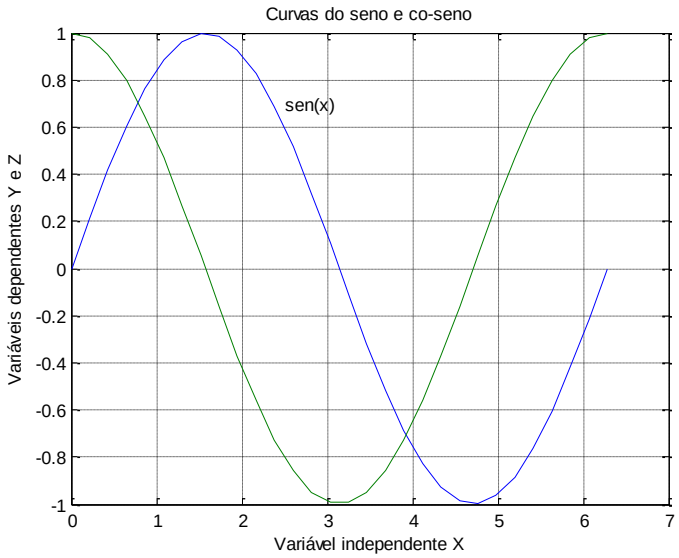
```
>> grid on, box on % restaura a caixa dos eixos e a grade
>> text (2.5, 0.7, 'sen(x)')
```

Se você quiser adicionar um texto, mas não quiser perder tempo imaginando as coordenadas a serem usadas, pode colocar um texto com o mouse. O comando

gtext muda para a janela de figuras em uso e desenha nela uma cruz que segue o mouse até que você dê um clique com o mouse ou pressione uma tecla.

Quando um desses dois fatos ocorrer, o texto será colocado com o canto inferior esquerdo do primeiro caractere naquela localização. Tente colocar um nome na segunda curva no gráfico:

```
>>gtext('cos(x)')
```



Em vez de usar textos individuais para identificar os conjuntos de dados em seu gráfico, você pode usar uma legenda. O comando legend cria uma caixa de legenda no canto superior direito do gráfico, exibindo o texto que você especificar para cada linha de seu gráfico. Se você deseja mover a legenda, basta dar um clique e manter o botão esquerdo do mouse pressionado próximo aos lados da legenda e arrastá-la para a posição desejada. O comando legend off apaga a legenda. Tente esse exemplo:

```
>>legend('sen(x)',  'cos(x)')
```

Tente mover a legenda por seu gráfico com o mouse. Então, remova a legenda com:

```
>>legend off
```

O comando legend funciona também com outros tipos de gráficos e aceita um argumento opcional especificando a localização inicial da legenda.

2.10.4 Impressão de Figuras

O gráfico poderá ser impresso usando o comando Print do menu File (na barra de menu da janela de figuras) ou usando os próprios comandos do MATLAB. Para imprimir uma janela de figuras, dê um clique com o mouse ou use o comando figure(n) para ativa-la e, então, use o comando print.

```
>>print    % imprime o gráfico atual em sua impressora.
```

O comando orient muda a orientação da página na impressão. O modo portrait, que é o padrão, imprime verticalmente no meio da página. O modo landscape imprime horizontalmente e preenche a página. O modo tall imprime verticalmente, mas preenchendo a página. O modo de impressão escolhido continua sendo usado até que você o mude ou até que termine sua sessão de trabalho no MATLAB.

```
>>orient                % qual é a orientação atual?
ans =
        portrait
>>orient landspace    % imprimindo de lado no papel
>>orient tall              % espichando a figura para preencher a página na vertical
```

2.10.5 Manipulação de Gráficos

Você pode adicionar curvas a um gráfico que já existe usando o comando hold. Depois de executado o comando hold on, o MATLAB não remove as curvas já existentes a cada novo comando plot. Em lugar disso, ele acrescenta as novas curvas no gráfico já existente. Entretanto, se os novos dados não se ajustam aos limites atuais dos eixos, estes são reescalados. O comando hold off libera a janela de figuras atual para novos gráficos. O comando hold se argumentos alterna o valor atribuído a hold. Se quisermos saber o estado da função hold é só digitar o comando ishold. O comando retorna 1 se hold on está em uso e 0 se hold on não estiver em uso.

Se você quiser desenhar dois ou mais gráficos em diferentes janelas, use o comando figure na janela de comandos, ou o item New Figure do menu File da janela de comandos ou da janela de figuras. O comando figure sem parâmetros cria uma nova janela.

Você pode escolher uma janela específica para ser usada como padrão selecionando-a com o mouse ou utilizando o comando figure(n), em que n é o número da janela que será usada pelos comandos plot subsequentes.

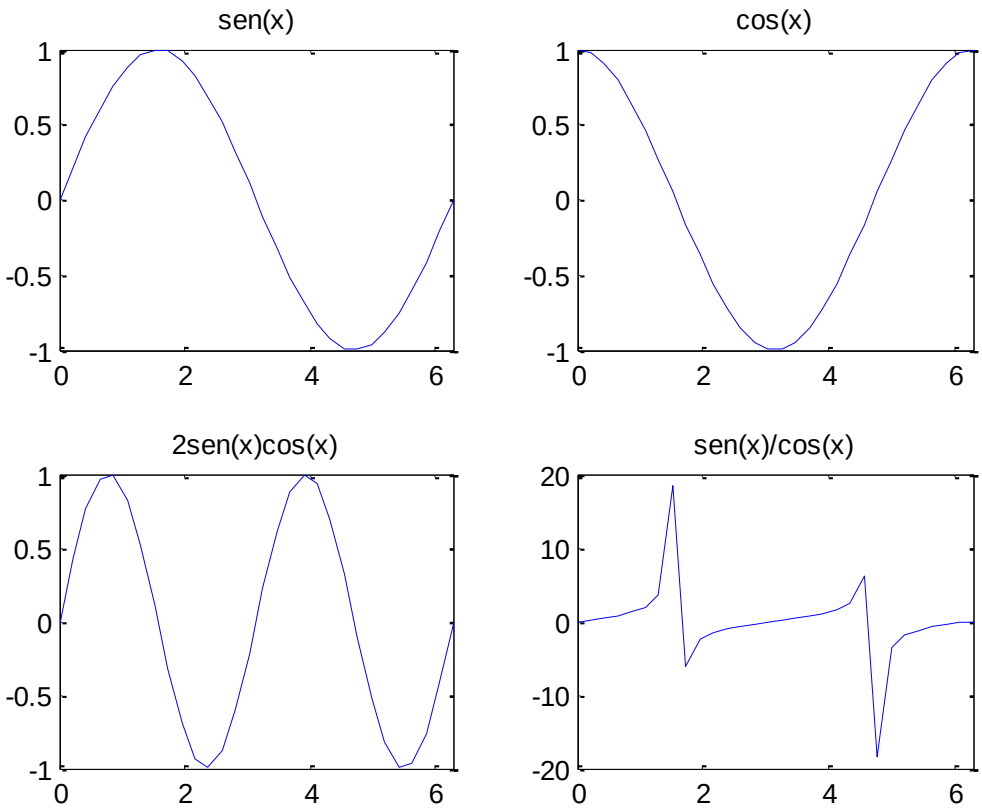
Por outro lado, uma janela pode conter mais de um conjunto de eixos. O comando subplot(m,n,p) subdivide a janela de figuras atual em uma triz com m por n regiões nas quais se pode traçar gráficos e ativa a região de ordem p. Os subgráficos são numerados da esquerda para a direita, primeiro na linha superior, depois na segunda linha e daí por diante. Por exemplo:

```
>> x=linspace(0,2*pi,30);
>> y=sin(x);
>> z=cos(x);
>> a=2*sin(x).*cos(x);
>> b=sin(x)./(cos(x)+eps);
>> figure (9);
>> subplot(2,2,1)   %selecionando o gráfico no alto, à esquerda
>> plot(x,y), axis([0 2*pi -1 1]), title('sen(x)')
>> subplot(2,2,2)   %selecionando o gráfico no alto, à direita
>> plot(x,z), axis([0 2*pi -1 1]), title('cos(x)')
>> subplot(2,2,3)   %selecionando o gráfico em baixo, à esquerda
>> plot(x,a), axis([0 2*pi -1 1]), title('2sen(x)cos(x)')
>> subplot(2,2,4)   %selecionando o gráfico em baixo, à direita
>> plot(x,b), axis([0 2*pi -20 20]), title('sen(x)/cos(x)')
```

Note que, quando um subgráfico está ativo, somente ele e seus eixos são modificados pelos comandos axis, hold, xlabel, ylabel, title, grid e box. Os demais subgráficos não são alterados. Além disso, o subgráfico ativo permanece em uso até que outro comando subplot ou figure seja executado. Se o novo comando subplot alterar o número de subgráficos na janela de figuras, alguns subgráficos traçados anteriormente serão apagados para dar espaço ao gráfico novo. Para voltar ao modo-padrão e usar toda a janela de figuras para um único par de eixos, use o comando subplot (1, 1, 1). Se você imprimir uma janela contendo vários gráficos, todos eles serão impressos na mesma página.

O comando zoom on ativa o zoom; o comando zoom off desliga o modo de ampliação.

Uma vez que tanto o comando legend quanto o comando zoom envolvem o uso do mouse na janela de figuras, eles interferem um com o outro. Assim, se desejarmos usar o zoom, é preciso assegurar que o comando legend off esteja ativo.



Podemos também plotar outros tipos de gráficos, exemplos:

A –

```
>> w = [0.51   0.35   0.10   0.02   0.02]     % valores percentuais
>> figure (10)
>> pie (w)
>> figure (11)
>> pareto (w)
```

B –

```
>> z = [8.3   7.2   5   32.8   18.7   10.2]    %relação candidato vaga por curso
>> figure (12)
>> bar (z)
```

C –

```
>> figure(13)
>> a = [0 0 1 1 1 0 0 0]
>> b = [0 0 1 1 0 0 0 0]
>> s = ~(a | b)
>> t = length (s)
>> stairs (s)    %plota o gráfico da operação NOR de A com B.
>> axis ([1 t -0.5 1.5])
```

Além de plotar gráficos, podemos ler e exibir imagem (**.jpg**) usando as funções imread () e imshow (). Exemplos desse tipo são mostrados nos Anexos de B até H.

2.11 Operações Relacionais e Lógicas

O MATLAB também inclui operações relacionais e lógicas. A finalidade desses operadores e funções é fornecer respostas do tipo Falso/Verdadeiro. Como entradas de todas as expressões relacionais e lógicas, o MATLAB considera qualquer número diferente de zero como Verdadeiro e o zero como Falso. A saída de todas as expressões lógicas e relacionais produzem 1 para Verdadeiro e 0 para Falso.

2.11.1 Operadores Relacionais

Os operadores relacionais do MATLAB incluem todas as comparações habituais:

Operador	Descrição
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a
==	Igual a
~=	Não igual a

Os operadores relacionais do MATLAB podem ser usados para comparar dois vetores de mesmo tamanho ou para comparar um vetor a um escalar. No segundo caso, o escalar é comparado a todos os elementos do vetor e o resultado tem o mesmo tamanho que o vetor. Por exemplo:

```
>>A=1:9, B=9-A
A =
        1   2   3   4   5   6   7   8   9
B =
        8   7   6   5   4   3   2   1   0
>>vf=A>4
vf =
        0   0   0   0   1   1   1   1   1
```

determina os elementos de A que são maiores que 4. Temos zero no resultado em que A não é maior do que 4 e um em que A > 4.

```
>>vf=A==B
vf =
        0   0   0   0   0   0   0   0   0
```

determina os elementos de A que são iguais àqueles de B. Observe que = e == têm significados diferentes: == compara duas variáveis e retorna um onde elas forem iguais e zero onde não forem; =, por outro lado, é usado para designar a saída de uma operação a uma variável.

```
>>vf=B-(A>2)
vf =
        8   7   5   4   3   2   1   0   -1
```

determina onde se tem A > 2 e subtrai o resultado de B. Esse exemplo nos mostra que, uma vez que as saídas de operações lógicas são arranjos numéricos de zero e um, elas também podem ser usadas em operações matemáticas.

```
>>B=B+(B==0)*eps
B =
         Columns 1 through 7
         8.0000    7.0000    6.0000    5.0000    4.0000    3.0000    2.0000
          Column 8 through 9
         1.0000   0.0000
```

é uma demonstração de como substituir os elementos iguais à zero em um vetor pelo número especial de MATLAB eps, cujo valor é de aproximadamente 2.2e-16 (ou 2.2×10^{-16}). Essa expressão em particular é algumas vezes útil para se evitar a divisão por zero como, por exemplo, em:

```
>>x=(-3:3)/3
x =
        -1.0000   -0.6667   -0.3333   0   0.3333   0.6667   1.0000
>>sin(x)./x
warning: Divide by zero
ans =
        0.8415   0.9276   0.9816   NAN   0.9816   0.9276   0.8415
```

o cálculo da função sin(x) / x nos dá uma mensagem de aviso, pois o quinto dado é igual à zero. Já que sin(0) / 0 é indefinido, o MATLAB retorna NAN (que significa não-número) à posição correspondente no resultado. Tente novamente, desta vez trocando o zero por eps:

```
>>x=x+(x==0)*eps;
>>sin(x)./x
ans =
```

0.8415 0.9276 0.9816 1.0000 0.9816 0.9276 0.8415

Agora temos a resposta correta, no limite, para sin(x)/x.

2.11.2 Operadores Lógicos

Os operadores lógicos permitem combinar ou negar expressões relacionais. Os operadores lógicos do MATLAB são:

Operador	Descrição
&	E
|	Ou
~	Não

Alguns exemplos do uso de operadores lógicos:

```
>>A=1:9
vf=A>4
vf =
     0   0   0   0   1   1   1   1   1
```

determina onde A é maior do que 4.

```
>>vf=~(A>4)
vf =
     1   1   1   1   0   0   0   0   0
```

nega o resultado acima, isto é, troca um por zero, e vice-versa.

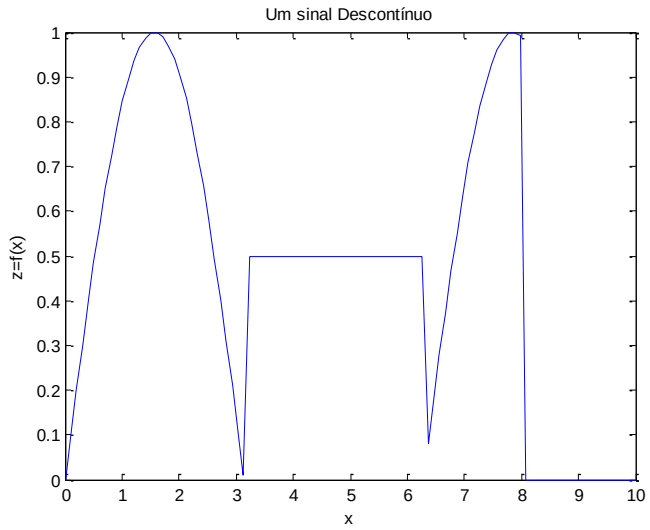
```
>>vf=(A>2)&(A<6)
vf =
     0   0   1   1   1   0   0   0   0
```

retorna um onde A for maior do que 2 E menor do que 6.

Finalmente, os recursos acima tornam fácil gerar vetores representando sinais com descontinuidades ou sinais que são compostos de segmentos de outros sinais. A ideia básica é multiplicar os valores de um vetor que você quiser manter por um e multiplicar os demais por zero. Por exemplo:

```
>>x=linspace(0,10,100);      %cria os dados.
>>y=sin(x);                  %calcula o seno.
>>z=(y>=0).*y;               %faz com que os valores negativos de sen(x) sejam
                                iguais a zero.
>>z=z+0.5*(y<0);             %soma 0.5 onde sin(x) é negativo.
>>z=(x<=8).*z;               %faz com que os valores acima de x=8 sejam
                                iguais  a zero.
```

```
>>plot(x, z)
>>xlabel('x'),ylabel('z=f(x)');
>>title('Um sinal Descontínuo')
```



Além dos operadores relacionais e lógicos básicos citados acima, o MATLAB possui ainda diversas outras funções lógicas e relacionais, como:

Função	Descrição
xor(x, y)	Operação ou exclusivo. Retorna um onde x ou y for diferente de zero (Verdadeiro). Retorna zero onde tanto x como y forem zero (Falso) ou ambos forem diferentes de zero (Verdadeiro).
any(x)	Retorna um se algum elemento do vetor x for diferente de zero. Retorna um para cada coluna de uma matriz x que tiver elementos diferentes de zero.
all(x)	Retorna um se todos os elementos de um vetor x forem diferentes de zero. Retorna um para cada coluna de uma matriz x que tiver todos os elementos diferentes de zero.
isnan(x)	Retorna um para os NaN de x.
isinf(x)	Retorna um para os Inf de x.
finite(x)	Retorna um para os valores finitos de x.

A tabela abaixo mostra a ordem de precedência para os operadores aritméticos, lógicos e relacionais, sendo que a linha superior tem a maior precedência.

Tabela de Precedência dos Operadores MATLAB
^ .^ ' .'

* / \ .* ./ .\
+ - ~ +(unário) -(unário)
: > < >= <= == ~=
| &

2.12 Texto

O poder principal do MATLAB consiste em sua capacidade de processar números. Entretanto, há situações em que é desejável manipular texto (exemplos nos Anexos de B até H), como ocorre quando damos nomes a eixos ou títulos a gráficos. No MATLAB, os textos são denominados cadeias de caracteres ou simplesmente strings. Os strings de caracteres são manipulados tal como vetores de linha:

```
>>t = 'Que tal esse string de caracteres?'
t =
        Que tal esse string de caracteres?
```

Um string de caracteres é simplesmente um texto cercado de aspas simples.

```
>>u=t(24:33)
u =
      caracteres
```

Os strings podem ser endereçados tal como os vetores. Aqui, os elementos de 24 a 33 constituem-se a palavra caracteres.

```
>>u=t(33:-1:24)
u =
     seretcarac
```

A manipulação de vetores também funciona. Essa é a palavra caracteres soletrada de trás para frente.

Assim como no caso das matrizes, os strings de caracteres podem ter múltiplas linhas, mas cada linha tem de ter o mesmo número de colunas. Consequentemente, torna-se necessário acrescentar espaços em branco nas linhas acima para tornar todas as linhas do mesmo comprimento. Operações matemáticas com strings também são possíveis! Entretanto, assim que você executa alguma operação matemática com um string, o string não é mais apresentado como um string. Em vez disso, é visualizado como um vetor de números no padrão ASCII. Esse padrão associa números inteiros de 0 a 255, com

256 caracteres diferentes, que incluem todas as teclas de seu teclado. Para ver a representação ASCII de um string, tome o valor absoluto ou some zero a ele.

```
>>s='ABCDEFG'
s =
     ABCDEFG
>>m=abs(s)
m =
        65  66  67  68  69  70  71
>>m=s+0
m =
        65  66  67  68  69  70  71
```

Como você pode ver, as letras do alfabeto encontram-se em ordem numérica. Agora ficou claro que strings e vetores são realmente a mesma coisa. Depois que um string foi convertido para sua representação ASCII, o string pode ser retransformado para que seja novamente representado como um string de caracteres usando-se a função MATLAB setstr:

```
>>setstr(m)
ans =
        ABCDEFG
```

E se agora somarmos 5 a s e se o convertermos novamente em string:

```
>>n=s+5
n =
        70  71  72  73  74  75  76
>>setstr(n)
ans =
        FGHIJKL
```

Finalmente, mudemos caracteres maiúsculos para minúsculos, somando-se a diferença entre a e A.

```
>>n=s+'a'-'A'
n =
        97  98  99  100  101  102  103
>>setstr(n)
ans =
        abcdefg
```

2.12.1 Conversão de Texto

O MATLAB possui várias funções de conversão, incluindo as seguintes:

Conversão de Texto	
base2dec	String de base x para inteiro decimal. Ex.: base2dec('10',16)=16 Converte um num. de q.q. base para base decimal.
bin2dec	String binária para inteiro decimal. Ex.: bin2dec ('111') = 7
char	String para ASCII. Ex.: char(65) = A
dec2base	Inteiro decimal para string de base x. Ex.: dec2base(5, 2) = 101
dec2bin	Inteiro decimal para string binária. Ex.: dec2bin (3) = 11
dec2hex	Inteiro decimal para string hexadecimal. Ex.: dec2hex (12) = c
double	ASCII para string. Ex.: double ('A') = 65
fprintf	Escreve texto formatado em um arquivo ou na tela.
hex2dec	String hexadecimal para inteiro decimal. Ex.: hex2dec ('c') = 12
int2str	Inteiro para string.
num2str	Número para string.
sprintf	Número para string, com controle de formato.
sscanf	String para número, com controle de formato.
str2num	String para número.

Em muitas situações, é desejável incorporar um resultado numérico dentro de uma string. Várias funções de conversão são úteis para esse propósito.

```
>>raio=2.5; area=pi*raio^2;
>>t = [' Um  círculo de raio ' num2str(raio) ...
        ' possui uma área de ' num2str(area) '.'];
>>disp(t)
Um círculo de raio 2.5 possui uma área de 19.635.
```

Aqui, a função num2str foi usada para converter números em strings e a concatenação de string foi usada para incorporar os números convertidos em uma frase. De maneira análoga, int2str converte inteiros em strings. As funções num2str e int2str recorrem a função sprintf, que possui uma sintaxe semelhante à da linguagem C para converter números em strings.

Abaixo, segue exemplo, do uso das funções disp e fprintf usadas com o mesmo objetivo:

```
>>a = 10;
>>b = 20;
```

```
>>c = 'oi';
>>fprintf ('O valor de a eh %d O valor de b eh %d O valor de c eh %s\n', a, b, c)
>>disp (['O valor de a eh ' num2str(a) ' O valor de b eh ' num2str(b) ' O valor de c eh ' c])
    O valor de a eh 10 O valor de b eh 20 O valor de c eh oi
    O valor de a eh 10 O valor de b eh 20 O valor de c eh oi
```

2.12.2 Funções de Texto

O MATLAB possui várias funções para manipulação de texto. Destacamos as seguintes:

Funções de Texto	
blanks(n)	Retorna uma string contendo n espaços em branco.
deblank(s)	Remove os espaços em branco no final de uma string.
eval(string)	Executa a string como um comando do MATLAB. Ex.: eval('sqrt(4)')
findstr(s1, s2)	Procura uma string dentro de outra.
ischar(s)	Verdadeira se o argumento for uma string.
isletter(s)	Verdadeira se houver alguma letra do alfabeto.
isspace(s)	Verdadeira se houver algum espaço em branco.
lasterr	Exibe a string do último erro ocorrido no MATLAB
lower(s)	Converte para minúsculas.
strcat(s1,s2,...)	Concatena horizontalmente strings.
strcmp(s1,s2)	Verdadeira se as strings forem idênticas.
strjust(s)	Ajusta uma matriz de strings em ambas as margens.
strmatch(s1,s2)	Procura possíveis ocorrências de uma string.
strncmp(s1,s2,n)	Verdadeira se os primeiros n caracteres forem idênticos.
strrep(s1,s2)	Substitui uma string por outra.
strtok(s)	Encontra o primeiro símbolo de uma string.
strvcat(s1,s2, ...)	Concatena strings verticalmente.
upper(s)	Converte para maiúsculas.

Algumas das funções listadas na tabela anterior podem ser usadas para verificações gramaticais elementares de strings. Por exemplo, a função findstr pode ser usada para encontrar a posição de uma string dentro de outra:

```
>>b = 'O rato roeu a roupa do rei de Roma';
>>findstr(b, ' ')          % procura os espaços em branco.
```

```
ans =
        2       7       12      14      20      23      27      30
>>findstr(b, 'r')        % procura a letra r minúscula
ans =
        3       8       15      24
>>find(b=='r')           % o comando find funciona para um único caracter.
ans =
        3       8       15      24
>>findstr(b, 'vaca')
ans =
        [ ]
>>findstr(b, 'roupa')    % procura a palavra roupa
ans =
        15
```

Note que essa função diferencia maiúsculas e minúsculas e retorna a string vazia quando nada é encontrado. A função findstr não funciona com matrizes de strings.

```
>>strrep(b, 'r','R')              % Converte para maiúsculas todas as letras r
ans =
        O Rato Roeu a Roupa do Rei de Roma
>>strrep(b, 'Roma', 'Riga')      % Troca Roma por Riga
ans =
        O rato roeu a roupa do rei de Riga
```

Conforme foi mostrado anteriormente, a função strrep realiza substituições simples de strings. Ela não funciona com matrizes de strings; é necessário reacomodar a matriz em um vetor antes de usá-la.

2.12.3 Vetores Celulares de Texto

Às vezes, o fato de todas as linhas em uma matriz de texto precisarem ter o mesmo número de colunas é problemático, principalmente quando há grande variação, de uma linha para outra, nas porções não ocupadas por espaços em branco. Esse problema pode ser eliminado usando-se os vetores celulares. Qualquer tipo de dado pode ser armazenado em um vetor celular, mas seu uso mais frequente é para armazenar strings de caracteres. Um vetor celular é um tipo de dado que lhe permite nomear e manipular com simplicidade um grupo de dados de vários tipos e tamanhos. Por exemplo:

```
>>C={' Que tal ' ; ' isto ' ; ' para um ' ; ' vetor celular de texto? '}
```

```
C =
        ‘ Que tal ‘
        ‘ isto ‘
        ‘ para um ‘
        ‘ vetor celular de texto? ‘
>>size(C)
ans =
        4       1
```

Observe que, para criar vetores celulares, são usadas as chaves e que as aspas que envolvem cada string são exibidas. Nesse exemplo, o vetor celular C possui 4 linhas e uma coluna.

Contudo, os elementos do vetor celular contêm strings de caracteres de comprimentos diferentes. Vetores celulares são recipientes que armazenam diferentes tipos de dados do MATLAB.

Vetores celulares são endereçados da mesma forma que os vetores usuais:

```
>>C(2 : 3)
ans =
        ‘ isto ‘
        ‘ para um ‘
>>C([4  3  2  1])
ans =
        ‘ vetor celular de texto? ‘
        ‘ para um ‘
        ‘ isto ‘
        ‘ Que tal ‘
>>C(1)
ans =
        ‘ Que tal ‘
```

Nesses exemplos, os resultados também são vetores celulares. Para recuperar o conteúdo de uma célula em particular, deve-se usar as chaves:

```
>>s=C{4}
s =
        vetor celular de texto?
>>size(s)
ans =
        1       23
```

Para extrair mais de uma célula, use a função deal:

```
>>[a, b, c, d] = deal(C{:})
a =
        Que tal
b =
        isto
c =
        para um
d =
        vetor celular de texto?
```

A forma C{:} denota todas as células como uma lista, ou seja, é equivalente a:

```
>>[a, b, c, d] = deal(C{1}, C{2}, C{3}, C{4})

a =
        Que tal
b =
        isto
c =
        para um
d =
        vetor celular de texto?
```

A função char converte o conteúdo de um vetor celular para um vetor de texto convencional:

```
>>s=char(C)
s =
        Que tal
        isto
        para um
        vetor celular de texto?
>>size(s)      % o resultado é um vetor de texto usual, preenchido com brancos
ans =
        4       23
>>ss=char(C(1:2))     % pode-se extrair partes
ss =
Que tal
Isto
```

```
>>size(ss)    % o resultado é um vetor de texto usual, preenchido com
brancos
ans =
        2       7
```

A conversão inversa é realizada pela função cellstr:

```
cellstr(s)
>>cellstr(s)
ans =
        ‘ Que tal ‘
        ‘ isto ‘
        ‘ para um ‘
        ‘ vetor celular de texto? ‘
```

A maioria das funções de texto do MATLAB trabalha tanto com os vetores usuais de texto como com os celulares.

2.13 Funções de Arquivos M

Quando você utiliza funções MATLAB tais como inv, abs, angle e sqrt, o MATLAB toma as variáveis que você definiu, calcula os resultados desejados usando sua entrada e depois lhe devolve os resultados. Os comandos executados pela função, assim como quaisquer variáveis intermediárias criadas por esses comandos, encontram-se ocultos. Você só visualiza o que entra e o que sai.

Essas propriedades fazem das funções ferramentas muito poderosas para executar comandos que incluem dentro deles funções matemáticas úteis ou seqüências de comandos que apareçam frequentemente quando você estiver resolvendo algum problema de maior porte. Por causa desse recurso, o MATLAB apresenta uma estrutura que lhe permite criar suas próprias funções na forma de arquivos M armazenados em seu computador. Por exemplo, veja a função potencia é um bom exemplo de uma função de arquivo M.

```
function x=potencia(y)
%calcula a potencia de y na base 2.
x = 2.^y;
```

Um arquivo M de função é semelhante a um arquivo de instrução, pois se trata também de um arquivo-texto com extensão .m. Assim como no caso dos arquivos

M de instrução, os arquivos M de função não são introduzidos a partir da janela de comandos, mas sim de arquivos-texto externos criados com um editor de texto. Um arquivo M de função difere de um arquivo de instrução pelo fato de a função comunicar-se com o espaço de trabalho do MATLAB somente por meio das variáveis passadas a ele e por meio das variáveis de saída que ele cria. As variáveis intermediárias dentro da função não aparecem nem interagem com o espaço de trabalho do MATLAB. Conforme pode ser visto no exemplo acima, a primeira linha de uma função M define o arquivo M como uma função, especifica seu nome e os nomes de suas variáveis de entrada e saída. A sequência contínua de linhas de comentário que se segue trata do texto que é apresentado em resposta ao comando help: help potencia. Finalmente, o restante do arquivo M contém os comandos MATLAB que criam as variáveis de saída.

2.14 Análise de Dados

Por causa de sua filosofia voltada para matrizes, o MATLAB executa prontamente análises estatísticas em vetores de dados. Por convenção, os vetores de dados são armazenados em matrizes orientadas por colunas. Isto é, cada coluna da matriz representa amostras individuais das variáveis. Por exemplo, consideremos a temperatura máxima (em graus Celsius) em três cidades durante o período de uma semana (sete dias), que se encontra armazenada na variável temps em um arquivo M de instrução.

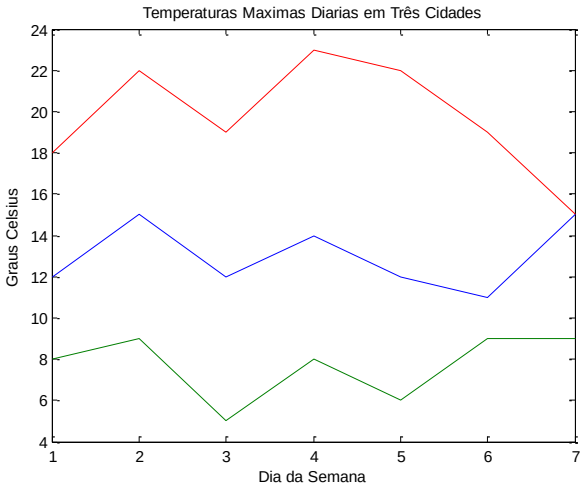
Quando executamos o arquivo M, colocamos a variável temps no espaço de trabalho do MATLAB. Se você quiser faze-lo por conta própria, a variável temps contém:

```
>>temps
temps =
     12     8     18
     15     9     22
     12     5     19
     14     8     23
     12     6     22
     11     9     19
     15     9     15
```

Cada linha contém as temperaturas máximas para um dado dia. Cada coluna contém as temperaturas máximas para uma cidade diferente. Para visualizar os dados, plote-os:

```
>> d=1:7;       %Número de dias da semana
```

```
>> plot(d,temps)
>> xlabel('Dia da Semana'),ylabel('Graus Celsius')
>> title('Temperaturas Máximas Diárias em Três Cidades')
```



O comando plot acima ilustra ainda uma outra forma de utilização do comando. A variável d é um vetor de comprimento 7; por outro lado, a variável temps é uma matriz 7 por 3. Com esses dados, o comando plot plota cada coluna de temps versus d.

Para ilustrar algumas das características de análises de dados de MATLAB, consideremos os seguintes comandos baseados nos dados de temperatura acima:

```
>>temp_med=mean(temps)
temp_med =
        13.0000    7.7143   19.7143
```

mostra que a terceira cidade teve a temperatura média mais alta. Aqui o MATLAB achou a média de cada coluna individualmente. Tomando-se a média uma vez mais, tem-se:

```
>>med_med=mean(temp_med)
med_med =
         13.4762
```

que acha a temperatura média total das três cidades. Quando a entrada de uma função de análise de dados é um vetor de linha ou de coluna, o MATLAB simplesmente faz a operação no vetor, retornando então um resultado escalar:

```
>>temp_max=max(temps)
temp_max =
        15     9    23
```

acha a temperatura mais alta máxima de cada cidade durante a semana.

```
>>[temp_max,x]=max(temps)
temp_max =
        15     9    23
x =
         2     2     4
```

acha a temperatura mais alta máxima de cada cidade e o índice de linha x, que indica onde o máximo apareceu. Por exemplo, x identifica o dia da semana no qual ocorreu a temperatura mais alta.

Para encontrar a temperatura mais alta mínima de cada cidade basta usar a função min.

```
>>desv=std(temps)
desv =
        1.6330    1.6036    2.8115
```

acha o desvio padrão de temps.

```
>>var_dia=diff(temps)
var_dia =
         3     1     4
        -3    -4    -3
         2     3     4
        -2    -2    -1
        -1     3    -3
         4     0    -4
```

calcula a diferença entre as temperaturas diárias mais altas, que descreve quanto à temperatura diária mais alta variou de dia para dia.

A análise de dados no MATLAB é feita em matrizes orientadas por coluna. As diversas variáveis são armazenadas em colunas individuais e cada linha representa uma observação diferente de cada variável. As funções estatísticas do MATLAB incluem:

Função Estatística	**Descrição**
corrcoef(x)	Coeficientes de correlação.
cov(x)	Matriz de covariância.
cumprod(x)	Produto cumulativo das colunas.
cumsum(x)	Soma cumulativa das colunas.
diff(x)	Calcula as diferenças entre os elementos.
hist(x)	Histograma ou gráfico de barras.

mean(x)	Média ou valor médio das colunas.
median(x)	Valor mediano das colunas.
prod(x)	Produto dos elementos nas colunas.
rand(x)	Números aleatórios uniformemente distribuídos.
randn(x)	Números aleatórios com distribuição normal.
sort(x)	Coloca as colunas em ordem crescente.
std(x)	Desvio padrão das colunas.
sum(x)	Soma dos elementos de cada coluna.

REFERÊNCIAS BIBLIOGRÁFICAS

HANSELMAN, Duane e Bruce Littlefield. **MATLAB-Versão do Estudante – Guia do Usuário**. São Paulo: Makron Book do Brasil, 1999, 303p. v.5.

LEÃO, M. **Introdução ao C++ Builder**. Rio de Janeiro: Axcel Books do Brasil, 1998. 165p.

MIZRAHI, Viviane V. **Treinamento em linguagem C++:** módulo 2. 2.ed. São Paulo: Makron Books, 1994. 318p.

Páginas da Internet:
www.md.cefetpr.br
www.scribd.com/doc/2298043/Apostila-Word-2007
www.unicamp.br

ANEXOS

ANEXO A

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

TÍTULO DO TRABALHO

NOME DOS ALUNOS

CURSO TÉCNICO EM ELETROTÉCNICA

PROFESSOR(A): NOME DO(A) PROFESSOR(A)

BELO HORIZONTE
ANO

Aluno: (NOME DO ALUNO 1)

Rua: (ENDEREÇO DO ALUNO)

Bairro:

Cidade:

Cep:

Tel: (TELEFONE DO ALUNO)

Aluno: (NOME DO ALUNO 2)

Rua: (ENDEREÇO DO ALUNO)

Bairro:

Cidade:

Cep:

Tel: (TELEFONE DO ALUNO)

LISTA DE FIGURAS

SUMÁRIO

1 – INTRODUÇÃO

O texto deve ser digitado em arial 12, espaçamento de 1,5 entre linhas, justificado e com recuo na primeira linha de 1,25, com espaçamento antes de 6 e depois de 12.

O título 1 deverá ser formatado da seguinte maneira: arial 14, negrito, letra maiúscula, alinhado à esquerda, recuo a esquerda e a direita em 0, especial nenhum, com espaçamento antes de 0 e depois de 42, espaçamento entre linhas simples. Todo Título 1 começa em uma nova página.

1.1 Título 2

O título 2 deverá ser formatado da seguinte maneira: arial 12, alinhado à esquerda, recuo a esquerda e a direita em 0, especial nenhum, com espaçamento antes de 24 e depois de 12, espaçamento entre linhas simples.

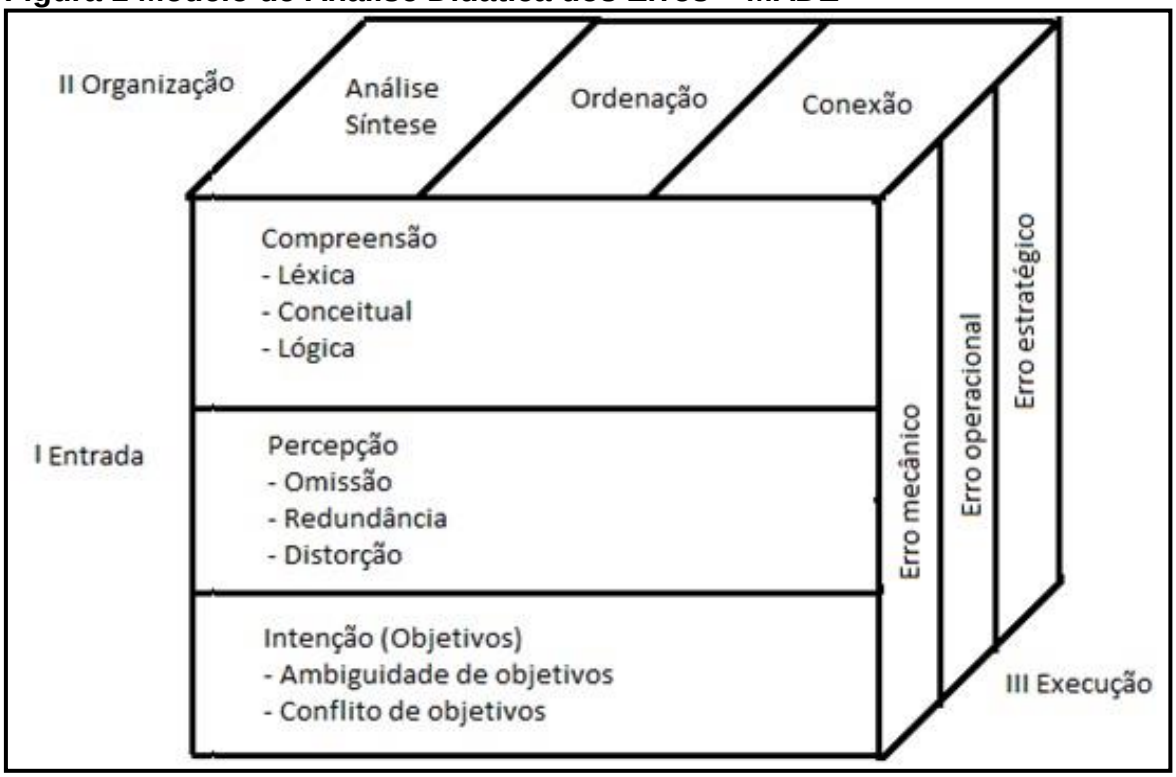
1.2 Numeração das páginas

Usa quebra de seção para que a numeração do trabalho ocorra somente a partir da página de Introdução e inicie com o número 5.

2 – DESENVOLVIMENTO

As Figuras, tabelas e quadros inseridos no trabalho tem que ser referenciado no corpo do texto, conforme descrito no exemplo a seguir: A Figura 1 mostra o MADE descrito por De La Torre (2007).

Figura 1 Modelo de Análise Didática dos Erros – MADE



Fonte: DE LA TORRE, 2007, p. 108.

Para a formatação das figuras, tabelas ou gráficos deve selecionar ao mesmo tempo, o título da figura (tabela ou gráfico), a figura (tabela ou gráfico) e a fonte. Em parágrafo escolha as seguintes opções: alinhado à esquerda, recuo a esquerda e a direita em 0, especial nenhum, com espaçamento antes e depois de 0, espaçamento entre linhas simples. Deixar uma linha em branco logo após a fonte da figura.

O título da figura deverá ser digitado conforme exemplo mostrado na Figura 1, escrito em arial 12 e negrito. A fonte deve ser digitada em arial 10.

2.1 Subtítulo

Se for quadro ou tabela, no lugar da palavra figura escreve quadro ou tabela. Sempre usar a formatação citada na Introdução.

Caso a tabela, quadro ou figura tenha sido elaborado pelo autor, na fonte se escreve da seguinte maneira – **Fonte: Elaborada pelo autor**.

Caso tenha sido retirada de um manual, na fonte tem que constar o nome do manual, ano e página.

3 – CONSIDERAÇÕES FINAIS

Exponha aqui suas considerações finais sobre a importância do desenvolvimento do trabalho.

4 – REFERÊNCIAS

Registre toda a bibliografia consultada seguindo as normas da ABNT. As referências devem ser digitadas em ordem alfabética.

Exemplos:

Referência de Livro:

BARROSO, J. M. **Conexões com a matemática**. 1. ed. v. 1. São Paulo: Moderna, 2010, 408 p.

Referência de Periódico:

RAMOS, M. L. P. D.; CURI, E. Análise de erro em avaliação de sistemas digitais: uma questão com lógica AND e flip-flop. **Revista Eletrônica em Educação Matemática**, Florianópolis, v.8, n.1, p. 232-247, 2013a.

Referência de Anais:

RAMOS, M. L. P. D. Detecção, identificação e retificação: as três fases no tratamento e na correção dos erros. In: ENCONTRO NACIONAL DE EDUCAÇÃO MATEMÁTICA, 11., 2013, Curitiba. **Anais...** Curitiba: ENEM-PR, p. 1-14, 2013. 1 CD-ROM.

Referência a Sites:

CEFET-MG. CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS. **Normas Acadêmicas dos Cursos da Educação Profissional Técnica de Nível Médio.** Belo Horizonte, 2014. Disponível em: <http://www.cepe.cefetmg.br/galerias/Arquivos_CEPE/Resolucoes_CEPE/Resolucoes_CEPE_2014/RES_CEPE_01_14.htm>. Acesso em: 04 fev. 2014.

Referência a Monografias, Dissertações ou teses:

PELHO, E. B. B. **Introdução ao conceito de função:** a importância da compreensão das variáveis. 2003. 146 f. Dissertação (Mestrado em Educação Matemática)- Pontifícia Universidade Católica de São Paulo, São Paulo, 2003.

5 – ASSINATURAS

Aluno 1: Nome completo do aluno

Aluno 2: Nome completo do aluno

ANEXO B

```
clc, clear all
b=input ('Digite um número decimal para ser usado na calculadora')
c=input ('Digite outro numero decimal para ser utilizado na calculadora')
p=length (b);
i=length (c);
if (p==1) & (i==1)
  disp ('Escolha uma operação para realizar')
  u=input ('somar, subtrair, multiplicar, dividir', 's');
  switch u
     case 'somar'
      t=b+c;
      sv=input ('Digite o resultado que você acha ser o correspondente')
      if sv==t
         disp ([ 'Você acertou, o resultado é:' num2str(t)])
         else disp (['Infelizmente você errou, o resultado era:' num2str(t)])
      end
     case 'subtrair'
      m=b-c;
      svl=input ('Digite o resultado que você acha ser o correspondente')
      if svl==m
         disp ([ 'Você acertou, o resultado é:' num2str(m)])
         else disp (['Infelizmente você errou, o resultado era:' num2str(m)])
      end
     case 'multiplicar'
      h=b*c;
      svll=input ('Digite o resultado que você acha ser o correspondente')
      if svll==h
         disp ([ 'Você acertou, o resultado é:' num2str(h)])
         else disp (['Infelizmente você errou, o resultado era:' num2str(h)])
      end
     case 'dividir'
      y=b/c;
      svlll=input ('Digite o resultado que você acha ser o correspondente')
      if svlll==y
```

```
        disp ([ 'Você acertou, o resultado é:' num2str(y)])
        else disp (['Infelizmente você errou, o resultado era:' num2str(y)])
      end
    otherwise
      disp ('Digite uma operação válida')
  end %switch
  else disp ('Digite somente um número')
end  %if
```

Trecho de código retirado e adaptado do trabalho das alunas Anna Gabrielle de Castro Santos e Sara Isabele dos Santos Bento Silva – 2018.

ANEXO C

```
clc
clear all
disp ('Bem-vindo, aqui será mostrado o funcionamento das portas lógicas principais.')
pause (1)
disp ('Escolha uma para começar.')
y={'1-and' '2-or' '3-nand' '4-nor' '5-xor' '6-xnor'};
for n=1:6
  fprintf ('Porta: %s\n',y{n})
end
a=input('Digite: ');
switch a
 case 1
   disp ('Essa é uma porta And, a saída será 1 quando todas as entradas forem 1')
   disp ('Digite os valores da primeira entrada:')
   for n=1:5
     x1(n)=input ('Digite: ');
   end
   disp ('Agora os valores da segunda entrada')
   for n=1:5
     x2(n)=input ('Digite: ');
   end
   saida=and (x1,x2)
   figure (1)
   stairs (saida)
   axis ([ 1 5 -0.5 1.5])
   figure (2)
   imshow (imread('and.JPG'))
case 2
   disp ('Essa é uma porta Or, quando qualquer entrada for 1, a saída será 1')
   disp ('Digite os valores da primeira entrada:')
   for n=1:5
    x1(n)=input ('Digite: ');
   end
   disp ('Agora os valores da segunda entrada')
```

```
   for n=1:5
     x2(n)=input ('Digite: ');
   end
   saida=or (x1,x2)
   figure (1)
   stairs (saida)
   axis ([1 5 -0.5 1.5])
   figure (2)
   imshow (imread('or.JPG'))
 case 3
   disp ('Essa é uma porta nand, quando todas as entradas forem 1 a saída sera 0')
   disp ('Digite os valores da primeira entrada:')
   for n=1:5
     x1(n)=input ('Digite: ');
  end
  disp ('Agora os valores da segunda entrada')
  for n=1:5
    x2(n)=input ('Digite: ');
  end
  saida=~(and(x1,x2))
  figure (1)
  stairs (saida)
  axis ([1 5 -0.5 1.5])
  figure(2)
  imshow(imread('nand.JPG'))
case 4
   disp ('Essa é uma porta nor,quando qualquer entrada for 1 a saída será 0')
   disp ('Digite os valores da primeira entrada:')
   for n=1:5
     x1(n)=input ('Digite: ');
   end
   disp ('Agora os valores da segunda entrada')
   for n=1:5
     x2(n)=input ('Digite: ');
   end
```

```
   saida=~(or(x1,x2))
   figure (1)
   stairs (saida)
   axis ([ 1 5 -0.5 1.5])
   figure (2)
   imshow (imread('nor.JPG'))
case 5
    disp ('Essa é uma porta xor, a saída será 1 quando as entradas estiverem em niveis
          lógicos diferentes')
    disp ('Digite os valores da primeira entrada:')
    for n=1:5
      x1(n)=input ('Digite: ');
    end
    disp ('Agora os valores da segunda entrada')
    for n=1:5
      x2(n)=input ('Digite: ');
    end
    saida=xor (x1,x2)
    figure (1)
    stairs (saida)
    axis ([1 5 -0.5 1.5])
    figure(2)
    imshow (imread('xor.JPG'))
 case 6
     disp ('Essa é uma porta xnor, a saída será 1 quando as entradas estiverem em níveis
           lógicos iguais')
     disp ('Digite os valores da primeira entrada:')
     for n=1:5
        x1(n)=input ('Digite: ');
    end
   disp ('Agora os valores da segunda entrada')
   for n=1:5
     x2(n)=input ('Digite: ');
   end
   saida=~(xor(x1,x2))
```

```
    figure (1)
    stairs (saida)
    axis ([1 5 -0.5 1.5])
    figure (2)
    imshow (imread('xnor.JPG'))
 otherwise
    disp ('você não digitou um item correspondente')
end %switch a
```

Trecho de código retirado do trabalho dos alunos Marcello Henrique Moura Pinto e Ricardo Rocha Dias Neves – 2018.

ANEXO D

```
clear all
clc
disp('Gostaria de executar esta parte de identificação de portas lógicas ou gostaria de pular para a parte dos gráficos ?')
Axx=input('(s-sim,n-não) : ','s');
while Axx~='s' & Axx~='S' & Axx~='n' & Axx~='N'
  disp('_____')
  disp('Por gentileza, apenas letras entre as opções sugeridas !')
  disp('Gostaria de executar identificação de portas lógicas ou finalizar o programa ?')
  Axx=input('(s-sim,n-não) : ','s');
end %while
if Axx=='s' | Axx=='S'
 WSC='s';
 while WSC~='n' & WSC~='N'
  disp('Identificação de Portas Lógicas :')
  for x=1:4
    ATP(x)=input('Entre com o número desejado (0 ou 1) : ','s');
  end %for
  disp('_____')
  switch ATP
    case '0001'
      disp('Porta identificada : AND')
      figure(1)
      ATPp=imread('portas-logicas-and.jpg');
      imshow(ATPp)
    case '0111'
      disp('Porta identificada : OR')
      figure(2)
      ATPo=imread('portas-logicas-or.jpg');
      imshow(ATPo)
    case '1110'
      disp('Porta identificada : NAND')
      figure(3)
      ATPi=imread('portas-logicas-nand.jpg');
```

```
        imshow(ATPi)
    case '1000'
        disp('Porta identificada : NOR')
        figure(4)
        ATPu=imread('portas-logicas-nor.jpg');
        imshow(ATPu)
    case '0110'
        disp('Porta identificada : XOR')
        figure(5)
        ATPy=imread('portas-logicas-xor.jpg');
        imshow(ATPy)
    case '1001'
        disp('Porta identificada : XNOR')
        figure(6)
        ATPt=imread('portas-logicas-xnor.jpg');
        imshow(ATPt)
    otherwise
        disp('Porta lógica não identificada !')
  end %switch
  disp('_____')
  WSC=input('Gostaria de rodar novamente ? (s-sim,n-não) : ','s');
 end   %while
 elseif Axx=='n' | Axx=='N'
    disp('_____OK_____')
end  %if
```

Trecho de código retirado do trabalho dos alunos Jhonata Rodrigo Teixeira Cunha e João Marcos Campos Caetano – 2018.

ANEXO E

```
clc, clear all
disp('Verificar TV de portas lógicas básicas e identificar a porta lógica referente a TV.')
R=[];
t=1;
c=1
while c==1
    R=[];
    t=1;
    for w=1:4
       R(t)=input('entre com as posições da tabela verdade desejada: (esta pergunta sera feita 4
                    vezes):')
       t=t+1;
   end %for
   if R==[0 0 0 1]
        J='AND';
        disp(J)
        c=0;
   elseif R==[0 1 1 1]
        J='OR';
        disp(J)
        c=0;
   elseif R==[1 1 1 0]
        J='NAND';
        disp(J)
        c=0;
   elseif R==[1 0 0 0]
        J='NOR';
        disp(J)
        c=0;
   elseif R==[0 1 1 0]
        J='XOR';
        disp(J)
        c=0;
   elseif R==[1 0 0 1]
```

```
        J='XNOR';
        disp(J)
        c=0;
    else
        disp('erro')
        c=input('voce deseja voltar ao programa?(se sim digite 1, se não digite 0):')
    end %if
end %while
```

Trecho de código retirado do trabalho dos alunos Felipe Moreira dos Reis e Gabriel Lucas Tinoco de Aguiar – 2018.

ANEXO F

```
clc
clear all
disp('Bem-vindo ao programa. Aqui você poderá realizar operações com ')
disp('módulos e plotar gráficos da função modular')
disp('Digite "Iniciar" para continuar e "Encerrar" para finalizar o programa')
a=input('Iniciar ou Encerrar programa? ','s')
while (strcmp(a,'Encerrar') ~= 1)
   disp('Digite 1, 2, 3, 4 ou 5')
   t=input('(1)Exercícios simples; (2)Gráfico simples; (3)Gráfico complexo; (4)Módulo de uma matriz; (5)Módulo de número binário (em decimal): ')
   j=input('Deseja realizar quantas operações? ')
   switch t
      case 1
         for v=1:j
           disp('Os exercícios simples são para visualizar o módulo de um ')
           disp('número ou de uma expressão')
           x=input('Entre com o número ou com a expressão: ')
           r=abs(x)
         end
      case 2
         for v=1:j
           disp('Para transladar o gráfico para a esquerda digite "Sim, esquerda"')
           disp('Para transladar o gráfico para a direita digite "Sim, direita"')
           disp('Para transladar o gráfico para cima digite "Sim, cima"')
           disp('Para transladar o gráfico para baixo digite "Sim, baixo"')
           disp('Para deixar na origem digite "Não" e deixe a constante em 0')
           m=input('Transladar gráfico? Se sim, para onde? ', 's')
           x=input('Entre com os valores de x (utilize um vetor): ')
           w=input('Constante de transladação: ')
           if (strcmp(m, 'Sim, esquerda') == 1)
             y=abs(x-w)
           elseif (strcmp(m, 'Sim, direita') == 1)
             y=abs(x+w)
```

```
      elseif (strcmp(m, 'Sim, cima') == 1)
        y=abs(x)+w
      elseif (strcmp(m, 'Sim, baixo') == 1)
        y=abs(x)-w
      elseif (strcmp(m, 'Não') == 1)
        y=abs(x+w)+w
      end
    plot(x,y)
    c=size(y, 2)
    end
case 3
    for v=1:j
      disp('Os gráficos complexos são aqueles transladados, simultaneamente, ')
      disp('para a horizontal e para a vertical. Digite "Esquerda e cima", "Esquerda ')
      disp('e baixo", "Direita e cima", "Direita e baixo"')
      m=input('Transladar gráfico para onde? ', 's')
      x=input('Entre com os valores de x (utilize um vetor): ')
      h=input('Constante de transladação horizontal: ')
      v=input('Constante de transladação vertical: ')
      if (strcmp(m, 'Esquerda e cima') == 1)
        y=abs(x-h)+v
      elseif (strcmp(m, 'Esquerda e baixo') == 1)
        y=abs(x-h)-v
      elseif (strcmp(m, 'Direita e cima') == 1)
        y=abs(x+h)+v
      elseif (strcmp(m, 'Direita e baixo') == 1)
        y=abs(x+h)-v
      end
    plot(x,y)
    c=size(y, 2)
    end
case 4
    for v=1:j
       x=input('Entre com a matriz (de mesma dimensão): ')
       y=abs(x)
```

```
        end
    case 5
        for v=1:j
            x=input('Entre com o número binário: ')
            y=bin2dec(num2str(abs(x)))
        end
    otherwise
        disp(['^ Opção inválida ^' t])
        y=NaN
  end
  a=input('Continuar ou Encerrar programa? ','s')
end
```

Trabalho dos alunos Jonas Amaral Carneiro Teixeira e Maisa Soares Barbosa – 2019.

ANEXO G

```
clc
clear all
disp('Este programa lerá os valores dos coeficientes a,b e c de uma função quadrática e te ajudará a compreendê-la melhor.')
pause(7)
k = input( 'Deseja iniciá-lo? Se sim, digite qualquer tecla. Se não, digite encerrar. ','s')
while (strcmp(k,'encerrar') ~= 1)
    a = input('Qual é o valor de A ')
    b = input('Qual é o valor de B ')
    c = input('Qual é o valor de C ')
    operacao = input('Voce desejsa encontrar: O "Delta", as "Raízes", os "Vertíces Max ou Min", o "Gráfico" ou "Resumo" ','s')
    switch  operacao
        case 'Delta'
         QNTSD= input('Você deseja calcular 1 ou 3 deltas?')
         if QNTSD==1
             D= b^2-4*a*c
          if D>0
             disp('esta função contém 2 raízes reais e diferentes')
          elseif D==0
             disp('esta função contém 2 raízes iguais')
          else
                disp('esta função não contém raízes reais')
          end
         elseif QNTSD==3
            for zz=1:3
               D = b^2-4*a*c
               if D>0
             disp('esta função contém 2 raízes ')
          elseif D==0
             disp('esta função contém 1 raíz')
          else
                disp('esta função não contém raízes reais')
```

```
            end
        if zz<3
        a = input('Qual é o valor de A ')
        b = input('Qual é o valor de B ')
        c = input('Qual é o valor de C ')
        else
           break
        end
          end
       else
          disp('Este programa esta apto para calcular apenas 1 ou 3 valores de delta')
       end
      case 'Raízes'
        D=b^2-4*a*c
        if D>=0
        X1 = (-b+sqrt(D))/a*2
        X2 = (-b-sqrt(D))/a*2
        else
          disp('Esta função não contém raízes reais')
        end
      case 'Vertíces Max ou Min'
        if a>0
           disp('Sua função possui concavidade voltada para cima, portanto possui vértice
mínimo.')
           pause(5)
           Xv=-b/2*a
           Yv=-(b^2-4*a*c)/4*a
           disp('Xv e Yv são os valores das coordenadas do vértice da função')
        else
           disp('Sua função possui concavidade voltada para baixo, portanto possui vértice
máximo.')
           pause(5)
           disp('Os valores das coordenadas do seu vértice são: ')
           pause(2)
           Xv=-b/2*a
```

```
            Yv=-(b^2-4*a*c)/4*a
        end
    case 'Gráfico'
        X = input('informe o intervalo do seu X em forma de vetor ')
        Y = a.*X.^2 + b.*X + c
        plot(X,Y)
        title ('Gráfico da sua função quadrática')
        xlabel ('x')
        ylabel ('y')
    case 'Resumo'
        imshow(imread('resumo.jpg'))
    otherwise
        disp('Este programa não realiza este comando')
end
k = input('Deseja continuar estudando funcao quadratica? Se sim, aperte em qualquer tecla. Se não,digite encerrar.','s')
end
```

Trabalho dos alunos Emanuelly Souza Coimbra Silva e Samuel Aguiar de Moraes – 2019.

ANEXO H

```
clc
clear all
disp('Olá parceirx, quer aprender mais sobre física?')
disp('Nesse jogo vamos viver grandes aventuras nos aprofundando ainda mais no mundo da dinâmica')
disp('Este jogo parte de conhecimentos prévios então esteja preparado!! ')
Jogo=input('Vamos começar?','s')
while(strcmp(Jogo,'não') ~= 1)
   x=input('Separamos alguns desafios especialmente para você, pois você é demais!! Escolha um número de 1 a 5 ou o 6 que é o desafio final  e divirta-se.')
   switch x
      case 1
        disp('Em uma competição de cabo de guerra, a equipe adversária está aplicando uma força de 5N pra esquerda e seu time está aplicando uma força de 3N pra direita, baseando-se na ideia de força  resultante, qual o mínimo de força a mais deverá ser aplicada para que sua equipe vire o jogo?')
         for n=1:3
            R=input('Resposta:maior que ')
            if R==2
               disp('Parabéns você acertou!!')
               break;
            else R~=2
               disp('Essa não é a resposta,tente novamente')
            end
      disp('Se esforce mais na próxima!')
        end
      case 2
         M=[0:1:15]
         a=[0:2:30]
         figure(1)
         plot(M,a)
         ylabel('A(m/s^2)')
         xlabel('M(kg)')
```

```
        disp('A partir do gráfico determine a força desde a aceleração 0 até a aceleração 30m/s^2')
        for n=1:3
            R=input('Resposta:')
            t=(15*30)/2;
            if R==t
                disp('Parabéns você acertou!!')
                break;
            else R~=t
                disp('Essa não é a resposta,tente novamente!')
            end
           disp('Se esforce mais na próxima!')
        end
      case 3
       y={'1' '4' 'k';'-7' '1' 'w';'9' '1' 'p'}
        disp('Complete k,w e p com os números necessários para que cada linha tenha a força resultante igual a 0')
        for n=1:3
            R=input('k='),RI=input('w='),Ry=input('p=')
            if (R==-5)&(RI==6)&(Ry==-10)
                k=-5;
                w=6;
                p=-10;
                fprintf('O valor de k eh %d O valor de w eh %d e O valor de p eh %d\n',k,w,p)
                disp(['O valor de k eh' num2str(k) 'O valor de w eh' num2str(w) 'O valor de p eh' num2str(p)])
                y{1,3}='-5';
                y{2,3}='6';
                y{3,3}='-10';
                y
                break;
            else (R~=-5)|(RI~=6)|(Ry~=-10)
                 disp('Essa não é a resposta,tente novamente!')
            end
           disp('Se esforce mais na próxima!')
```

```
        end
    case 4
        disp('Analise as afirmações a seguir :')
        primeira='É possível definir a segunda lei de Newton em função da quantidade de movimento.'
        segunda='Um objeto depositado sobre uma superfície qualquer sofrerá a ação da força Normal como reação à força Peso.'
        terceira='A massa é a grandeza que representa a dificuldade imposta por um corpo à mudança de seu estado inicial'
        for n=1:2
            R=input('Quantas das alternativas anteriores são verdadeiras?')
            if R==2
                disp('Parabéns você acertou!!')
                 break;
            else R~=2
                disp('Essa não é a resposta,tente novamente!')
            end
           disp('Se esforce mais na próxima!')
        end
    case 5
        a=[6 -8 3;4 5 -6;7 8 -9]
        disp('Dada a matriz diga as coordenadas do numero que somado a -8 da força resultante igual a zero.')
        disp('l=linha e c=coluna')
        l=input('Em que linha o número está?')
        c=input('Em que coluna o numero está?')
        for n=1;3
            if (l==3)&(c==2)
            [l,c]=find(a>=8)
            disp('Parabéns você acertou!')
            break;
            else (l~=3)|(c~=2)
                disp('Essa não é a resposta,tente novamente!')
            end
            disp('Se esforce mais na próxima!')
```

```
    end
case 6
    disp('Marcos e seus amigos brincaram de queda de braço.')
    disp('Quando dois amigos faziam a mesma força por certo tempo, empatavam.')
    M=6;,J=-6;,P=7;,C=-8;,D=5;,A=-5;
    fprintf('primeiro jogo;Marcos fez %d e João fez %d\n',M,J);
    disp(['primeiro jogo:Marcos fez:' num2str(M) 'Joao fez:' num2str(J)])
    fprintf('Segundo jogo:Pedro fez: %d e Caio fez: %d\n',P,C);
    disp(['Segundo jogo:Pedro fez:' num2str(P) 'e Caio fez: ' num2str(C)])
    fprintf('Terceiro jogo:Diego fez: %d e Arthur fez: %d\n',D,A);
    disp(['Terceiro jogo:Diego fez:' num2str(D) 'e Arthur fez:' num2str(A)])
    Lm=M+J;,Km=P+C;,Jm=D+A;
    R=input('Quais os dois jogos tiveram o mesmo resultado ?','s')
    if (strcmp(R,'primeiro e segundo')==1)
      s=isequal(Lm,Km)
      switch s
        case 1
          disp('Você acertou!')
        case 0
          disp('você errou!')
      end
      elseif (strcmp(R,'primeiro e terceiro')==1)
        s=isequal(Lm,Jm)
         switch s
        case 1
          disp('Você acertou!')
           otherwise
          disp('Você errou!')
         end
      else (strcmp(R,'segundo e terceiro')==1)
        s=isequal(Km,Jm)
        switch s
        case 1
          disp('Você acertou!')
          otherwise
```

```
                disp('você errou!')
            end
        end
    otherwise
        disp(['Não encontrado:' num2str(x)])
end
Jogo=input('Vamos começar?','s')
End
```

Trabalho das alunas Gabrielly Beatriz Felício de Souza e Isabella Carla Alves – 2019.