UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

# ELT 410 – PROCESSAMENTO DIGITAL DE SINAIS
# RELATÓRIO REFERENTE À PRÁTICA 1

MATHEUS MORAES PEREIRA SALES DE AZEVEDO - 99892
matheus.m.sales@ufv.br

**Resumo:** Este relatório fundamenta-se em realizar um estudo de comandos, funções e aplicações básicas do software interativo MATLAB. Além disso, há uma breve análise dos resultados obtidos.
**Palavras-chave:** funções, matriz, MATLAB.

## Introdução

Criado pelo norte americano Cleve Moler, o software iterativo de alto desempenho *Matrix Laboratory* (MATLAB) [1], proporciona aos usuários uma muitos recursos dentro do processamento de sinais, utilizado para análise e criação de dados. Utiliza uma linguagem de programação de alto nível com o intuito de tornar mais simples, prática e objetiva a construção de gráficos, análise numérica, processamento de sinais e cálculo com matrizes. Sendo assim, uma importante ferramenta na área da engenharia e afins. Dessa forma, esse relatório tem como objetivo conhecer um pouco mais sobre essa ferramenta.

## Materiais e métodos

Iniciando os procedimentos requeridos pelo roteiro da prática 2, foram criados dois vetores da seguinte forma; *x = 0 : 0.2 : 3;* (valores variando entre 0 e 3 com intervalos de 0.2) e *y = exp(-x) + sin(x);* (y como uma função dependente dos valores de x). Por conseguinte, esses valores foram armazenados, através do comando *save('z.mat','z'),* em uma pasta *z* no *workspace*. Logo após, com o comando *rand();*, foi criada uma matriz quadrada de ordem seis com termos aleatórios e dessa matriz foi extraída uma matriz B de ordem menor (2x2) selecionando determinados termos da primeira. Com o comando *disp* é possível visualizar B.

**Verificando afirmativas:** Nessa etapa da pratica foram realizados testes com o intuito de verificar diferentes formas de realizar operações com matrizes usando comandos do software MATLAB. Para que seja possível a verificação, foi criado um vetor de valores para x variando de 1 a 10 em intervalos de uma unidade. Por conseguinte, usando a lógica *if* e *else*, foram testadas as afirmativas na ordem proposta pelo roteiro, que serão enumeradas de 1 a 4 neste documento. Caso as afirmativas forem verdadeiras, o código retornará ao usuário a letra V e caso seja falsa retornará a letra F.

**Segundo grau:** Essa função, designada de segundograu(a,b,c); irá retornar ao usuário as raízes de uma equação de segundo grau cujo os coeficientes são fornecidos pelo mesmo. Para que uma equação seja considerada como sendo do segundo grau é necessário que existam duas incógnitas, x por exemplo, em que uma delas seja de grau 2 *($x^2$)* ficando no seguinte formato $ax^2+bx^1+c$. Dessa forma, é evidente que para isso ocorrer, é crucial que o termo "*a" que acompanha* a incógnita $x^2$ não seja nulo. A vista disso, a função logo em seu início de código verifica, através do comando *if* se *a==0,* caso seja verdadeiro a igualdade, o programa retornará, com o comando *disp*, a seguinte mensagem "*===A equação não e de segundo grau===".* caso seja falso o programa seguirá normalmente para realizar os cálculos. Para encontrar as raízes da equação a função usará a fórmula de *Bhaskara*. Ademais, a função irá classificar as raízes (com os comandos *if*, *switch* e *case)* como sendo reais e iguais, reais e distintas e complexas conjugadas; de acordo com o valor encontrado no delta da fórmula de *Bhaskar*a

**Função Fatorial:** Foi elaborada uma função nomeada fat() que, dado um número fornecido de entrada pelo usuário (p), irá calcular o valor fatorial do mesmo e então retorná-lo para o utilizador. Primeiramente, o fatorial de um número só se é possível ser calculado se o número for natural inteiro positivo [2]. Portanto, é necessário que o programa garanta que a entrada cumpra com os requisitos supracitados. Dessa forma, o cálculo matemático do fatorial, dado pela multiplicação desse número por todos os seus antecessores até chegar ao número 1, foi incrementado em um loop finito while que só será verdadeiro uma vez que a entrada cumpra com os requisitos, ou seja, n>0. Ademais, temos para o número zero uma forma diferente de determinar o seu fatorial, mas como o valor já se é conhecido (0! = 1), é preciso que a função retorne para o usuário o valor 1 caso a entrada fornecida seja 0. Para desempenhar tal papel foi usado o comando if.

**Soma de duas Matrizes:** Essa função denominada *soma_matriz(A,B)* realiza a soma de duas matrizes fornecidas pelo utilizador. Temos como propriedade de matrizes que, a soma de uma matriz A com uma B é dada pela soma termo a termo de ambas as matrizes. Dessa forma, é notória a necessidade de que as matrizes sejam de mesma ordem, ou seja, possuam o mesmo número de linhas quanto de colunas gerando assim, uma de mesma ordem das demais. Consequentemente, é indispensável que, antes da função realizar o cálculo, verifique se as matrizes fornecidas atendem aos requisitos supramencionados. Para isso, foi utilizado o comando *size()* que retorna em forma vetorial a ordem da matriz. Uma vez que as duas matrizes atendem aos requisitos

simplesmente é realizado a soma *(A+B)*, entretanto foi empregado o comando *while* que será verdadeiro quando as matrizes não atenderem as premissas e então pede para o usuário que entre novamente com os valores dos termos das matrizes, e esse loop só será quebrado quando for fornecido valores coerentes.

**Multiplicação de duas matrizes:** Intitulada como *multpl_matriz(A,B)*, essa função irá retornar dado duas matrizes A e B, a multiplicação entre ambas. Carecemos saber que, para realizar a multiplicação entre duas matrizes é crucial, dada uma multiplicação entre uma matriz A com B, ou seja, *A*B*, o número de colunas de A tem que ser obrigatoriamente igual ao número de linhas de B. Lembrando que multiplicação de matrizes não são comutativas. Por conseguinte, é evidente que a função criada realize a averiguação das matrizes fornecidas pelo usuário para certifica-se que atendem aos requisitos aludidos anteriormente. Para isso, novamente através do comando *size()* obtemos os valores do número de linhas e colunas das matrizes disponibilizadas. Uma vez que, ambas atendem aos requisitos, simplesmente é feita a multiplicação entre ambas. Foi implementado o comando *while* que será verdadeiro quando o número de linhas e colunas forem diferentes, retornando para o usuário a seguinte frase *"Impossível multiplicar"* e pedindo que entre novamente com os valores.

**Sistema linear:** Essa função irá retornar a resolução de um sistema linear de três variáveis. Dentre as diversas formas de resolver sistemas lineares, a escolhida para essa prática foi a regra de *Cramer*, que diz que os valores das incógnitas de um sistema linear são dados por frações cujo o denominador é o determinante da matriz dos coeficientes das incógnitas e o numerador é o determinante da matriz dos coeficientes das incógnitas após a substituição de cada coluna pela coluna que representa os termos independentes do sistema. Dessa forma, primeiramente criada duas matrizes de zeros armazenadas em *A(3x3) e B(3x1*; matriz resposta), por conseguinte, foi pedido ao usuário que fornecesse os termos da matriz dos coeficientes, para isso foi usado o comando *for* que percorrer a matriz A e modifica termo a termo pelos valores fornecidos. Da mesma forma, foi criado a matriz B.

Na regra de Cramer um sistema linear possui três classificações; sistema possível indeterminado, sistema possível determinado e sistema impossível [3]. Da mesma forma, essa função retornará uma das classificações de acordo com os valores dos determinantes, e para isso, foram usados os comandos *if e else.* E por fim, o programa calcula os valores das variáveis e retorna-los para o usuário.

## Resultados

Com a criação das lógicas condicionais, a fim de verificar a veracidade das afirmativas, executou-se as linhas de código referentes a terceira parte do roteiro, obtendo-se resultados afirmativos no caso das sentenças 1 e 3; e negativas em 2 e 4.

Após finalizar todas as funções em *scripts* diferentes da plataforma principal, fez-se necessária a realização de testes das mesmas. Desta maneira, foram executadas as funções *fat(n), soma_matriz(A,B), segundograu(a,b,c) multpl_matiz(A,B) e s_lin()*, com valores aleatórios, observando-se que todas estas apresentaram resultados coerentes para os cálculos previstos.

## Discussão

Em primeira análise, pode-se afirmar que as funções criadas apresentaram um nível de eficiência bastante satisfatório, uma vez que, em muitos casos, o programa consegue prever alguns erros do usuário e força-lo a introduzir os dados de maneira coerente, como no exemplo da função *fat(n)* que imprime na tela mensagens de erro quando são introduzidos dados incoerentes para a definição de tal função, que neste caso se restringe à números decimais ou negativos.
Ademais, é possível inferir que algumas funções apresentaram um grau de informação interessante, pois permite ao usuário praticar e testar algumas aplicações matemáticas como a regra de *Cramer*, uma vez que, embora o objetivo principal seja retornar as soluções das equações, o programa também consegue fornecer a classificação do sistema linear.

## Conclusão

O software se mostrou muito eficiente em trabalhos envolvendo manipulação de matrizes e vetores, haja visto que o MATLAB apresenta certa facilidade em operações abrangendo tais entes matemáticos como a concatenação de matrizes e declaração das mesmas. Em face a esta ideia, outra ferramenta muito útil trabalhada nesta prática foi a criação de funções, possibilitando o encurtamento de muitos códigos de programação e facilitando a manipulação por parte do usuário.

## Referências

[1] MATLAB, Disponível em <https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html>
Acesso em 03 de setembro de 2019.
[2] INEP, Disponível em <https://www.educamaisbrasil.com.br/enem/matematica/fatorial>
Acesso em 02 de setembro de 2019.
[3] B.R.OLIVEIRA, V.C.S.FERREIRA, F.A.O.CRUZ, Sistemas lineares.