Introdução ao MatLab

- Apresentação do cronograma e da ementa do curso;
- Breve histórico do *software*;
- Interface do MatLab
- Como abrir, editar e chamar um código;
    - Digitar *edit* na interface do MatLab
    - Para chamar o código: *edit codename.m*
    - Quando tiver * ao lado do nome do código, significa que teve alterações que não foram salvas: *codename.m**
    - Para atribuir variável usamos =
    - Para usar o símbolo de igualdade usamos ==
        - Exemplo: verifique a validade da sentença: 5-6==3+-4 Se resultar em 1 a sentença é dita verdadeira (true) e se resultar em 0 é dita falsa (false)
    - Fazer comentário no código (%)
    - Para que na próxima linha tenha a continuidade do seu código se deve colocar ...
        - Exemplo: (5+6)-32*5 - ... 1/52+32
        - Tudo na linha após ... automaticamente vira comentário
    - Para criar blocos de comentário fazemos
        - %{

          Textotextotextotexto

          %}
    - Para dividir em seções: %%
    - Para transformar em comentário: ctrl+R
    - Para deixar de ser comentário: ctrl+T
- Operações básicas usadas no MatLab;
    - Declarar variável:
        - Exemplo: renda = 1500;
        - Agua = 100;
    - Para saber quais são as variáveis do código e seu tamanho etc : whos
    - Função disp()
- Construir programa para calcular seus gastos mensais
    - Calcular quanto você pode gastar por dia se você gastar duas vezes mais em um fim de semana do que você gasta em uma semana
    - Toda vez que lançar uma nova variável, clique em run, precisa rodar todo o programa se não o matlab diz que a variável nova não foi definida.
    - O ; não permite que seja impresso na tela a linha
- Mudar preferencias de cor, ir em *home – preferences – colors*
    - Ajustar de acordo com sua preferência
- Imprimir texto: clc, disp (‘texto’)
- Mostrar uma imagem: figure
    - Vai abrir uma janela vazia, podemos imprimir qualquer coisa nessa janela, por exemplo: imagesc(randn(20)) > imprime uma imagem de 20 numeros aleatórios
    - Podemos mudar a cor da imagem com colormap jet, colormap parula, colormap etc

- Gerar vetores: vetor coluna, vetor linha e matriz de vetor.
  - *vL* = [1 2 4 5 3]; vetor linha usando [ ]
  - *vC* = [1; 2; 5; 6;]; vetor coluna usando ;
  - *vT* = [1 2 4 1]'; vetor coluna obtido pela transposição de um vetor linha
  - vT = transpose ([1 2 4 1]); mesma coisa que anterior, faz a transposição do argumento
  - m = [1 0 2; 5 4 8]; gera matriz, no caso uma matriz 2x3
  - mum = ones (linha, coluna); gera matriz cujos elementos são apenas números um
  - mzero = zeros(linha, coluna); gera matriz zero
  - mrandn = randn (linha,coluna); gera matriz cujos elementos são aleatórios
- Trabalhar com textos
  - Wholetext = 'texto';
  - Wordsep=regexp(wholetext,' ','split'); separar cada palavra em diferentes células.
  - numchars = cellfun(@length,wordsep); Contar quantas letras tem em cada célula, no caso palavra.
  - words2keep = numchars==4; Mostra em true (1) e false (0) as células que contém 4 elementos.
  - words2keep = numchars~=4; O inverso do anterior;
  - wordsep2 = wordsep(words3keep); Retira os false (0) e imprime o true (1)
  - >> strfind(wholetext,'i') Encontra a posição do caracter : digitar no command window
  - namestar = strfind(wholetext,targname); Encontra a posição de uma variável.
  - newtext = [wholetext(1:namestar-1) 'Thais' wholetext(namestar+length(targname):colorstar-1) 'green']; substituição de elementos originais por outros.
  - A ideia até aqui foi encontrar a posição no texto dos elementos que quero substituir para depois conseguir fazer essa substituição e imprimi-la. Trabalhamos com caracteres e strings.
- Mudar algarismos significativos de pi
  - Pi retorna o valor de pi
  - Round(pi) retorna apenas um algarismo significativo de pi
  - Format long retorna mais casas decimais
- Informações de arquivos e pastas
  - A=dir; mostra os elementos que tem no diretório.
  - Escrever A(5) no command window mostra os dados referentes ao arquivo de numero 5.
  - Mkdir namepaste escrito no command window cria uma pasta
  - Ls no command window mostra todos os arquivos na pasta
  - A.isdir mostra quais arquivos são do diretório, só que vai mostrar vários resultados independentes que não são muito úteis.
  - [A.isdir] o mesmo que o anterior, só em formato de um vetor
  - A.name mostra os nomes dos arquivos
  - {everything([everything.isdir]).name} no command window mostra os nomes dos arquivos de diretório no formato de vetor. Usamos {} em vez de [], para que os nomes fiquem separados, caso contrário eles ficam juntos.

    - somethings = dir('*variab*');procura arquivos que tenham em seu nome variab
    - cellfun(@length,foldernames); mostra o comprimento dos elementos referentes a foldernames, que no caso se refere aos diretórios.
- Plotar funções trigonométricas e Gaussianas.
    - O objetivo é traduzir as funções para o MatLab interpretar e plotar.
    - Zoom em command window permite usar uma lupa para ampliar figura
    - Estamos calculando as funções trigonométricas de forma discreta, pois estamos determinando um valor para a variável dependente, no caso, o tempo.
    - plot(x,gaus,'linew',2), o que é esse 2 mesmo?
    - Depois de plotar as curvas, recomenda-se brincar com os valores das variáveis, para ver quais os efeitos na curva.
    - Usamos . em num./den quando dividimos dois vetores, mas quando se está trabalhando apenas com escalar não é preciso.
- Gráfico de Linha
    - Primeiro plotar um gráfico de linhas aleatórias.
    - Função gfc é get current figure (“obter figura atual”)
- Gráficos de barra
    - Apresentar a função bsxfun(@plus,m,v); A ideia é fazer uma soma de dois vetores.
    - Vamos estar plotando gráficos de barras aleatórias, no caso é o valor médio dessas matrizes.
    - ‘xlim’ se refere a limites do eixo x
    - Errorbar(x,y,erro)
- Pontos
    - O objetivo é plotar dados em gráficos de pontos.
    - linspace(x1,x2,n); espaçamento dos n pontos entre x1 e x2.
    - Se digitar h no command window vai aparecer as propriedades da variável h, que no caso é um gráfico.
    - Get(h) no command window vai mostrar todas as propriedades de h.
    - set(h,'LineStyle',':') no command window serve para modificar determinada propriedade da variável h.
- Esfera em um cubo
    - O objetivo aqui é plotar primeiro 12 arestas formando um cubo e depois uma esfera dentro desse cubo.
    - Para desenhar o cubo, a dica é primeiro desenhar linha por linhas. A questão é acertar as coordenas dessas linhas. Isso usando plot3([0 0],[0 1],[0 0],'k','linew',2), etc.
    - Após desenhar as linhas recomendo rotacionar para melhorar a visualização do cubo.
    - Existem duas soluções: redimensionar a esfera para que ela fique dentro do cubo, ou refazer o cubo de acordo com a esfera unitária.
- Cubo colorido:
    - Linspeace(x,y,n) cria vetor linear e [X,Y] = meshgrid(n); cria uma malha para o gráfico.
    - A ideia é usar a função de criar malhas para gráficos para fazer um cubo composto por essas malhas.
- Superfícies Gaussianas Texturizadas

    - imagesc(gaus2d) em command window apenas para verificar se escreveu certo a equação da gaussiana.
    - Usamos surf então para gerar uma superfície, onde a altura, eixo z, é representada pela gaussiana. A cor é redundante quando se trata de altura. Surf(x,y,z).
    - Imshow(img) no command window vai mostrar a imagem que tiver com o nome img
- Bola flutuante
    - Pause(.05) é importante dentro while, para que o programa não fique rodando indefinidamente.
- Funções anônimas
    - O objetivo é criar um novo tipo de função no matlab. No caso será uma função que calcula a primeira e segunda derivada.
    - Existem duas formas de fazer uma derivada segunda no matlab: usar a função diff duas vezes na mesma função ou colocar a ordem da derivada desejada
        - Diff(diff(x)) ou
        - Diff(x,2)
    - É importante que tudo tenha o mesmo tamanho, se não o matlab não consegue fazer a operação. Dá uma mensagem de erro com relação ao length.
    - Para criar a nova função precisamos de um novo script. A principal diferença desse script para os outros que usamos até agora, é que este deve começar com function, pois vamos inserir uma nova função no matlab.
    - Devemos colocar ao lado de function o nome da função e suas entradas
        - Function diff(x)
    - Como estamos criando uma nova função, é importante detalhar no script como comentário para que serve essa função, e quais os tipos de dados de entradas que o usuário pode colocar.
    - Digitando help nomedoarquivo.m obtemos as informações contidas no código.
- Funções simbólicas
    - O objetivo é poder analisar equações matemáticas sem ter dados para as variáveis.
    - Não é recomendável usar ezplot
    - Definimos uma função anônima, nome anterior embutida, como uma função qualquer que é definida usando o símbolo @(entradas) e definição da função
        - Fun = @(x) x.^2;
        - F = @(x,y) x+y;
    - Quando há algo interessante, ou importante, acontecendo com a sua função o fplot foca o gráfico nessa região interessante para melhor visualização e análise.
    - fimplicit(f)plota a função implícita definida por f(x,y)
    - ezpolar(fun)plota a curva polar rho = fun(theta)sobre o domínio padrão 0 theta<<2π.
- Limites de funções
    - O objetivo é calcular o limite de funções usando o matlab.
    - Para esses tipos de operações é necessário gerar uma função que seja simbólica, syms variável. Isso significa que não vai existir

necessariamente um valor pré-definido para essa variável. Então quando for calcular o limite de uma expressão matemática, iremos obter a expressão algébrica, e não um resultado numérico.
        - Syms x
    - Caso seja desejo plotar determinada função, é recomendado estipular um intervalo.
    - Limit(fx,x,3) é um exemplo de como calcular o limite da função fx, onde a variável de interesse x tende a 3.
- Derivada de Funções
    - O objetivo agora é aprender a calcular derivadas de funções no matlab, sendo estas funções algébricas.
    - Usamos de novo syms variável.
    - Podemos transformar uma função em uma string usando char(função). Fazemos isso para conseguirmos criar uma legenda para gráficos compostos de mais de uma curva.
- Integração de funções
    - O objetivo é sair integrando funções usando a função int (fun) do matlab.
    - Para resolver uma integral definida usamos vpa ou escrevemos de novo no command window
        - Int(exp(x),1,2)
            - Ans = exp(2) – exp(1)
            - exp(2) – exp(1)
            - ans = valor da coisa
        - vpa(int(exp(x),1,2))
            - ans = valor da coisa
    - q = integral(fun,xmin,xmax)numericamente integra a função fun de xmin para xmax usando quadratura adaptativa global e tolerâncias de erro padrão.
- Resolvendo equações diferenciais
    - O objetivo é mostrar como trabalhar e resolver equações diferenciais no matlab
    - Encontramos a solução geral com dsolve(fun)
    - Sabemos que não existe uma solução única para edo’s, mas sim uma solução geral que depende da constante de integração C1,k,etc.
    - Se quisermos uma solução particular devemos anunciar no código as condições inicias do problema. dsolve(eq,y(0)==2)
    - Quiver gera vetores de velocidade
    - Lembrando que (:) é capaz de vetorizar matrizes.
    - quiver(x,y,u,v) plota vetores como setas nas coordenadas especificadas em cada par correspondente de elementos em x e y. As matrizes x, y, u, e v devem ser todos do mesmo tamanho e contêm componentes de posição e de velocidade correspondente. No entanto, x e y também podem ser vetores. Por padrão, as setas são dimensionadas para não se sobreporem, mas você pode dimensioná-las para serem maiores ou menores, se desejar.
    - [X,Y] = meshgrid(x,y)retorna coordenadas da grade 2D com base nas coordenadas contidas nos vetores x e y. X é uma matriz em que cada linha é uma cópia de x e Y é uma matriz em que cada coluna é uma cópia de y. A grade representada pelas coordenadas X e Y tem length(y)linhas e length(x)colunas.

- Referências:
  - COHEN, Mike X; BUCHALKA, Tim. **Master MATLAB through Guided Problem Solving**. Disponível em: https://www.udemy.com/course/master-matlab-through-guided-problem-solving/. Acesso em: 02 mar. 2020.