

Laboratório

Cálculo Numérico

SUMÁRIO

SUMÁRIO.....	2
ÍNDICE DE FIGURAS.....	5
ÍNDICE DE TABELAS.....	5
ÍNDICE DE PROGRAMAS.....	5
ÍNDICE DE EQUAÇÕES.....	5
INTRODUÇÃO.....	6
AULA PRÁTICA 1 - Ambiente de programação Octave.....	7
OBJETIVO:.....	7
INTRODUÇÃO.....	7
Prompt de comando do Octave.....	7
Operações básicas e Expressões Logicas.....	8
Constantes e Variáveis.....	9
Comentário e pontuações.....	11
Variáveis literais.....	11
Obtendo informações da área de trabalho.....	12
Funções matemáticas.....	12
Comando de ajuda.....	13
Números Complexos.....	15
VETORES E MATRIZES.....	16
Matrizes.....	19
Operações com matrizes.....	19
Gráficos no Octave.....	23
ATIVIDADES.....	28
AULA PRÁTICA 2 - SCRIPTS E FUNÇÕES NO OCTAVE.....	32
OBJETIVO.....	32
INTRODUÇÃO.....	32
Controles de Fluxo.....	35
Subprograma function.....	41
ATIVIDADES.....	42
AULA PRÁTICA 3 - Decomposição LU, Cholesky e LDL^T	43
OBJETIVO.....	43
INTRODUÇÃO.....	43
Método de Gauss.....	43
Decomposição LU.....	43
Decomposição de Cholesky.....	44
Decomposição LDL^T	44
ATIVIDADES.....	44
AULA PRÁTICA 4 - Métodos iterativos estacionários.....	2
OBJETIVO.....	2

INTRODUÇÃO.....	2
Condições de convergência.....	2
Métodos iterativos estacionários.....	3
ATIVIDADES.....	3
AULA PRÁTICA 5 - Análise de erros na solução de sistemas lineares.....	4
OBJETIVO.....	4
INTRODUÇÃO.....	4
ATIVIDADES.....	4
AULA PRÁTICA 6 - INTERPOLAÇÃO POLINOMIAL.....	6
Introdução.....	6
Atividades.....	6
AULA PRÁTICA 7 - INTERPOLAÇÃO UTILIZANDO SPLINES.....	9
INTRODUÇÃO.....	9
Splines cúbicos naturais.....	9
Splines cúbicos extrapolados.....	9
ATIVIDADES.....	9
AULA PRÁTICA 8 - Ajuste de curvas.....	11
INTRODUÇÃO.....	11
REGRESSÃO LINEAR SIMPLES.....	11
ATIVIDADES.....	11
AULA PRÁTICA 9 - Regressão linear multivariáveis e ajuste polinomial.....	13
OBJETIVO.....	13
INTRODUÇÃO.....	13
ATIVIDADES.....	13
AULA PRÁTICA 10 - Integração numérica.....	15
OBJETIVO.....	15
INTRODUÇÃO.....	15
ATIVIDADES.....	15
AULA PRÁTICA 11 - Integração numérica método de Gauss-Legendre.....	16
OBJETIVO.....	16
INTRODUÇÃO.....	16
ATIVIDADES.....	16
AULA PRÁTICA 12 - Integração dupla pelo método de Newton-Cotes.....	17
OBJETIVO.....	17
ATIVIDADES.....	17
AULA PRÁTICA 13 - Raízes de Equações.....	18
OBJETIVO.....	18
INTRODUÇÃO.....	18
Método de Horner.....	18
Limites das raízes reais.....	18
ATIVIDADES.....	18
AULA PRÁTICA 14 - Equações Diferenciais Ordinárias.....	20
OBJETIVO.....	20

INTRODUÇÃO.....	20
ATIVIDADES.....	20
Bibliografia.....	22

ÍNDICE DE FIGURAS

Figura 1 – <i>Prompt</i> de comando do Octave.....	8
Figura 2 - Opções da aba Arquivo.....	8
Figura 3 - Exemplo de comando errado no <i>prompt</i> do Octave.....	9
Figura 4 - Correção de comando no <i>prompt</i>	9
Figura 5 - Exemplo de operadores relacionais.....	10
Figura 6 - Exemplo de gráfico bidimensional linear com uma variável e índice.....	25
Figura 7 - Gráfico do seno de t.....	25
Figura 8 - Múltiplos gráficos em apenas um eixo.....	26
Figura 9 - Gráfico do seno impresso na posição à esquerda da figura.....	26
Figura 10 – Gráfico do seno impresso na posição esquerda da figura e do cosseno impresso na posição direita.....	27
Figura 11 - Gráficos de tipos diferentes em eixos diferentes.....	27
Figura 12 - Gráfico do seno com rótulos para os eixos e nome para gráfico.....	29
Figura 13 - Gráfico exemplo de script.....	43

ÍNDICE DE TABELAS

Tabela 1 - Lista de comandos das teclas.....	8
Tabela 2 - Operações aritméticas básicas.....	9
Tabela 3 – Operações relacionais.....	9
Tabela 4 - Operadores lógicos relacionais.....	9
Tabela 5 - Formatos de números do prompt de comando do Octave.....	10
Tabela 6 - Regras para construção de nomes de variáveis.....	10
Tabela 7 - Variáveis especiais do Octave.....	11
Tabela 8 - Comentários e pontuação.....	11
Tabela 9 - Algumas funções matemáticas.....	14
Tabela 10 - Construção de vetores.....	16
Tabela 11 - Operações vetoriais.....	18
Tabela 12 - Algumas funções de matrizes.....	23
Tabela 13 - Tipos de linhas dos gráficos.....	27
Tabela 14 - Marcadores de pontos nos gráficos.....	27
Tabela 15 – Cores das linhas e dos marcadores nos gráficos.....	27
Tabela 16 - Lista de comandos para arquivos de <i>script</i> Octave.....	34

ÍNDICE DE PROGRAMAS

%Programa 1 - Exemplo de script do Octave.....	32
Programa 2 – Exemplo 1 de utilização da função input.....	34
Programa 3 – Exemplo 2 de utilização da função M-file “input”.....	34
Programa 4 – Exemplo 3 de utilização da função M-file “input”.....	34

ÍNDICE DE EQUAÇÕES

Equação 1 - Sistema linear.....	43
Equação 2 - Decomposição LU.....	43
Equação 3 - Sistema linear LU.....	43
Equação 4 - Decomposição de Cholesky.....	44
Equação 5 - Diagonal principal da de decomposição de Cholesky L.....	44
Equação 6 - Termos inferiores da matriz L de Cholesky.....	44
Equação 7 - Sistema linear de Cholesky.....	44
Equação 8 - Métodos iterativos.....	2
Equação 9 - Equação de convergência.....	2
Equação 10 - Condição de convergência dos métodos de Jacobi e Gauss-Seidel.....	3

INTRODUÇÃO

O cálculo numérico é uma ferramenta fundamental na engenharia moderna, já que aumenta a cada dia a utilização dos computadores para cálculos cada vez mais elaborados. Os computadores surgiram na década de 40 justamente com o objetivo de melhorar a capacidade das pessoas em resolver problemas matemáticos. Os computadores inicialmente utilizavam apenas válvulas como unidade fundamental e estes componentes eram grandes e apresentavam um aquecimento muito grande. Devido a este fato foram escolhidas apenas as operações básicas para serem realizadas por estes computadores. As operações escolhidas foram, aritméticas, soma e subtração e as operações lógicas AND, OR, XOR e NOT. Então para que estas máquinas fossem utilizadas para o fim que foram criadas que era a utilização para realizar cálculos matemáticos com um tempo menor foi preciso desenvolver uma ferramenta matemática, esta ferramenta matemática é o cálculo numérico. Desde então computadores mais modernos foram criados e novas funcionalidades foram sendo descobertas para estes equipamentos. Computadores modernos são capazes de fazer um número muito maior de operações do que os primeiros eram capazes. Existem computadores atuais que são capazes de realizar mais de 500 operações diferentes entre elas operações aritméticas mais elaboradas como multiplicações, divisões, raiz quadrada entre outras.

O cálculo numérico surgiu então com o objetivo de através de funções matemáticas adequadas modelar um problema para que este seja transformado em um algoritmo que possa ser executado por um microprocessador. Com o passar do tempo os microprocessadores foram ganhando novas funções como descrito acima e passou a realizar as funções elementares aritméticas todas. Mesmo com este crescimento o cálculo numérico ainda é essencial. Quanto mais os computadores se popularizam para soluções de problemas mais pessoas utilizam estas máquinas e seguem então novas necessidades. Este fato já foi chamado de buraco branco do tempo. Então as ferramentas de cálculo numérico também se modernizaram para acompanhar o desenvolvimento destas máquinas e quando antes eram utilizadas para resolver problemas simples como encontrar a raiz quadrada de um número agora são utilizadas para resolver problemas muito mais complexos como previsão da colisão de galáxias.

Por todos os motivos aqui descritos acredito que esta disciplina seja de fundamental importância a todos os estudantes de engenharia que têm como objetivo principal de seus cursos fazer contas e atualmente fazer contas utilizando o computador. Então acredito que o aluno será tão melhor engenheiro quanto mais ele souber fazer contas utilizando estas máquinas. Daí a necessidade de se ter uma disciplina que relacione o cálculo numérico com a programação de computadores e é isto que espero da disciplina que seja uma integração destas duas ferramentas poderosas.

AULA PRÁTICA 1 - Ambiente de programação Octave

OBJETIVO:

- ✓ Introduzir o ambiente de programação Octave.

INTRODUÇÃO

O Octave é um "software" de alto desempenho destinado a fazer cálculos com matrizes e vetores, podendo funcionar como uma calculadora ou como uma linguagem de programação científica (FORTRAN, Pascal, C, etc.). Entretanto, os comandos do Octave são mais próximos da forma como escrevemos expressões algébricas, tornando mais simples o seu uso. Atualmente, o Octave é definido como um sistema interativo e uma linguagem de programação para computação técnica e científica em geral, integrando a capacidade de fazer cálculos, visualização gráfica e programação.

Prompt de comando do Octave

O programa será aberto clicando 2 vezes sobre o ícone do Octave. Você observará como na Figura 1 a janela a parte da janela referente apenas ao *prompt* de comando.

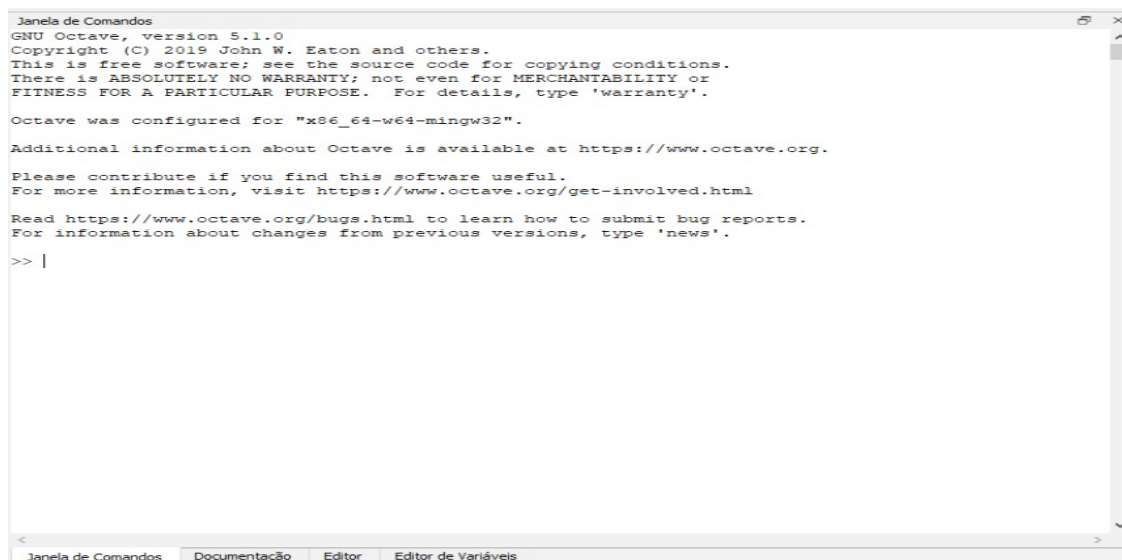


Figura 1 – *Prompt* de comando do Octave

Através do *prompt* de comando é possível realizar qualquer operação ou executar qualquer função ou programa criado para o Octave. Também no *prompt* de comando é possível visualizar os resultados das operações realizadas. Na Figura 2 podemos observar uma operação de soma de dois números 10+15 e o resultado desta soma mostrado logo abaixo.

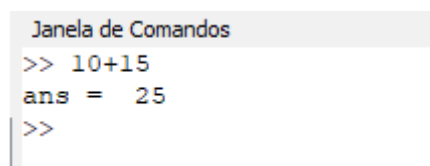


Figura 2 - Opções da aba Arquivo

As teclas com setas podem ser usadas para se encontrar comandos dados anteriormente, para execução novamente ou sua reedição. Por exemplo, suponha que você entre com o comando `sen(0)` como mostrado na Figura 3 aparecerá uma resposta indicando que o comando é indefinido pois não existe a função `sen(0)`.

```
Janela de Comandos
>> sen(0)
error: 'sen' undefined near line 1 column 1
>> |
```

Figura 3 - Exemplo de comando errado no *prompt* do Octave

Isto acontece porque para se determinar o seno de um ângulo é necessário digitar em inglês o comando *sin*. No lugar de rescrever a linha inteira, simplesmente pressione a tecla “seta para cima”. O comando errado retorna, e você pode, então, é possível mover o cursor para trás usando a tecla “seta para esquerda” ou o ponto de inserção com o “mouse” ao lugar apropriado para inserir a letra *i* alterando o comando que agora funcionará corretamente como mostrado na Figura 4.

```
Janela de Comandos
>> sin(0)
ans = 0
>> |
```

Figura 4 - Correção de comando no *prompt*











Tecla	Descrição
	Retorna a linha anterior
	Retorna a linha posterior
	Move um espaço para esquerda
	Move um espaço para direita
	Move uma palavra para direita
	Move uma palavra para esquerda
	Move para o começo da linha
	Move para o fim da linha
	Apaga um caractere à direita
	Apaga um caractere à esquerda

Tabela 1 - Lista de comandos das teclas

Note que o Octave chamou o resultado de *ans* (*answer*=resposta). Além das teclas com setas, podem-se usar outras teclas para reeditar a linha de comando, como mostrado na Tabela 1.

Operações básicas e Expressões Lógicas

O Octave oferece as operações aritméticas básicas mostradas na Tabela 2.

A ordem nas expressões segue a ordem matemática – potência, seguida da multiplicação e da divisão, que por sua vez são seguidas pelas operações de adição e subtração. Parêntesis podem ser usados para alterar esta ordem. Neste caso, os parêntesis mais internos são avaliados antes dos mais externos.

Operação	Símbolo	Exemplos
Adição, $a+b$	+	5+6
Subtração, $a-b$	-	19-4.7
Multiplicação, $a.b$	*	5.02 * 7.1
Divisão, $a \div b$	/ ou \	45/5 ou 5\45
Potência, a^b	^	3.02^0.35

Tabela 2 - Operações aritméticas básicas

Uma expressão se diz lógica se os operadores são lógicos e os operandos são relações e/ou variáveis do tipo lógico. Os operadores relacionais realizam comparações entre valores do mesmo tipo. Os operadores relacionais utilizados pelo Octave são mostrados na Tabela 3

Operador relacional	Descrição
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a
==	Igual a
~=	Diferente de

Tabela 3 – Operações relacionais

Note que (=) é usado para atribuição de um valor a uma variável, enquanto que (==) é usado para comparação de igualdade. No Octave os operadores relacionais podem ser usados para comparar vetores de mesmo tamanho ou escalares. O resultado de uma relação ou de uma expressão lógica é verdadeiro ou falso; contudo, no Octave o resultado é numérico, sendo que 1 significa verdadeiro e 0 significa falso. Como mostrado na Figura 5:

Figura 5 - Exemplo de operadores relacionais

Os operadores lógicos permitem a combinação ou negação das relações lógicas. Os operadores lógicos relacionais do Octave são mostrados na

Operador lógico	Descrição	Uso
&	E	Conjunção
	OU	Disjunção
~	NÃO	Negação

Tabela 4 - Operadores lógicos relacionais

Constantes e Variáveis

O Octave faz cálculos simples e científicos como uma calculadora. Para tal, os comandos devem ser digitados diretamente no *prompt(>>)* do Octave, já que este se trata de um software interativo. Por exemplo:

```
>> 3*25 + 5*12
ans =
135
```

Observe que no Octave a multiplicação tem precedência sobre a adição. Uma constante numérica no Octave é formada por uma sequência de dígitos que pode estar ou não precedida de um sinal positivo (+) ou negativo (-) e pode conter um ponto decimal (.). Esta sequência pode terminar ou não por uma das letras e, E, d ou D, seguida de outra sequência de dígitos precedida ou não de um sinal de (+) ou de (-). Esta segunda sequência é a potência de 10 pela qual a primeira sequência deve ser multiplicada. Por exemplo,

```
>> 1.23e-1
```

significa 0,123.

O formato em que uma constante numérica é mostrada no Octave segue, como opção padrão, os seguintes critérios: se um resultado é inteiro, o Octave mostra o número como inteiro; quando o resultado é real, o Octave mostra o número com 5 dígitos à direita do ponto decimal; se os dígitos do resultado estiverem fora desta faixa, o Octave mostra o resultado usando a notação científica. Este padrão pode, entretanto, ser modificado utilizando-se o comando *format* mais o valor do formato conforme tabela abaixo. Considere a Tabela 5 com exemplos dos formatos numéricos do Octave:

Formato	Exemplo	Comentário
short	33.50000	5 dígitos decimais (padrão)
long	33.5000000000000000	16 dígitos
short e	3.3500e+001	5 dígitos mais expoente
long e	3.3500000000000000e+001	16 dígitos mais expoente
short g	3.3500e+001	Mesma coisa que short e mas não imprime zeros não significativos
Long g	3.3500000000000000e+001	Mesma coisa que long e mas não imprime zeros não significativos
short eng	123.00e+003	5 dígitos mais expoente divisível por 3
long eng	123.00000000000000e+003	16 dígitos mais expoente divisível por 3
hex	4040c00000000000	Hexadecimal
bank	33.50	2 dígitos decimais
+	+	Positivo, negativo ou zero
rational	67/5	Racional

Tabela 5 - Formatos de números do prompt de comando do Octave

As variáveis são utilizadas para armazenar informação. Por exemplo:

```
>> q1=3, p1=25, q2=5, p2=12
q1 = 3
p1 = 25
q2 = 5
p2 = 12
>> total=q1*p1+q2*p2
total =
135
```

Primeiro, criamos quatro variáveis, q1, p1, q2 e p2, atribuindo a elas valores. Observe que o sinal de igual (=) aqui significa atribuição. O que estiver à direita do sinal de igual é “colocado” na variável que estiver à esquerda. Finalmente, criamos uma variável chamada total que recebeu o total o resultado das operações entre as variáveis.

Os nomes das variáveis devem consistir-se de uma única palavra, conforme as regras expressas na Tabela 6:

Regras de construção das variáveis	Comentários/Exemplos
Variáveis com letras minúsculas são diferentes de variáveis com as mesmas letras mas com qualquer uma maiúscula.	Tota ≠ total ≠ TOTAL e ≠ ToTaL são variáveis diferentes
As variáveis podem consistir de até 19 caracteres	Contador_de_pontosG
As variáveis devem começar com uma letra e não podem conter caracteres especiais exceto <i>underline</i> (“_”).	Contador_de_pontosG, Var_1, p_2_1

Tabela 6 - Regras para construção de nomes de variáveis

As variáveis podem ser redefinidas a qualquer momento, bastando para isso atribuir-lhes um novo valor. Variáveis em Octave não possuem tipo podendo assim ser atribuídas valores a elas de qualquer tipo.

Alguns nomes são usados para variáveis predefinidas, ou seja, são variáveis especiais do Octave, portanto não é aconselhável utilizar estes nomes para variáveis criadas pelo usuário. Estas variáveis estão mostradas na Tabela 7.

Variáveis especiais	Significado
ans	Variável usada para exibir os resultados que não são atribuídos a nenhuma variável
pi	Número pi
eps	Menor número tal que, quando adicionado a 1, cria um número maior que 1 no computador
Inf ou Inf	Significa infinito
NAN ou nan	Significa que o resultado não é definido como por exemplo 0/0 ou inf/inf
i e j	Unidade imaginária [$\sqrt{-1}$]
nargin	Número de argumentos de entrada de uma função
nargout	Número de argumentos de saída de uma função
realmin	Menor número que o computador pode armazenar
realmax	Maior número que o computador pode armazenar

Tabela 7 - Variáveis especiais do Octave

Comentário e pontuações

Exemplo dos caracteres mostrados na Tabela 8:

```
>> q1=3, p1=25, ...
q2=5; p2=12; %Exemplo de uso da vírgula, ponto e vírgula e três pontos
q1 =
3
p1 =
25
```

Os espaços em branco entre os operadores (aritméticos, lógicos, relacionais) e as variáveis (ou constantes) são opcionais. O mesmo para vale para a vírgula, o ponto e vírgula e o símbolo de porcentagem. No entanto, o espaço em branco entre a última variável (ou constante) de uma linha e os três pontos é obrigatório (veja exemplo anterior).

Símbolo	Função
,	Separa comandos dados em uma mesma linha. Separa as colunas de uma matriz
;	Separa comandos dados em uma mesma linha. Se o último caractere da declaração é um ponto e vírgula a impressão na tela é suprimida, mas a tarefa é realizada. Separa as linhas de uma matriz.
%	Todo e qualquer caractere depois do símbolo de porcentagem é tomado como comentário
...	Pode-se continuar uma certa expressão na próxima linha usando espaço em branco e três pontos "...", ao final das linhas incompletas.

Tabela 8 - Comentários e pontuação

Variáveis literais

Uma variável pode conter uma cadeia de caracteres em vez de um número. Estes caracteres são manipulados como vetores linha(assunto que será tratado mais adiante). A cadeia de caracteres deve estar limitada por apóstrofes ('cadeia de caracteres') para ser atribuída a uma variável literal.

Por exemplo:

```
>> a='Octave'
a =
Octave
```

Obtendo informações da área de trabalho

Os exemplos de declarações mostrados nos itens acima criaram variáveis que são armazenadas na Área de Trabalho (*Workspace*) do Octave. Executando

```
>> who
```

obtêm-se uma lista das variáveis armazenadas na Área de Trabalho:

```
Variables in the current scope:
```

```
a      ans  p1   p2   q1   q2
```

Que mostra as seis variáveis geradas em nossos exemplos anteriores, incluindo *ans*. Uma informação mais detalhada mostrando a dimensão de cada uma das variáveis correntes é obtido com *whos* que para nosso exemplo produz:

```
>> whos
```

```
Variables in the current scope:
```

Attr	Name	Size	Bytes	Class
====	====	====	=====	=====
	a	1x6	6	char
	ans	1x1	8	double
	p1	1x1	8	double
	p2	1x1	8	double
	q1	1x1	8	double
	q2	1x1	8	double

```
Total is 11 elements using 46 bytes
```

Em qualquer momento, podemos ver o valor que está contido em uma variável, simplesmente digitando no *prompt* o seu nome.

```
>> total
total =
135
```

As variáveis no espaço de trabalho podem ser removidas incondicionalmente usando o comando *clear*. Por exemplo:

```
>> clear p2 %remove a variávelep2
>> clear %remove todas as variáveis do espaço de trabalho
```

O comando *save* é usado para gravar as variáveis do espaço de trabalho em um arquivo (.mat) em disco. O comando *load* é usado para recuperar os dados gravados em um arquivo pelo comando *save* e colocá-los no espaço de trabalho. Maiores informações a respeito da sintaxe destes comandos pode ser obtida através do comando *help*, a ser tratado posteriormente. O comando *clc* limpa a janela de comandos e coloca o cursor na posição inicial.

Funções matemáticas

O Octave tem uma série de funções científicas pré-definidas. A palavra função no Octave tem um significado diferente daquele que tem na Matemática. Aqui, função é um comando, que pode ter alguns argumentos de entrada e alguns de saída. Algumas dessas funções são intrínsecas, ou seja, não podem ser alteradas pelo usuário. Outras funções estão disponíveis em uma biblioteca externa distribuída com o programa original, que são na realidade arquivos com a extensão ".m" criados a partir das funções intrínsecas. A biblioteca externa pode ser constantemente atualizada à medida que novas aplicações são desenvolvidas. As funções do Octave, intrínsecas ou arquivos ".m", podem ser utilizadas apenas no ambiente Octave.

As categorias gerais de funções matemáticas disponíveis no Octave incluem:

- Matemática elementar;
- Funções especiais;
- Matrizes elementares e especiais;
- Decomposição e fatoração de matrizes;
- Análise de dados;
- Polinômios;
- Solução de equações diferenciais;
- Equações não-lineares e otimização;
- Integração numérica;
- Processamento de sinais.

A maioria das funções pode ser usada da mesma forma que seria escrita matematicamente. Por exemplo:

```
>> x=sqrt(2)/2
x =
0.7071
>> y=acos(x)
y =
0.7854
>> y_graus=y*180/pi
y_graus =
45.0000
```

Estes comandos calculam o arco cujo cosseno é $\sqrt{2}/2$, inicialmente em radianos, depois em graus. Na Tabela 9 vemos uma lista de algumas funções científicas disponíveis.

Comando de ajuda

No Octave, pode-se obter ajuda sobre qualquer comando ou função. Isto pode ser feito basicamente de três formas: interativamente através do menu de barras, através do comando *help* ou do comando *lookfor*. Digitando-se simplesmente o comando *help*,

```
>> help
```

O Octave mostra uma listagem de todos os pacotes disponíveis. Ajuda sobre um pacote específico ou sobre um comando ou função específica é obtida com o comando *help* <tópico>, onde tópico pode ser o nome de um pacote, de um comando ou função. Por exemplo:

```
>> help sign
```

```
'sign' is a built-in function from the file libinterp/corefcn/mappers.cc
```

```
-- sign (X)
    Compute the "signum" function.
```

```
    This is defined as
```

```
    -1, x < 0;
```

```
sign (x) = 0, x = 0;
          1, x > 0.
```

For complex arguments, 'sign' returns 'x ./ abs (X)'.

Note that 'sign (-0.0)' is 0. Although IEEE 754 floating point allows zero to be signed, 0.0 and -0.0 compare equal. If you must test whether zero is signed, use the 'signbit' function.

See also: signbit.

Comando	Descrição
abs (x)	Valor absoluto de x.
acos (x)	Arco cujo cosseno é x.
asin (x)	Arco cujo seno é x.
atan (x)	Arco cuja tangente é x
conj (x)	Conjugado complexo
cos (x)	Cosseno de x.
cosh (x)	Cosseno hiperbólico de x.
exp (x)	Exponencial e^x .
floor (x)	Arredondamento em direção ao $-\infty$
gcd (x, y)	Máximo divisor comum de x e y.
lcm (x, y)	Mínimo múltiplo comum de x e y.
log (x)	Logaritmo de x na base e .
log10 (x)	Logaritmo de x na base 10.
rem (x, y)	Resto da divisão de x por y.
round (x)	Arredondamento para o inteiro mais próximo
sign (x)	Retorna apenas o sinal de um número
sin (x)	Seno de x
sinh (x)	Seno hiperbólico de x.
sqrt (x)	Raiz quadrada de x.
tan (x)	Tangente de x.
tanh (x)	Tangente hiperbólica de x.

Tabela 9 - Algumas funções matemáticas

O comando *help* é a maneira mais simples de se obter auxílio no caso do usuário conhecer o tópico em que ele quer assistência. Note que no exemplo mostrado a função *SIGN* está escrita em letras maiúsculas somente para destacar. Deve-se lembrar que todos os comandos do Octave devem ser escritos em letras minúsculas. Portanto, para utilizar esta função deve-se digitar:

```
>> sign (x)
```

O Comando *lookfor* provê assistência pela procura através de todas as primeiras linhas dos tópicos de auxílio do Octave e retornando aquelas que contenham a palavra-chave especificada. O interessante deste comando é que a palavra-chave não precisa ser um comando do Octave. Sua sintaxe é *lookfor <palavra-chave>*, onde palavra-chave é a cadeia de caracteres que será procurada nos comandos do Octave. Por exemplo, para se obter informações sobre funções para se resolver integral:

```
>> lookfor integral
ELLIPKE Complete elliptic integral.
EXPINT Exponential integral function.
DBLQUAD Numerically evaluate double integral.
INNERLP Used with DBLQUAD to evaluate inner loop of
integral.
```

QUAD Numerically evaluate integral, low order method.
 QUAD8 Numerically evaluate integral, higher order method.
 COSINT Cosine integral function.
 SININT Sine integral function.
 ASSEMA Assembles area integral contributions in a PDE problem.
 COSINT Cosine integral function.
 FOURIER Fourier integral transform.
 IFOURIER Inverse Fourier integral transform.
 SININT Sine integral function.
 BLKPIDCON The output of the block is the sum of proportional, integral and

Apesar da palavra integral não ser um comando do Octave, ela foi encontrada na descrição de 14 comandos. Tendo esta informação, o comando *help* pode ser usado para exibir informações a respeito de um comando específico, como por exemplo:

```
>> help quad
```

Números Complexos

Algumas linguagens de programação requerem um tratamento especial para números complexos, o que não é o caso do Octave. Números complexos são permitidos em todas as operações e funções no Octave. Os números complexos são introduzidos usando-se as funções especiais *i* e *j*. Eles podem ser representados de várias maneiras. Por exemplo:

```

>> z1=3+4*i
z1 =
3.0000 + 4.0000i
>> z2=3+4*j
z2 =
3.0000 + 4.0000i
>> z1+z2
ans =
6.0000 + 8.0000i

```

- **Identidade de Euler:** relaciona a forma polar de um número complexo com a sua forma retangular.

$$M \angle \theta \equiv M \cdot e^{j\theta} = a + bj,$$

onde

$$M = \sqrt{a^2 + b^2},$$

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

$$a = M \cdot \cos \theta$$

$$b = M \cdot \sin \theta$$

No Octave, a conversão entre as formas polar e retangular de um número complexo utiliza as seguintes funções:

- **real:** parte real de um número complexo
- **imag:** parte imaginária de um número complexo
- **abs:** calcula o valor absoluto ou módulo de um número complexo
- **angle:** calcula o ângulo de um número complexo

Exemplo:

```

>> x=1-4i
x =
1.0000 - 4.0000i
>> a=real(x)
a =
1

```

```
>> b=imag(x)
b =
-4
>> M=abs(x)
M =
4.1231
>> theta=angle(x)*180/pi
theta = -75.9638
```

VETORES E MATRIZES

O Octave permite a manipulação de linhas, colunas, elementos individuais e partes de matrizes. Na Tabela 10, tem-se um resumo das diversas formas de se construir um vetor no Octave.

Comando	Descrição
X=primeiro : último	Cria um vetor x começando com o valor primeiro, incrementando-se de 1(um) em 1(um) até atingir o valor último ou o valor mais próximo possível de último
X=primeiro : incremento : último	Cria um vetor x começando com o valor primeiro, incrementando-se do valor incremento até atingir o valor último ou o valor mais próximo possível de último.
X=linspace(primeiro,último,n)	Cria um vetor x começando com o valor primeiro e terminado no valor último, contendo n elementos linearmente espaçados.
X=logspace(primeiro,último,n)	Cria um vetor x começando com o valor 10^{primeiro} e terminando no valor $10^{\text{último}}$, contendo n elementos logaritmicamente espaçados.
X=[2 2*pi sqrt(2) 2-3j]	Cria um vetor x contendo os elementos especificados

Tabela 10 - Construção de vetores

Exemplo 1:

```
>> x = 1 : 5
```

Gera um vetor linha contendo os números de 1 a 5 com incremento unitário. Produzindo

```
X =
1 2 3 4 5
>> x=1:10.5
x=
12345678910
```

Exemplo 2:

```
>> z = 6 : -1 : 1
z =
6 5 4 3 2 1
```

Exemplo 3:

Pode-se, também, gerar vetores usando a função *linspace*. Por exemplo,

```
>> k = linspace (0, 1, 6)
K =
0 0.2000 0.4000 0.6000 0.8000 1.0000
```


Gera um vetor linearmente espaçado de 0 a 1, contendo 6 elementos.

```
>> x=linspace(1,10.5,5)
x=
1.0000 3.3750 5.7500 8.1250 10.5000
```

Exemplo 4:

```
>> x=logspace(0,2,5)
x=
1.0000 3.1623 10.0000 31.6228 100.00
```

Exemplo 5:

```
>> x=[8 6 8.10 5-6j]
x=
8.0000 6.0000 8.1000 5.0000-6.0000i
```

Nos exemplos acima os vetores possuem uma linha e várias colunas (vetores linha). Da mesma forma podem existir vetores coluna (uma coluna e várias linhas). Para se criar um vetor coluna elemento por elemento estes devem estar separados por (;). Por exemplo:

```
>> v=[1.5;-3.2;9]
v =
1.5000
-3.2000
9.0000
```

Esses vetores coluna podem também ser criados a partir dos comandos utilizados anteriormente para criar os vetores linha, acompanhados do símbolo ('), que é o operador de transposição. Exemplo:

```
>> y=(1:0.5:3) '
y =
1.0000
1.5000
2.0000
2.5000
3.0000
>> z=[0 -2.3 4 sqrt(33)] '
z =
0
-2.3000
4.0000
5.7446
```

Endereçamento de vetores

No Octave, cada um dos elementos de um vetor podem ser acessados através de seu índice que identifica cada uma das colunas. Por exemplo :

```
>> x=1:10
x=
12345678910
>> x(3) % Acessa o terceiro elemento de x
ans =
3
```

```
>> x(5) % Acessa o quinto elemento de x
ans =
5
```

Esses elementos de um vetor também podem ser acessados em blocos. Por exemplo:

```
>> c=linspace(10,40,7)
c =
10 15 20 25 30 35 40
>> c(3:5) % terceiro a quinto elemento de c
ans =
20 25 30
>>c(5:-2:1) % quinto, terceiro e primeiro elementos de c
ans =
30 20 10
```

O endereçamento indireto também é possível, permitindo referenciar os elementos em qualquer ordem:

```
>> c([4 1]) %quarto e primeiro elementos
ans = 25 10
```

No caso de vetores coluna, os comandos acima funcionam de maneira similar. Por exemplo:

```
>> d=c'
d =
10
15
20
25
30
35
40
17
>> d([4 1]) %quarto e primeiro elementos
ans =
25
10
>> d(5:-2:1)
ans =
30
20
10
```

Operações entre vetores

As operações básicas entre vetores só são definidas quando estes tiverem o mesmo tamanho e orientação (linha ou coluna). Estas operações são mostradas na Tabela 11.

Seja $a = [a_1 \ a_2 \ \dots \ a_n]$, $b=[b_1 \ b_2 \ \dots \ b_n]$ e c um escalar		
operação	expressão	resultado
Adição escalar	$A+c$	$[a_1+c \ a_2+c \ \dots \ a_n+c]$
Adição vetorial	$A+b$	$[a_1+b_1 \ a_2+b_2 \ \dots \ a_n+b_n]$
Multiplicação por escalar	$A*c$	$[a_1*c \ a_2*c \ \dots \ a_n*c]$
Multiplicação elemento a elemento	$a \cdot b$	$[a_1*b_1 \ a_2*b_2 \ \dots \ a_n*b_n]$
Divisão elemento a elemento	$a./b$	$[a_1/b_1 \ a_2/b_2 \ \dots \ a_n/b_n]$
Potenciação por um escalar elemento a elemento	$a.^c$	$[a_1^c \ a_2^c \ \dots \ a_n^c]$
Potenciação de um escalar elemento a elemento por um vetor	$c.^a$	$[c^{a_1} \ c^{a_2} \ \dots \ c^{a_n}]$
Potenciação de um vetor por outro elemento a elemento	$a.^b$	$[a_1^{b_1} \ a_2^{b_2} \ \dots \ a_n^{b_n}]$

Tabela 11 - Operações vetoriais

Matrizes

O Octave trabalha essencialmente com um tipo de objeto, uma matriz numérica retangular (1x1; 2x2; 3x3; i(linha) x j (coluna); etc).

Os elementos de cada linha da matriz são separados por espaços em branco ou vírgulas e as colunas separadas por ponto e vírgula, colocando-se colchetes em volta do grupo de elementos que formam a matriz. Por exemplo, entre com a expressão

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

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

As linhas das matrizes também podem ser definidas através dos comandos utilizados anteriormente para se definir vetores linha. Por exemplo:

```
>> A=[1:3;linspace(4,9,3);0:.5:1]
```

```
A =  
1.00 2.00 3.00  
4.00 6.50 9.00  
0 0.50 1.00
```

Os elementos de uma matriz (ou de um vetor) também podem ser definidos por operações ou funções matemáticas. Por exemplo:

```
>> B=[15 7;sqrt(36) cos(pi/3);12/7 2.5^2]
```

```
B =  
15.0000 7.0000  
6.0000 0.5000  
1.7143 6.2500
```

Operações com matrizes

As operações com matrizes no Octave são as seguintes:

- Transposta;
- Adição;
- Subtração;
- Multiplicação;
- Divisão à direita;
- Divisão à esquerda;
- Exponenciação;

Transposta

O carácter apóstrofo, " ' ", indica a transposta de uma matriz. Considere os exemplos a seguir:

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

```
A=  
1 2 3  
4 5 6  
7 8 0
```

```
>> B = A'
```

```
B = 1 4 7  
2 5 8  
3 6 0
```

```
>> x = [-1 0 2]'
X =
-1
0
2
19
```

Adição e Subtração

A adição e subtração de matrizes são indicadas, respectivamente, por "+" e "-". As operações são definidas somente se as matrizes tiverem as mesmas dimensões. Por exemplo, a soma com as matrizes mostradas acima, $A + x$, não é correta porque A é 3×3 e x é 3×1 . Porém,

```
>> C = A + B
```

é aceitável, e o resultado da soma é

```
C =
2 6 10
6 10 14
10 14 0
```

A adição e subtração também são definidas se um dos operadores é um escalar, ou seja, uma matriz 1×1 . Neste caso, o escalar é adicionado ou subtraído de todos os elementos do outro operador. Por exemplo:

```
>> y = x - 1
```

```
Y =
-2
-1
1
```

Multiplicação

A multiplicação de matrizes é indicada por "*". A multiplicação $x*y$ é definida somente se a segunda dimensão de x for igual à primeira dimensão de y . A multiplicação

```
>> x' * y
```

```
ans =
4
```

É evidente que o resultado da multiplicação $y'*x$ será o mesmo. Existem dois outros produtos que são transpostos um do outro.

```
>> x*y'
```

```
Ans =
2 1 -1
0 0 0
-4 -2 2
```

```
>> y*x'
```

```
Ans =
2 0 -4
1 0 -2
-1 0 2
20
```

O produto de uma matriz por um vetor é um caso especial do produto entre matrizes. Por exemplo A e X,

```
>> b = A*x
```

```
B =  
5  
8  
-7
```

Naturalmente, um escalar pode multiplicar ou ser multiplicado por qualquer matriz.

```
>> pi*x  
ans =  
-3.1416  
0  
6.2832
```

Além da multiplicação matricial e escalar, podemos ter a multiplicação por elemento de matrizes de mesma dimensão. Esse tipo de operação é feita utilizando-se um ponto (.) antes do operador de multiplicação (*). Ou seja, se A e B são matrizes definidas por $A = [a_{11} a_{12} \dots a_{1n}; a_{21} a_{22} \dots a_{2n}; \dots; a_{m1} a_{m2} \dots a_{mn}]$ e $B = [b_{11} b_{12} \dots b_{1n}; b_{21} b_{22} \dots b_{2n}; \dots; b_{m1} b_{m2} \dots b_{mn}]$, então $A.*B = a_{ij}.*b_{ij}$. Por exemplo:

```
>> A.*B  
ans =  
1 8 21  
8 25 48  
21 48 0
```

Divisão

Existem dois símbolos para divisão de matrizes no Octave "\ " e "/". Se A é uma matriz quadrada não singular, então $A \setminus B$ e B/A correspondem respectivamente à multiplicação à esquerda e à direita da matriz B pela inversa da matriz A, ou $\text{inv}(A)*B$ e $B*\text{inv}(A)$, mas o resultado é obtido diretamente. Em geral,

- $X = A \setminus B$ é a solução de $A*X = B$
- $X = B/A$ é a solução de $X*A = B$

Por exemplo, como o vetor b foi definido como $A*x$, a declaração

```
>> z = A\b  
  
z =  
-1  
0  
2
```

A divisão por elemento entre matrizes é definida de maneira similar à multiplicação por elemento, ou seja, $A./B = a_{ij}/b_{ij}$ e $A.\setminus B = a_{ij} \setminus b_{ij}$, onde A e B têm mesma dimensão.

Exponenciação

A expressão A^p eleva A à p-ésima potência e é definida se A é matriz quadrada e p um escalar. Se p é um inteiro maior do que um, a exponenciação é computada como múltiplas multiplicações. Por exemplo,

```
>> A^3
Ans =
279 360 306
684 873 684
738 900 441
```

A exponenciação por elemento entre matrizes é definida de maneira similar à multiplicação por elemento, ou seja, $A.^B = a_{ij}^{b_{ij}}$, onde A e B têm mesma dimensão. De maneira similar, a potenciação por elemento entre uma matriz e um escalar apresenta as seguintes formas: $A.^c = a_{ij}^c$ e $c.^A = c^{a_{ij}}$

Elementos das Matrizes

Um elemento individual da matriz pode ser indicado incluindo os seus subscritos entre parênteses. Por exemplo, dada a matriz A:

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

a declaração

```
>> A(3,3) = A(1,3) + A(3,1)
```

```
A =
1 2 3
4 5 6
7 8 10
```

```
>> A(1:3,2)
ans =
2
5
8
```

```
>> A(1:3,2:3)
```

é uma submatriz 3x2, que consiste das três linhas e das últimas duas colunas de A.

```
ans =
2 3
5 6
8 10
22
```

Utilizando os dois pontos no lugar de um subscrito denota-se todos elementos da linha ou coluna. Por exemplo,

```
>> A(1:2,:)
ans =
1 2 3
4 5 6
```

é uma submatriz 2x3 que consiste da primeira e segunda linhas e todas colunas da matriz A.

- Funções: o Octave possui algumas funções que se aplicam a matrizes como, por exemplo, as funções `size` (fornece o número de linhas e colunas de uma matriz) e `length` (fornece o maior valor entre o número de linhas e colunas). O Octave tem também funções que se aplicam individualmente a cada coluna da matriz produzindo um vetor linha com os elementos correspondentes ao resultado de cada coluna. Se a função for aplicada à transposta da matriz, os resultados serão relativos a cada

linha da matriz. Se o argumento da função for um vetor, o resultado será um escalar. Algumas dessas funções são mostradas na Tabela 12.

Função	Descrição
sum	Soma dos elementos
prod	Produto dos elementos
mean	Média aritmética
std	Desvio padrão
max	Maior elemento
min	Menor elemento
sort	Ordena em ordem crescente

Tabela 12 - Algumas funções de matrizes

Submatrizes

Sendo B uma matriz 5x5 unitária, podemos defini-la através da seguinte função:

```
>> B = ones (5)
B =
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

Sendo C uma matriz de zeros 3x4, podemos defini-la como:

```
>> C=zeros (3,4)
C =
0 0 0 0
0 0 0 0
0 0 0 0
```

Para que o Octave gere uma matriz de números aleatórios entre 0 e 1, utilizamos a função rand(veja também a função *randn*, utilizando o comando *help*). Exemplo:

```
>> D=rand(2,3)
D =
0.2190 0.6789 0.9347
0.0470 0.6793 0.3835
```

Gráficos no Octave

Vimos então que o Octave fornece uma ferramenta poderosa para trabalhar com vetores e matrizes. Mas existe outra funcionalidade do Octave que é tão importante quanto as operações vistas até aqui que são as operações para trabalhar com gráficos. Esta tarefa de construir gráficos se torna muito simples fácil quando se está utilizando esta ferramenta matemática. Com o Octave é possível construir uma grande quantidade de gráficos de diferentes formatos como gráficos de barra, diagramas de pareto, além de gráficos em duas e três dimensões em qualquer tipo de escala. Por simplicidade veremos aqui apenas as funções básicas de impressão de gráficos que são as funções plot, subplot e fplot.

Plot

Função utilizada para plotar gráficos bidimensionais utilizando uma escala dos eixos linear. Se Y é um vetor, plot(Y) produz um gráfico linear dos elementos de Y versos o índice dos elementos de Y. Por exemplo, para plotar os números [0.0, 0.48, 0.84, 1.0, 0.91, 0.6, 0, 14], entre com o vetor e execute o comando plot:

```
>> Y = [0.0, 0.48, 0.84, 1.0, 0.91, 0.6, 0, 14];
>> plot(Y)
```

O resultado dos comandos anteriores pode ser visto no gráfico mostrado na Figura 6.

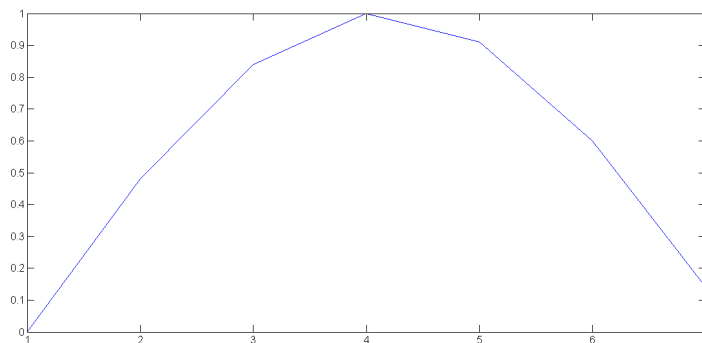


Figura 6 - Exemplo de gráfico bidimensional linear com uma variável e índice

Se X e Y são vetores com dimensões iguais, o comando `plot(X,Y)` produz um gráfico bidimensional dos elementos de X versos os elementos de Y, por exemplo

```
>> t = 0:0.05:4*pi;  
>> y = sin(t);  
>> plot(t,y)
```

A Figura 7 mostra o gráfico produzido pelos comandos acima.

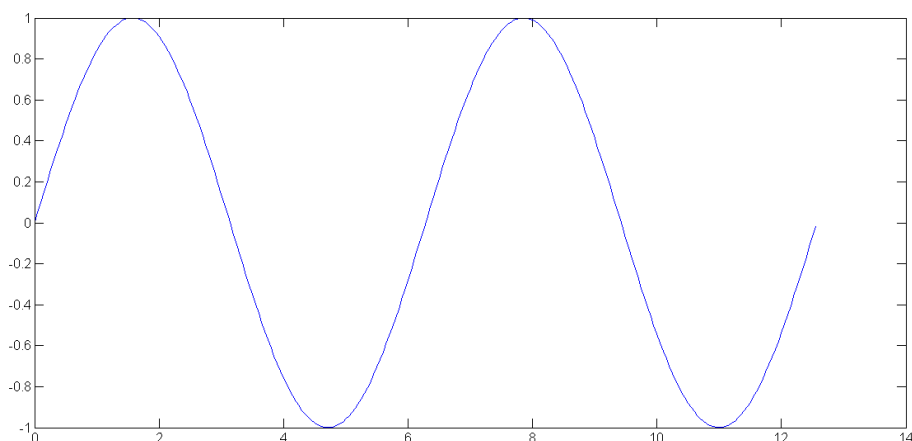


Figura 7 - Gráfico do seno de t

O Octave pode também plotar múltiplas linhas e apenas um gráfico. Existem duas maneiras, a primeira é usado apenas dois argumentos, como em `plot(X,Y)`, onde X e/ou Y são matrizes. Então:

- Se Y é uma matriz e X um vetor, `plot(X,Y)` plota sucessivamente as linhas ou colunas de Y versos o vetor X.
- Se X é uma matriz e Y é um vetor, `plot(X,Y)` plota sucessivamente as linhas ou colunas de X versos o vetor Y.
- Se X e Y são matrizes com mesma dimensão, `plot(X,Y)` plota sucessivamente as colunas de X versos as colunas de Y.
- Se Y é uma matriz, `plot(Y)` plota sucessivamente as colunas de Y versos o índice de cada elemento da linha de Y.

A segunda, e mais fácil, maneira de plotar gráficos com múltiplas linhas é usando o comando `plot` com múltiplos argumentos. Por exemplo:

```
>> plot(t, sin(t), t, cos(t), t, sin(t + pi), t, cos(t + pi))
```


A Figura 8 representa um gráfico onde foram impressos várias funções diferentes.

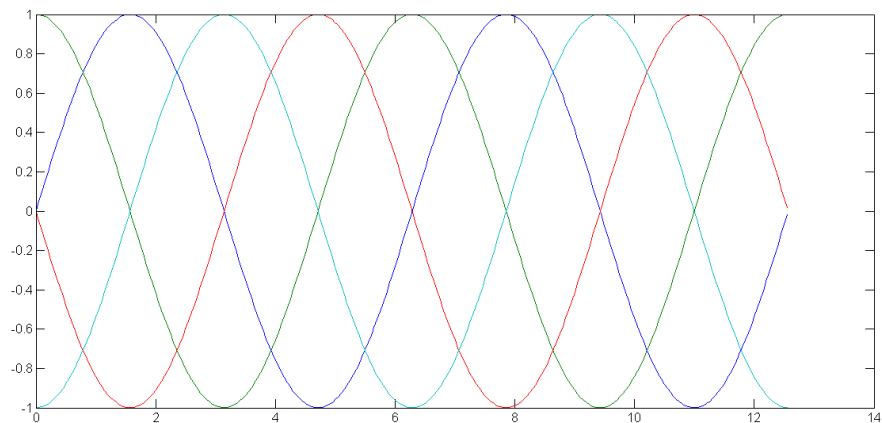


Figura 8 - Múltiplos gráficos em apenas um eixo

Existe ainda uma outra forma de imprimir vários gráficos em uma mesma figura mas em eixos diferentes que é utilizando o comando subplot este comando possui a seguinte sintaxe, subplot(m,n,p) divide a figura em uma matriz m x n e imprime o gráfico na posição p da matriz, exemplo:

```
>> subplot(1,2,1), plot(t,sin(t))
```

Que imprime o gráfico em um eixo independente à esquerda da Figura 9.

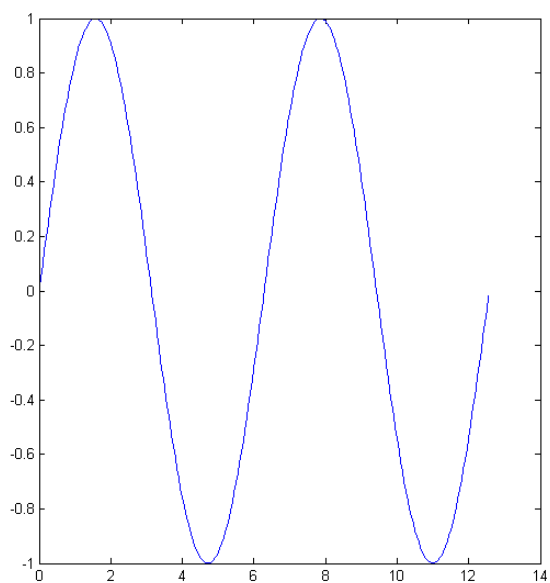


Figura 9 - Gráfico do seno impresso na posição à esquerda da figura

Executando o commando

```
>> subplot(1,2,2), plot(t,cos(t))
```

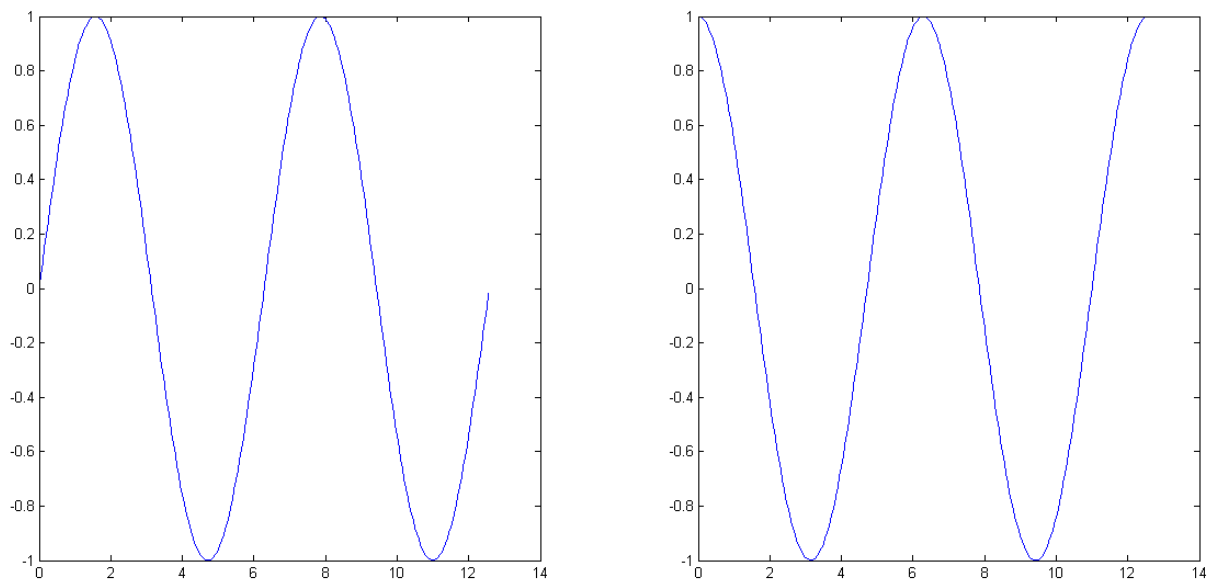


Figura 10 – Gráfico do seno impresso na posição esquerda da figura e do cosseno impresso na posição direita.

Que imprime o gráfico em um eixo independente à direita da Figura 10.

Os tipos de linhas, símbolos e cores usados para plotar gráficos podem ser controlados se os padrões não são satisfatórios. Por exemplo,

```
>> X = 0:0.05:1;
>> subplot(121), plot(X,X.^2,'k*')
>> subplot(122), plot(X,X.^2,'k --')
```

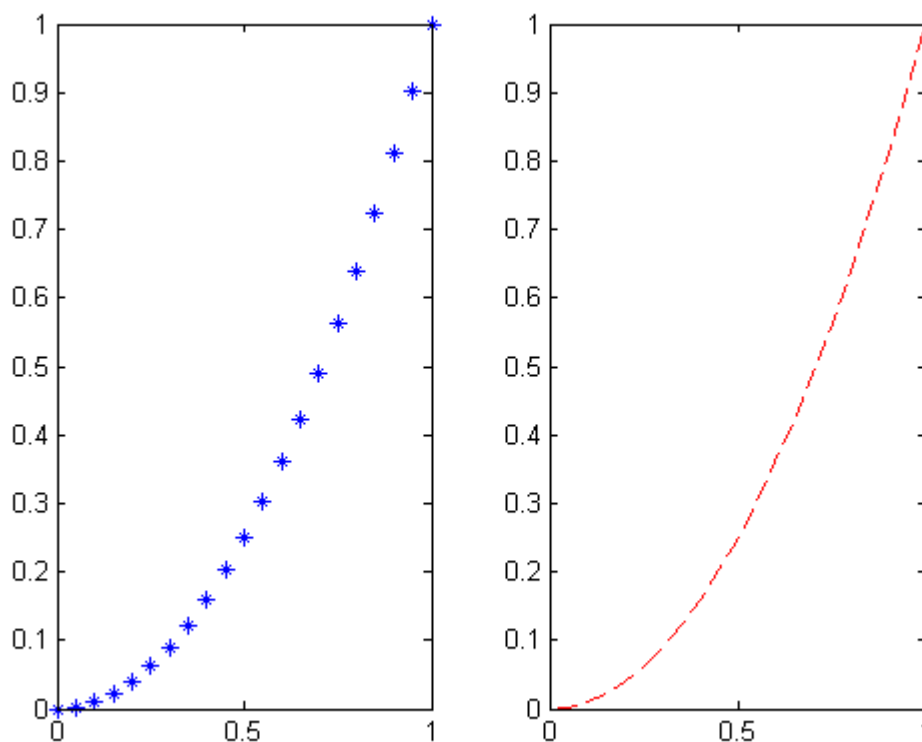


Figura 11 - Gráficos de tipos diferentes em eixos diferentes.

A Figura 11 mostra o gráfico impresso em dois eixos diferentes com os tipos de linhas também diferentes. As tabelas Tabela 13, Tabela 14 e Tabela 15.

Tipo de linha	
—	_____
--	-----
-.	-.-.-.-.-
.

Tabela 13 - Tipos de linhas dos gráficos

Tipos de marcadores de pontos	
.
*	*****
o	oooooooooooo
+	+++++
x	xxxxxxxxxxxx

Tabela 14 - Marcadores de pontos nos gráficos.

Cores	
y	Amarelo
m	Lilás
c	Azul claro
r	Vermelho
g	Verde
b	Azul
w	Branco
k	Preto

Tabela 15 – Cores das linhas e dos marcadores nos gráficos.

As funções title, xlabel e ylabel são utilizadas para alterar o título, o rótulo de x e o rótulo de y do gráfico respectivamente. Exemplo

```
>> plot(sin(linspace(-pi,pi,10)));
>> title('Gráfico da função f(x)=seno(x), -pi<x<pi')
>> xlabel('x')
>> ylabel('f(x)')
>> grid
```

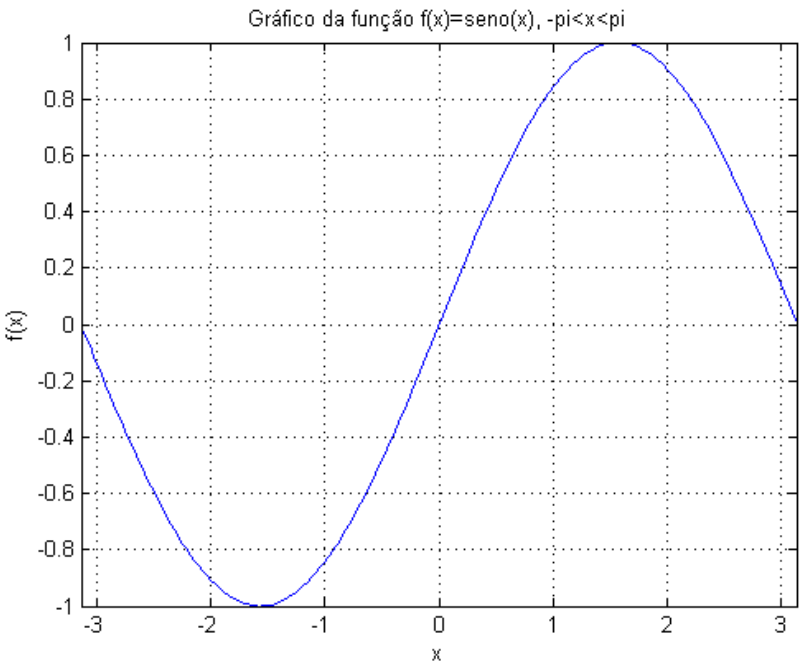


Figura 12 - Gráfico do seno com rótulos para os eixos e nome para gráfico.

ATIVIDADES

1.1 Faça as operações utilizando o prompt de comando do Octave e complete a tabela

Operação	Resposta
12.6987+ 91.3376	
63.2359 X 9.7540	
27.84 ^{54.68}	
e ^{0.15}	
log ₂ (970)	
a = 957+ 485.37	
b = (957+ 485.37) X (35.8 + 7.8)	
c = [(957+ 485.37) X (35.8 + 7.8)] ^{-41.57}	
d = {[(957+ 485.37) X (35.8 + 7.8)] ^{-41.57} }x-29.2207	
cos(90°)	
sen(45°)	
√15.57	

1.2 Faça as operações com vetores e matrizes utilizando o prompt de comando do Octave e complete a tabela.

Termos		Operação	Resultado
A=[10.7773 -15.5478]	B = [-25.7740 -24.3132]	A+B	
		A-B	
		Multiplica de B por A elemento a elemento	
		Divisão de B por A elemento a elemento	
$A = \begin{bmatrix} 32.8 & 22.3 & -32.3 \\ -20.6 & 45.3 & -19.4 \\ 46.8 & 40.2 & 18.2 \end{bmatrix}$	$B = \begin{bmatrix} -45.0 & 11.8 & 31.3 \\ 46.5 & -26.5 & 1.0 \\ 6.1 & -29.5 & 5.4 \end{bmatrix}$	Divida o elemento A(2,1) pelo elemento A(1,1)	
		Divida o elemento A(3,1) pelo elemento A(1,1)	
		Multiplique a divisão do elemento A(2,1) e elemento A(1,1) por todos os elementos da linha 1 da matriz A e some com todos os elementos da segunda linha.	
		Multiplique a divisão do elemento B(3,2) e elemento B(1,1) por todos os elementos da linha 1 da matriz A e some com todos os elementos da segunda linha de B.	
		A*B ^T	
		Crie o vetor com os valores [100.0 100.1 100.2 ... 1000]	

1.3 Resolva o problema abaixo utilizando o Octave

Você trabalha em uma empresa que monta peças para trator. No motor deste tratores são utilizados 3 tipos de aços carbono diferentes SAE 1020, 8620 e 5115. São produzidas peças para 3 tratores diferentes retroescavadeira, escavadeira e trator de esteira. Estes tratores utilizam os tipos de aço conforme tabela abaixo

	1020	8620	5115	x1000
Retroescavadeira	0,5	1	0,1	
Escavadeira	3	5	0,3	
Esteira	4	7	0,8	

- A) Se sua empresa produz 20 retroescavadeira, 30 escavadeira e 5 tratores de esteira por mês. Qual a quantidade de cada tipo de aço deverá ser comprada no mês?
- B) Sabendo que o preço por unidade dos aços, está de acordo com a tabela abaixo quanto será gasto por mês em material?

Aço	Preço
1020	5,00
8620	20,00
5115	15,00

- C) Quanto será gasto para construir cada tipo de trator?

1.4 Faça os gráficos que se pede abaixo..

- a. $y = [1 : 10]$
- b. $x = [-31.4724 \ -40.5792 \ 37.3013 \ -41.3376]$ e $y = x^2 + 4$
- c. \cos no intervalo de -10 até 10
- d. $x = [-45.7507 \ -46.4889 \ 34.2387 \ -47.0593]$ e $y = e^{x-40}$

1.5 Usando o Octave utilize a quantidade de algarismos significativos indicado e diga qual foi a ordem do erro de arredondamento.

- a. $\frac{\sqrt{5}+3}{0,3541}$, 5 algarismos Resposta: _____
- b. $\frac{e^3 + \ln(5)}{\sin(3) + \tan(0,5)}$ 4 algarismos Resposta: _____
- c. $\log_3 5$, 3 algarismos Resposta: _____
- d. $\sqrt[3]{3,16}$, 5 algarismos Resposta: _____

1.6 Utilize a série de Maclaurin para calcular o valor do $\sin(x)$ com o x e a quantidade de termos pedida.

Para determinar a série de Maclaurin deve-se seguir os seguintes passos:

1º → Calculam-se as derivadas sucessivas para a função no caso $\sin(x)$ no ponto $x = 0$, então teremos:

$$f(x) = \sin(x) \Rightarrow f(0) = 0$$

$$f'(x) = \cos(x) \Rightarrow f'(0) = 1$$

$$f''(x) = -\sin(x) \Rightarrow f''(0) = 0$$

$$f'''(x) = -\cos(x) \Rightarrow f'''(0) = -1 \dots$$

2º → Substituem-se os valores na fórmula de Maclaurin e então teremos:

$$f(x) = f(0) \frac{x}{1!} + f'(0) \frac{x^2}{2!} + f''(0) \frac{x^3}{3!} + \dots + \frac{x^n}{n!} f^{(n)}(0)$$

- a. Calcule o $\sin(2)$ usando os 6 primeiros termos da série da fórmula obtida deixando o resultado com todas as casas decimais.

Resposta: _____

- b. Calcule direto o $\sin(2)$.

Resposta: _____

- c. Compare os resultados obtidos. Qual é a ordem dos erros obtidos?

1.7 A função $y = \sqrt[3]{x}$ pode ser aproximada pela fórmula:

$$f(x) = \frac{2}{3} + \frac{x}{3} - \frac{1}{9}(x-1)^2 + \frac{5}{81}(x-1)^3 - \frac{10}{243}(x-1)^4 + \frac{22}{729}(x-1)^5$$

A fórmula foi obtida do polinômio de Taylor cuja forma geral é:

$$f(x) = f(a) + \frac{(x-a)^1}{1!} f'(a) + \frac{(x-a)^2}{2!} f''(a) + \dots + \frac{(x-a)^n}{n!} f^{(n)}(a)$$

Para obter a fórmula foi considerado $a=1$, calculadas as derivadas sucessivas no ponto 1, substituídas no polinômio de Taylor e foram feitas algumas simplificações.

Para verificar a presença do erro de truncamento preencha a tabela anotando os valores com 6 algarismos significativos

x	$y = \sqrt[3]{x}$	$f(x) = \frac{2}{3} + \frac{x}{3} + \dots$	$y - f(x)$
0,7			
0,8			
0,9			
1,0			
1,1			

Preencha novamente a tabela:

x	$y = \sqrt[3]{x}$	$f(x) = \frac{2}{3} + \frac{x}{3} + \dots$	$y - f(x)$
10,3			
10,8			
11,3			

A aproximação melhorou ou piorou?

AULA PRÁTICA 2 - SCRIPTS E FUNÇÕES NO OCTAVE

OBJETIVO

- ✓ Aprender construir programas utilizando arquivos no Octave

INTRODUÇÃO

Para resolver problemas simples, é cômodo e eficiente utilizar o Octave como se fosse uma calculadora, entrando-se com os comandos diretamente no *prompt*. Ou seja, cada linha de comando é introduzida na Janela de Comandos e processada imediatamente. Entretanto, à medida que o número de comandos aumenta, ou quando se deseja mudar o valor de uma ou mais variáveis e executar novamente os comandos, o melhor é utilizar o Octave como uma linguagem de programação, ou seja, utilizar o Octave para executar sequências de comandos armazenadas em arquivos de roteiro (script). Esses arquivos que contêm as declarações do Octave são chamados arquivos ".m" (ou M-files), como, por exemplo, exemplo1.m. Esses M-files são os programas fontes do Octave e consistem de sequências de comandos normais do Octave, possibilitando incluir outros arquivos ".m" escritos no formato texto (ASCII).

Para escrever um programa (ou arquivo .m) no Octave, escolha File na barra de menu. Dentro do menu File escolha *New* e selecione M-file. Abre-se, então, um editor de textos, onde pode-se escrever os comandos do Octave. Para editar um arquivo já existente, selecione a opção Open M-File, a partir do menu File. Os arquivos podem, também, ser editados fora do Octave utilizando qualquer editor de texto. Escreva, por exemplo o Programa 1:

```
%=====
```

%Programa 1 - Exemplo de script do Octave

```
% Este exemplo plota uma função seno nas seguintes  
% condições:  
% sen(x)  
% 2*sen(x)  
% 2*sen(x+45)  
% 2*sen(x-90)  
% 2*sen(2*x)  
%=====
```

```
x=0:360;
```

```
% Seno com amplitude A=1 e defasagem phi=0 graus
```

```
A=1;  
phi=0;  
y=A*sin(2*pi*x/360+2*pi*phi/360);
```

```
% Seno com amplitude A=2 e defasagem phi=0 graus
```

```
A=2;  
z=A*sin(2*pi*x/360+2*pi*phi/360);
```

```
% Seno com amplitude A=2 e defasagem phi=45 graus
```

```
phi=45;  
v=A*sin(2*pi*x/360+2*pi*phi/360);
```

```
% Seno com amplitude A= 2 e defasagem phi=-90 graus
```

```
phi=-90;  
32  
w=A*sin(2*pi*x/360+2*pi*phi/360);
```

```
% Seno com amplitude A= 2 e defasagem phi=0 graus
phi=0;
u=A*sin(2*pi*2*x/360+2*pi*phi/360);

% Plotagem do resultado
plot(x,y,'k-',x,z,'k--',x,v,'k-.',x,w,'k.',x,u,'ko')
grid
xlabel('Valores de x em graus')
ylabel('y,z,v,w e u')
title('Estudo de defasagem e amplitude de um seno')
legend('sen(x)', '2*sen(x)', '2*sen(x+45)', '2*sen(x-90)', '2*sen(2*x)')
```

Uma vez escrito o programa, entre no menu File da janela do editor de textos e escolha a opção *Save as...* Nesta opção do menu, salve o programa como prog1.m no seu diretório de trabalho. Em seguida, volte à janela de comandos do Octave e use o comando *cd* ou a opção *Set Path...* do menu File para ir ao diretório onde o programa prog1.m foi salvo. Em seguida, digite o nome do arquivo (sem a extensão .m) para executar o programa:

```
>>prog1
```

O gráfico obtido é mostrado na Figura 13.

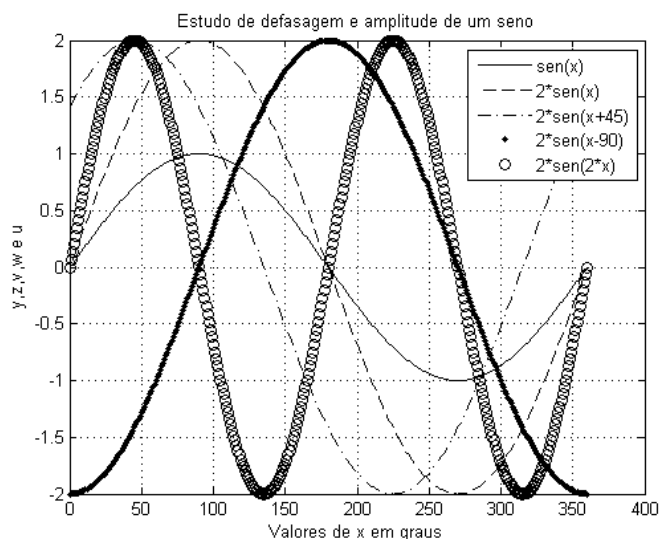


Figura 13 - Gráfico exemplo de script.

Por causa da grande utilidade dos M-files, o Octave possui diversas funções que tornam os M-files ainda mais interessantes. Estas funções estão listadas na Tabela 16.

Comando	Descrição
echo on	É usado para que os comandos do M-file sejam mostrados na janela de comandos durante a execução.
echo off	É usado para suprimir a exibição dos comandos feita através do echo on.
input	Permite entrada de dados durante a execução do programa via teclado.
pause	Faz uma pausa na execução do programa até que uma tecla qualquer seja pressionada.
pause(n)	Faz uma pausa de n segundos na execução do programa.
disp(ans)	Visualiza os resultados sem mostrar os nomes das variáveis.
keyboard	Passa o controle temporariamente para o teclado ("Type return to quit")

Tabela 16 - Lista de comandos para arquivos de script Octave.

Como exemplo, considere os seguintes programas (M-file) :

```
%=====
```

Programa 2 – Exemplo 1 de utilização da função input.

```
%=====
```

```
%Plota uma função  $y=ax^2 + bx + c$  no intervalo  $-5 < x < 5$ 
clear
aux='s';
while aux== 's',
a=input('a =');
b=input('b =');
c=input('c =');
x=-5:0.1:5;
y=a*x.^2+b*x+c;
plot(y)
figure(1)
pause
clc
close
aux=input('Plotar outro ? (s/n) ==> ','s');
end
```

Repare, além do uso do comando *input*, o uso do carácter % (comentário) no texto, do comando *clear* (apaga todos os dados da memória), *pause* (provoca uma pausa na execução do arquivo até que qualquer tecla seja digitada), *clc* (limpa a Janela de Comando), *figure(1)* (mostra a Janela Gráfica número 1) e *close* (fecha todas as Janelas Gráficas).

Programa 3 – Exemplo 2 de utilização da função M-file “input”

```
%=====
```

```
% Programa para traçar a curva :
%
%  $y=A.\sin(x+\phi)$ ,
%
% sendo que os valores de  $x$ [rad],  $A$  e  $\phi$ [graus] devem ser
% entrados via teclado durante a execução do programa
%
x=input('Entre com o vetor x [rad]> ');
A=input('Entre com o valor de A> ');
phi=input('Entre com o valor de phi [graus]> ');

y=A*sin(x+pi*phi/180);
plot(x,y,'r');
grid on
title('Exemplo de utilização da função "input"')
xlabel('x em rad/s')
ylabel('y=A.sin(x+phi)')
```

Programa 4 – Exemplo 3 de utilização da função M-file “input”

```
% Programa decsomat.m
```

```
%=====
```

```
% Programa para gerar uma matriz com elementos aleatórios
% entre -10 e 10 e decompô-la na soma de três matrizes :
% uma triangular inferior, uma diagonal e outra triangular
% superior
%-----
n = input('Ordem da matriz : ');
A = fix(20*(rand(n) -0.5 * ones(n)));
```

```

D = diag(diag(A));
L = tril(A) - D;
U = triu(A) - D;
A, L, D, U
% Fim do programa
%----->> decsomat

```

Ordem da matriz : 3

```

A =
-5 3 0
-9 8 6
3 -2 -9
L =
0 0 0
-9 0 0
3 -2 0
D =
-5 0 0
0 8 0
0 0 -9
U =
0 3 0
0 0 6
0 0 0

```

Controles de Fluxo

Estruturas Condicionais

Uma estrutura condicional permite a escolha do grupo de comandos a serem executados quando uma dada condição for satisfeita ou não, possibilitando desta forma alterar o fluxo natural de comandos. Esta condição é representada por uma expressão lógica.

Estrutura if-end

A estrutura condicional mais simples do Octave é:

```

if <condição>
    <comandos>
end

```

Se o resultado da expressão lógica <condição> for 1 (verdadeiro) então a lista de <comandos> será executada. Se o resultado for 0 (falso) os <comandos > não serão executados. Por exemplo, considere o arquivo estcond1.m cujo conteúdo é:

```

a = input('Entre com o valor de a : ');
if a >= 0
    b = sqrt(a)
end

```

Para executá-lo basta fornecer o seu nome na área de trabalho

```

>> estcond1
Entre com o valor de a : 2
b = 1.4142

```

Neste exemplo, a raiz quadrada de a será atribuída a b somente se o valor de a for maior ou igual a 0. Considere o arquivo banana.m:

```
custo=5;
bananas=10;
if bananas>5
custo=0.1*custo;
end
custo
>>banana
custo =
0.5000
```

No exemplo acima, a expressão bananas > 5 é verdadeira, assim o comando :

```
custo=0.1* custo
Exemplo 2 :
custo=5;
bananas=5;
if bananas>5
custo=0.1*custo;
end
custo
>>banana
custo =
5
```

Neste exemplo, a expressão bananas > 5 é falsa, assim o comando :

```
custo=0.1* custo
```

não foi executado. Assim o custo continua igual a 5.

Estrutura if-else-end

No caso de haver duas alternativas, uma outra estrutura condicional deve ser usada:

```
if <condição>
    <comandos 1>
else
    <comandos 0>
end
```

Se o resultado da expressão lógica <condição > for 1 (verdadeiro) então a lista <comandos 1> será executada. Se <condição> for 0 (falso) então será a lista <comandos 0> a ser executada. Por exemplo, o programa do arquivo estcond2.m

```
a = input('Entre com o valor de a : ');
if a > 0
b = log(a)
else
b = exp(a)
```

quando executado resultará

```
>> estcond2
```

```
Entre com o valor de a : 5
b = 1.6094
```

Se a for positivo, então o logaritmo natural de a será atribuído a b e se a for negativo ou nulo, então b será igual ao exponencial de a .

Exemplo: Plote uma função retangular utilizando-se a estrutura if-else-end.

```
%
x=linspace(0,2*pi,100); % Criou-se 100 amostras entre 0 e 2*pi
%
for n=1:100
    if x(n)<=pi
        f(n)=1; %Faz f(t)=1 para 0<t<=pi,i.e.,
                %as primeiras 50 amostras de
                %f(t) são iguais a 1
    else
        f(n)= -1; % Faz f(t)=-1 para pi<t<=2*pi,
        % i.e., as últimas 50 amostras de
        % f(t) são iguais a 1
    End
end
plot(x,f, 'ko'); grid on
title('Função retangular')
xlabel('t em radianos')
ylabel('f(t)')
```

Estrutura if-elseif-end

Quando houver mais de duas alternativas, a estrutura if-else-end do Octave torna-se

```
if <condição 1>
    <comandos 1>
elseif <condição 2>
    <comandos 2>
elseif <condição 3>
    <comandos 3>
    .
    .
else
    <comandos 0>
end
```

A lista <comandos 1> será executada se <condição 1> for igual a 1 (verdadeiro), já a lista <comandos 2> será executada se <condição 2> for 1 e assim para as outras condições. Se nenhuma das condições for 1 então <comandos 0> será executada. Quando a primeira <condição> for satisfeita e os <comandos> executados, a estrutura if-elseif-end será abandonada, ou seja, o controle do processamento será transferido para o comando imediatamente após o *end*. Por exemplo, `estcond3.m`

```
a = input('Entre com o valor de a : ');
if a <= -10
    b = exp(a)
elseif a < 0
    b = 1/a
elseif a <= 10
    b = a^2
elseif a < 100
    b = sqrt(a)
else
    b = 10
end
```

quando executado resultará

```
>> estcond3
```

```
Entre com o valor de a : 4  
b = 16
```

Deste modo foi executado o primeiro comando para o qual a condição $a \leq 10$ foi satisfeita, ou seja, apesar da condição $a < 100$ ser também verdadeiro, o comando referente a ela não foi executado. Assim, na estrutura if-elseif-end é executada somente uma única lista de comandos.

Estruturas de repetição

A estrutura de repetição faz com que uma sequência de comandos seja executada repetidamente até que uma dada condição de interrupção seja satisfeita. O Octave possui duas estruturas de repetição: as estruturas for-end e a while-end

Estrutura for-end

A estrutura for-end permite que um grupo de comandos seja repetido um número específico de vezes. Sua sintaxe é:

```
for <variável>=<arranjo>  
    <comandos>  
end
```

onde <variável> é a variável-de-controle que assume todos os valores contidos no vetor linha <arranjo> . Assim, o número de repetições da lista <comandos > é igual ao número de elementos no vetor <arranjo>. A variável-de-controle não pode ser redefinida dentro da estrutura for-end. O laço for é o controlador de fluxo mais simples e usado na programação Octave.

Analisando a expressão:

```
for i=1:5  
X(i)=i^2  
end
```

pode-se notar que o laço for é dividido em três partes:

- A primeira parte ($i=1$) é realizada uma vez, antes do laço ser inicializado.
- A segunda parte é o teste ou condição que controla o laço, ($i \leq 5$).
- Esta condição é avaliada; se verdadeira, o corpo do laço ($X(i)=i^2$) é executado.

A terceira parte acontece quando a condição se torna falsa e o laço termina. O comando end é usado como limite inferior do corpo do laço. Vamos considerar um exemplo, executando o programa estrep1.m abaixo:

```
n = input('Valor de n : ');  
s = 0;  
n2 = n^2;  
40  
for i = 1:2:2*n-1  
    s = s + i;  
end,  
n2, s
```

```
>> estrep1  
Valor de n : 5  
n2 = 25  
s = 25
```

Este exemplo mostra que a soma dos n primeiros números ímpares é igual ao quadrado de n , pois para $n=5$ a variável-de-controle i assume os valores 1 3 5 7 9. Deve ser observado o uso do (;) para suprimir a exibição de resultados intermediários no cálculo de s .

Estrutura while-end

A estrutura while-end, ao contrário da for-end, repete um grupo de comandos um número indefinido de vezes. Sua sintaxe é

```
while <condição>
    <comandos>
end
```

Enquanto a expressão lógica <condição> for verdadeira a lista <comandos> será repetida. No laço while apenas a condição é testada. Por exemplo, na expressão

```
a = 1; b = 15;
while a<b,
    clc
    a = a+1
    b = b-1
    pause(1)
end
disp('fim do loop')
```

a condição $a < b$ é testada. Se ela for verdadeira o corpo do laço, será executado. Então a condição é testada novamente, e se verdadeira o corpo será executado novamente. Quando o teste se tornar falso o laço terminará, e a execução continuará no comando que segue o laço após o end. Ao contrário do loop for, que executa um conjunto de comandos um número fixo de vezes, o loop while executa um conjunto de comandos um número indefinido de vezes. Os comandos entre as instruções while e end são executadas enquanto todos os elementos na expressão forem verdadeiras.

Exercício: Construa o vetor $y = [64 \ 32 \ 16 \ 4 \ 2 \ 1]$, usando o loop while

```
Solução:
num=128;
n=0;
while num>1
    num=num/2;
    n=n+1;
    y(n)=num;
end
y
```

Por exemplo, em precisao.m

```
n = 0;
Epsilon= 1;
while 1 + Epsilon > 1
    n = n + 1;
    Epsilon = Epsilon / 2;
end
n, Epsilon, eps
>> precisao
n = 53
Epsilon = 1.1102e-16
eps = 2.2204e-16
```

Epsilon é a chamada precisão da máquina, ou seja, o maior número que somado a 1 é igual a 1. Comparada com a variável especial eps do Octave

```
>> 1+eps-1
ans = 2.2204e-16
>> 1+Epsilon-1
ans = 0
```

Note que quando eps é somado a 1 resulta em um número maior do que 1. O mesmo não ocorre com Epsilon, porque qualquer valor igual ou menor do que ele somado a 1 será simplesmente 1.

Comando break(estruturas com interrupção no interior)

A estrutura while-end permite que um grupo de comandos seja repetido um número indeterminado de vezes. No entanto, a condição de interrupção é testada no início da estrutura. Em várias situações em programação se faz necessário interromper a execução da repetição verificando a condição no interior da estrutura e não no seu início. O comando break interrompe a execução de uma estrutura while-end ou for-end e transfere a execução para o comando imediatamente seguinte ao end. Em repetições aninhadas, o break interrompe a execução apenas da estrutura mais interna. Uma repetição com condição de interrupção no interior pode ter a forma

```
While 1
    <comandos 1>
    if <condição>
        break
    end
    <comandos 2>
end
```

A estrutura while-end é executada indefinidamente a princípio pois a condição do while é sempre verdadeira. Contudo, quando a <condição> do if for satisfeita o comando break será executado causando a interrupção da repetição while-end. Por exemplo, o programa no arquivo estrep3.m

```
while 1
    a=input('Entre com a, a>0 : ');
    if a <= 0
        break
    end
    disp(rats(a))
end
```

lista continuamente a representação racional de um número fornecido enquanto este for positivo. Deste modo,

```
>> estrep3
```

```
Entre com a, a>0 : pi
355/113
Entre com a, a>0 : sqrt(2)
1393/985
Entre com a, a>0 : -8
```

Considere mais um programa para exemplificar o uso do comando break:

```
%Programa para criar e modificar uma matriz A
for i = 1:5,
    for j = 1:5,
        if i == j
            A(i,j) = 2;
```

```

        elseif abs(i-j) == 1
            A(i,j) = -1;
        else
            A(i,j) = 0;
        end
    end
end
end
clc
x = 's';
for i = 1:5,
    if x == 'q',
        break
    end
    j = 1;
    while j<=5,['A('num2str(i) ', ' num2str(j)') = 'num2str(A(i,j))']
        x = input('Modifica? (s-sim, n-não, p-próxima linha, q-sair) =>');
        if x == 's',
            A(i,j) = input('Entre com o novo valor de A(i,j) ');
            j=j+1;
            clc
        end
        if x == 'n',
            j=j+1;
            clc
        end
        if x == 'p',
            clc
            break
        end
        if x == 'q',
            clc
            break
        end
    end
end
end
end

```

Subprograma function

Outro tipo de arquivo de roteiro é usado para o próprio usuário criar novas funções para o Octave. Na realidade, várias funções do Octave são arquivos .m. Uma função é criada no Octave como um arquivo .m, porém começando sempre com o seguinte cabeçalho:

```
function [variáveis de saída] = Nome_da_Função (variáveis de entrada)
```

Todas as variáveis temporárias usadas na função são variáveis locais e, com isso, após a execução da função, elas são removidas do espaço de trabalho. Como exemplo, veja como o Octave implementa a função trace:

```

%TRACE Sum of diagonal elements.
% TRACE(A) is the sum of the diagonal elements of A, which is
% also the sum of the eigenvalues of A.
% Copyright (c) 1984-98 by The MathWorks, Inc.
function t = trace(a)
t = sum(diag(a));

```

As linhas de comentário (prefixadas por %) de uma função, quando introduzidas imediatamente após o cabeçalho da função, definem o help on-lineda própria função.

Veja agora um exemplo de uma função escrita pelo usuário e como ela é utilizada por um programa do Octave:


```
function azr=rmz(a)
%al=rmz(a) removes the leading zero elements of a vector
%until a possible scalar variable remains
azr=a;
while (azr(1)==0)&(length(azr)>1),
    azr(1)=[];
end

%Programa que utiliza uma função criada pelo usuário
a=[0 0 0 1 0 3 6 0];
al=rmz(a);
a
al
```

Considere também a seguinte exemplo:

- Abra um arquivo, salvando-o com nome de prog_funcao.m
- Digite os seguintes comandos neste arquivo

```
% prog_funcao.m
% CRIANDO UMA SUBROTINA
v = 1:1:10;
m = media(v);
s = sprintf('\n A média é: %4.2f', m);
disp(s);
% final do programa prog_funcao.m
```

- Agora crie o seguinte arquivo, com o nome de media.m

```
function x = media(u)
% function x = media(u) calcula a média do vetor u, colocando o resultado
em x
x = sum(u)/length(u);
% final da subrotina media.m
```

- Na linha de comando do Octave, digite:

```
>> prog_funcao
>> echo on
>> prog_funcao
>> echo off
```

ATIVIDADES

- 2.1 Para cada função abaixo faça um *script* que mostre o gráfico da função para o intervalo e o número de pontos desejados
- a. $Y=2x^2+1$, $[-8\ 20]$, 100 pontos
 - b. $Y=\cos^2(x)+\sin^2(x)$, $[-12\ 14]$, 50 pontos
 - c. $Y=e^{x^3+5}$, $[-20\ -2]$, 10 pontos
 - d. $Y=\log_3(x^5+2)$, $[10\ 11]$, 200 pontos
 - e. $Y=\cos(3x^3+\pi)+x^3$, $[1\ 13]$, 50 pontos
- 2.2 Faça um *script* que leia a partir do teclado dois números e imprima uma mensagem com o maior e o texto “O maior número é: “.
- 2.3 Faça uma função que receba do usuário um número inteiro positivo e gere a sequência de Fibonacci com esta quantidade de números e retorne um vetor com a sequência.
- 2.4 Faça uma função onde o usuário entrará pelo teclado, três números diferentes (você deverá verificar se os números são mesmo diferentes) você deverá organizá-los em ordem crescente. Então solicitar que o usuário digite mais um número qualquer, diferente dos outros três digitados (você deverá verificar se são mesmo diferentes). Retorne um vetor com os quatro números em ordem decrescente.

AULA PRÁTICA 3 - Decomposição LU, Cholesky e LDL^T

OBJETIVO

- ✓ Utilizar métodos exatos para solução de sistemas lineares.

INTRODUÇÃO

No estudo da engenharia existe uma infinidade de problemas que dependem da solução de sistemas lineares. Daí a importância de se estudar tal ferramenta em todo e qualquer curso desta modalidade. Atualmente com o advento dos computadores surge uma necessidade ainda maior de ferramentas computacionais para determinar a solução dos problemas, principalmente na área de engenharia. É fundamental também que estas ferramentas sejam eficientes do ponto de vista computacional, ou seja, o tempo utilizado para resolver os problemas sejam cada vez menores. O que veremos nesta aula prática então são ferramentas para solução de sistemas lineares como o descrito na Equação 1 que fornece uma solução exata. O que consideramos por solução exata são soluções onde o erro é igual à zero ou é menor do que a precisão do computador.

$$Ax=b$$

Equação 1 - Sistema linear

Estudaremos nesta aula prática quatro soluções diferentes para o problema de se resolver um sistema linear, método de Gauss, decomposição LU, decomposição de Cholesky e decomposição LDL^T.

Método de Gauss

No método de Gauss são executadas operações l-elementares nas linhas de tal forma a se reduzir a matriz dos coeficientes a uma matriz triangular superior ou inferior e a partir daí utilizar uma metodologia de substituições sucessivas ou retroativas para encontrar os valores das variáveis dependentes.

O método de Gauss é o mais rudimentar de todos e possui custo computacional alto. Este método ainda é inflexível quanto a variações no vetor de termos independentes (b), o que significa que caso haja alterações neste vetor é preciso resolver todo o sistema novamente já que no processo de escalonamento o vetor de termos independentes também é considerado.

Decomposição LU

Na tentativa de melhorar o método de Gauss foi proposto um método um pouco melhor quanto a flexibilidade da solução que é a decomposição da matriz dos coeficientes em duas outras matrizes uma L e outra U, de tal forma que fica de acordo com a Equação 2.

$$A=LU$$

Equação 2 - Decomposição LU

Onde L é uma matriz triangular inferior com a diagonal principal igual a um e os termos abaixo dela iguais aos multiplicadores obtidos do escalonamento da matriz dos coeficientes e U é a matriz que resultou do escalonamento. Então o sistema linear será da forma descrita na Equação 3

$$LUx=b$$

$$Ux=y$$

$$Ly=b$$

Equação 3 - Sistema linear LU

Neste caso não é necessário fazer um novo escalonamento cada vez que o vetor de termos independentes é trocado já que o sistema $Ly=b$ é responsável por ajustar os valores de b para solução correta do sistema.

Decomposição de Cholesky

A decomposição de Cholesky surgiu da necessidade de um algoritmo mais eficiente para solução de sistemas lineares já que tanto o método de decomposição LU quanto o método de Gauss possui um custo computacional bastante alto. Este método então surgiu da observação que quando a matriz dos coeficientes é simétrica ($A=A^T$) e definida positiva ($vAv^T > 0$) é possível fazer uma decomposição de acordo com a Equação 4

$$A = LL^T$$

Equação 4 - Decomposição de Cholesky

Onde a matriz L é obtida de acordo com a

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}, j=1, 2, \dots, n.$$

Equação 5 - Diagonal principal da de decomposição de Cholesky L

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}}{l_{jj}}, j=1, 2, \dots, n-1 \text{ e } i=j+1, j+2, \dots, n$$

Equação 6 - Termos inferiores da matriz L de Cholesky

O sistema que utiliza a decomposição de Cholesky ficará da forma mostrada na Equação 7.

$$\begin{aligned} L^T x &= y \\ Ly &= b \end{aligned}$$

Equação 7 - Sistema linear de Cholesky

Decomposição LDL^T

Na decomposição de Cholesky as condições de utilização são muito rígidas já que a matriz tem que ser simétrica e definida positiva. A matriz precisa ser definida positiva para garantir que todos os termos da diagonal principal da matriz L seja reais, já que eles dependem de uma raiz quadrada. Uma solução para se flexibilizar as condições de utilização da decomposição de Cholesky é fazer com que a diagonal principal não dependa da raiz quadrada, então surgiu a decomposição LDLT que a matriz dos coeficientes precisa ser apenas simétrica.

ATIVIDADES

3.1 Implemente o algoritmo de substituições sucessivas e retroativas.

3.2 Resolva os sistemas abaixo utilizando os algoritmos implementados no item anterior.

$$\begin{aligned} \text{a)} \quad & \begin{cases} 2x - 3y = -1 \\ 5y = 4 \end{cases} \\ \text{b)} \quad & \begin{cases} 4x - 5y + 2z = 1 \\ 3y - z = 5 \\ 2z = -2 \end{cases} \\ \text{c)} \quad & \begin{cases} 4x + y - 4z + w = 0,5 \\ -2y + 8z - 3w = 7 \\ 9z - 4w = 3 \\ -10w = 30 \end{cases} \end{aligned}$$

3.3 Faça as operações que se pede sobre as matrizes utilizando o Octave.

$$\text{a)} \quad A = \begin{bmatrix} 3 & 5 \\ -2 & 4 \end{bmatrix}$$

i) Divida o elemento da segunda linha e primeira coluna pelo elemento da primeira linha primeira coluna executando o comando abaixo no Octave.

$$m_{21} = -A(2,1)/A(1,1)$$

ii) Multiplique toda a primeira linha pelo valor retornado utilizando o comando

$$L1 = m_{21} * A(1,:)$$

iii) Some a segunda linha com o resultado do item anterior utilizando o comando.

$$A(2,:) = L1 + A(2,:)$$

iv) O que se observa do resultado?

$$b) A = \begin{bmatrix} 2 & 1 & -3 \\ 4 & -2 & 5 \\ 1 & 2 & -7 \end{bmatrix}$$

i) $L = \text{eye}(3) \rightarrow$ Cria uma matriz identidade 3x3

ii) $L(2,1) = A(2,1)/A(1,1) \rightarrow$ O elemento $L(2,1)$ da matriz L recebe o resultado da divisão.

iii) $L(3,1) = A(3,1)/A(1,1) \rightarrow$ O elemento $L(3,1)$ da matriz L recebe o resultado da divisão.

iv) $A(2,:) = -L(2,1) * A(1,:) + A(2,:)$

v) $A(3,:) = -L(3,1) * A(1,:) + A(3,:)$

vi) $L(3,2) = A(3,2)/A(2,2)$

vii) $A(3,:) = -L(3,2) * A(2,:) + A(3,:)$

viii) Qual foi o resultado encontrado?

3.4 Faça o escalonamento das matrizes abaixo utilizando o Octave.

$$a) \begin{bmatrix} 3 & 1 & 4 \\ 8 & 1 & 2 \\ 2 & 5 & 6 \end{bmatrix}$$

$$b) \begin{bmatrix} -10 & 2 & 4 & 6 \\ 5 & -8 & 4 & 1 \\ 3 & 9 & 12 & 5 \\ -4 & 0 & 5 & 2 \end{bmatrix}$$

3.5 Utilizando o dispositivo prático encontre o sistema triangular inferior e resolva os sistemas abaixo com e sem pivotação parcial utilizando apenas 3 algarismos significativos na resposta. Determine o erro absoluto utilizando a fórmula:

$$\frac{\|x - x^{otimo}\|_{\infty}}{\|x^{otimo}\|_{\infty}}$$

Onde x^* é o valor exato da resposta ou aproximado utilizando 6 algarismos significativos.

$$a) \begin{cases} 5x - 2y + 2z = 2 \\ 3x + 2y + 4z = -1 \\ 4x - 3y + z = 3 \end{cases}$$

$$b) \begin{cases} x + y + z = 2 \\ x - 2y - 2z = -1 \\ 2x + y + z = 3 \end{cases}$$

$$c) \begin{cases} x + y + 6z = 11 \\ x + 1,5y + 2z = 4,5 \\ 6x + 2y + 0,5z = 14 \end{cases}$$

$$d) \begin{cases} 3x + 2y + 6z = 1 \\ 12x + 8y + 24z = 4 \\ 6x + 4y + 5z = 10 \end{cases}$$

$$e) \begin{cases} 4x + 5y + 7z = 5 \\ 2x + 4y + 3z = 3 \\ 6x + 3y + 8z = 8 \end{cases}$$

$$f) \begin{cases} 3x_1 + 7x_2 + 8x_3 + x_4 = 3 \\ 4x_1 + 5x_2 + 2x_3 + 8x_4 = 8 \\ 6x_1 + 3x_2 + 2x_3 + 5x_4 = 5 \\ 2x_1 + 4x_2 + x_3 + 3x_4 = 9 \end{cases}$$

$$g) \begin{cases} x_1 + 2x_2 - 3x_3 = 4 \\ 3x_1 - x_2 + 5x_3 = 2 \\ 4x_1 + x_2 + 2x_3 = -2 \end{cases}$$

$$h) \begin{cases} x_1 + x_2 + x_3 = 2 \\ 2x_1 + 3x_2 + 2x_3 = 5 \\ 2x_1 + 3x_2 + 2x_3 = (\sqrt{3} + 1) \end{cases}$$

$$i) \begin{cases} 3x_1 + x_2 + 5x_3 = 7 \\ x_1 + 10x_2 + 2x_3 = 3 \\ 5x_1 + 2x_2 + 11x_3 = 5 \end{cases}$$

$$j) \begin{cases} 12x_1 - 5x_2 + 4x_3 - x_4 + 8x_5 - 7x_6 = 0 \\ -5x_1 + 25x_2 + 7x_3 - 4x_4 + 9x_5 - 32x_6 = -3 \\ 4x_1 + 7x_2 - 6x_3 + 3x_4 - 3x_5 + 4x_6 = 14 \\ -x_1 - 4x_2 + 3x_3 + 17x_4 + 5x_5 + 2x_6 = 6 \\ 8x_1 + 9x_2 - 3x_3 + 5x_4 + x_5 + 15x_6 = 4 \\ -7x_1 - 32x_2 + 4x_3 + 2x_4 + 15x_5 + 23x_6 = 3 \end{cases}$$

$$k) \begin{cases} -3x_1 - 24x_2 + 5x_3 - 17x_4 = -152 \\ -24x_1 + 5x_2 - 2x_3 + 4x_4 = 31 \\ 5x_1 - 2x_2 + 3x_3 - 8x_4 = 64 \\ -17x_1 + 4x_2 - 8x_3 + x_4 = 11 \end{cases}$$

$$l) \begin{cases} 20x_1 - 7x_2 - 9x_3 - 2x_4 + 2x_5 = 92 \\ -7x_1 + 14x_2 + 6x_3 + 2x_4 - 5x_5 = 63 \\ -9x_1 + 6x_2 + 25x_3 + 4x_4 - 3x_5 = -235 \\ -2x_1 + 2x_2 + 4x_3 + 6x_4 - 4x_5 = 94 \\ 2x_1 - 5x_2 - 3x_3 - 4x_4 + 13x_5 = -61 \end{cases}$$

3.6 Implemente os métodos de decomposição LU, Cholesky e LDL^T em Octave e resolva os sistemas lineares do exercício anterior, com todos os métodos que forem possíveis. O método implementado deve receber como entrada as matrizes dos coeficientes e dos termos independentes e fornecer o vetor resultado, o erro e o determinante da matriz dos coeficientes, o programa deverá avaliar ainda se o sistema tem solução e se a solução é única. Compare os resultados.

3.7 Utilize os algoritmos implementados e implemente um algoritmo para inversão de matrizes. Inverta as matrizes que são possíveis inverter. O programa deverá determinar quais matrizes são possíveis de inverter.

AULA PRÁTICA 4 - Métodos iterativos estacionários

OBJETIVO

- ✓ Utilizar métodos aproximados para solução de sistemas lineares.

INTRODUÇÃO

Algumas vezes a solução exata de um sistema linear é muito difícil de se obter, por exemplo quando a matriz dos coeficientes é esparsa, ou então a resposta deste sistema não possui critérios muito rígidos de precisão, nestes casos um método iterativo pode ser mais eficiente do que um método exato. Os métodos iterativos utilizam a busca de uma solução criando uma sequência de aproximantes onde cada um dos quais são obtidos através do seu anterior. Um método iterativo qualquer pode ser representado pela Equação 8,

$$x^{k+1} = M x^k + c$$

Equação 8 - Métodos iterativos

onde,

$x^{k+1} \rightarrow$ É o aproximante da próxima iteração,

$M \rightarrow$ Matriz de iteração,

$x^k \rightarrow$ Aproximante da iteração atual,

$c \rightarrow$ Fator constante.

Condições de convergência

Um método é dito convergente se ele obedece a Equação 9

$$\lim_{K \rightarrow \infty} x^k = x^{opt}$$

Equação 9 - Equação de convergência

onde x^{opt} é a solução exata do sistema.

Teorema 4-1 – A condição necessária e suficiente do processo iterativo definido pela Equação 8 é que $\max |\lambda_i| < 1$, onde λ_i , são os autovalores da matriz de iteração M.

Verificar a convergência de um método utilizando o Teorema 4 -1 pode ser mais custoso computacionalmente do que encontrar a solução por um método exato já que a determinação de autovalores é uma tarefa de um alto custo computacional. Então foi proposta uma condição suficiente para convergência sendo ela exposta pelo Teorema 4.2.

Teorema 4-2 – É condição suficiente para a convergência dos métodos iterativos de Jacobi e Gauss-Seidel que a matriz dos coeficientes A seja diagonal estritamente dominante como mostra a Equação 10.

$$|a_{ii}| > \sum_{j=1}^n |a_{ij}|, i=1, 2, \dots, n$$

Equação 10 - Condição de convergência dos métodos de Jacobi e Gauss-Seidel

Métodos iterativos estacionários

Os métodos iterativos estacionários são aqueles nos quais a matriz de iteração não se altera ao longo de toda execução. Os métodos iterativos estacionários mais comumente utilizados para solução de sistemas lineares são os métodos de Jacobi, Gauss-Seidel e sobre relaxação sucessiva.

ATIVIDADES

4.1 Implemente os algoritmos de Jacobi, Gauss-Seidel. Verifique a condição de convergência e resolva os sistemas abaixo, com o(s) método(s) que é garantida a convergência. A solução deverá ter um erro máximo de 10^{-4} ou o número máximo de 100 iterações.

$$\text{a) } \begin{cases} 7x_1 + x_2 + 5x_3 = 7 \\ x_1 + 10x_2 + 2x_3 = 3 \\ 5x_1 + 20x_2 + 11x_3 = 5 \end{cases}$$

$$\text{b) } \begin{cases} 4x_1 + x_2 + 5x_3 + 3x_4 = 5 \\ 4x_1 + 5x_2 + 2x_3 + x_4 = 7 \\ 7x_1 + 6x_2 + 10x_3 + 5x_4 = 6 \\ 3x_1 + 9x_2 + x_3 + 5x_4 = 3 \end{cases}$$

$$\text{c) } \begin{cases} 5x_1 + 3x_2 + x_3 = 10 \\ 1,5x_1 + 10x_2 + x_3 = 2 \\ 0,7x_1 + 2,3x_2 + 2x_3 = 5 \end{cases}$$

$$\text{d) } \begin{cases} 21x_1 + 12x_2 + 5x_3 = 17 \\ 4x_1 + 45x_2 + 13x_3 = 11 \\ 9x_1 + 14x_2 + 32x_3 = 21 \end{cases}$$

$$\text{e) } \begin{cases} 0,3x_1 + 0,07x_2 + 0,1x_3 = 0,15 \\ 0,014x_1 + 0,05x_2 + 0,0023x_3 = -1,4 \\ 0,09x_1 + 0,05x_2 + 0,18x_3 = -0,31 \end{cases}$$

4.2 Compare os resultados para solução dos sistemas de cada método.

4.3 Implemente o algoritmo de sobre relaxação sucessiva e resolva os sistemas acima. Faça um algoritmo que testes vários ômegas diferentes e retorne o melhor resultado.

AULA PRÁTICA 5 - Análise de erros na solução de sistemas lineares

OBJETIVO

- ✓ Determinar a qualidade da solução de um sistema linear baseado na análise do erro da solução

INTRODUÇÃO

Quando uma pequena variação em qualquer uma das matrizes do sistema linear causa uma grande diferença na solução deste sistema dizemos que o sistema é malcondicionado. Através do número de condição, que é baseado em uma das normas da matriz de termos independentes, é possível determinar se o sistema é ou não malcondicionado.

ATIVIDADES

5.1 Faça um algoritmo que determine se o sistema é malcondicionado (obs.: não pode ser utilizada a função cond do Octave). Resolva os sistemas que não são malcondicionados pelos métodos de decomposição ou iterativos. Utilize o método mais barato computacionalmente que produza uma solução com um erro máximo de 10^{-4} . Some 0,01 e determine a diferença entre as duas soluções utilizando a fórmula:

$$\frac{\|x - x_{+0,01}\|_{\infty}}{\|x_{+0,01}\|_{\infty}}$$

$$a) \begin{cases} 3,250x_1 + 1,625x_2 + 1,083x_3 + 0,8125x_4 = 3,520 \\ 1,625x_1 + 1,083x_2 + 0,8125x_3 + 0,6500x_4 = 4,496 \\ 1,083x_1 + 0,8125x_2 + 0,6500x_3 + 0,5417x_4 = 4,008 \\ 0,8125x_1 + 0,6500x_2 + 0,5417x_3 + 0,4643x_4 = 3,490 \end{cases}$$

$$b) \begin{cases} -3x_1 - 24x_2 + 5x_3 - 17x_4 = -152 \\ -24x_1 + 5x_2 - 2x_3 + 4x_4 = 31 \\ 5x_1 - 2x_2 + 3x_3 - 8x_4 = 64 \\ -17x_1 + 4x_2 - 8x_3 + x_4 = 11 \end{cases}$$

$$c) \begin{cases} 3x_1 + 1,5x_2 + x_3 + 0,75x_4 = 32 \\ 1,5x_1 + x_2 + 0,75x_3 + 0,6x_4 = 16,2 \\ x_1 + 0,75x_2 + 0,6x_3 + 0,5x_4 = 10,85 \\ 0,75x_1 + 0,6x_2 + 0,5x_3 + 0,43x_4 = 8,16 \end{cases}$$

$$d) \begin{cases} 0,01x_1 + 0,4x_2 + 0,125x_3 + x_4 = 2,0685 \\ 0,4x_1 + 0,25x_2 + 2x_3 + 0,32x_4 = -0,085 \\ 0,125x_1 + 2x_2 + 0,5x_3 + 1,02x_4 = 2,4425 \\ x_1 + 0,32x_2 + 1,02x_3 + 0,045x_4 = -0,1424 \end{cases}$$

$$e) \begin{cases} 3x_1 + 4x_2 = 1 \\ -9x_1 - 12x_2 = 4 \end{cases}$$

5.2 Implemente o software para construção da matriz de Hilbert. Multiplique a matriz criada pelo algoritmo por 4, 8 e 7,45. Verifique o número de condição para as matrizes resultante dos produtos. Multiplique a linha 1 pelos mesmos fatores e repita a verificação. Descreva o que você observou.

AULA PRÁTICA 6 - INTERPOLAÇÃO POLINOMIAL

INTRODUÇÃO

A interpolação polinomial é uma técnica fundamental em qualquer engenharia. Pois toda engenharia é baseada na análise de dados e a partir destes dados a obtenção de uma determinada conclusão. A idéia fundamental é que com base em um conjunto de dados obtidos através de medidas realizadas é possível avaliar situações que não estão descritas pelos dados obtidos e com base nestas informações ser possível a obtenção de dados que estão no intervalo.

1. Interpolação através da solução de sistemas lineares

A interpolação através de sistemas lineares é um método fácil para solução de interpolação, no entanto seu custo envolve resolver um sistema linear o que muitas vezes têm que ser pela utilização do método de decomposição LU que é muito caro computacionalmente.

ATIVIDADES

6.1. Construa um algoritmo para interpolação que solucione o sistema linear utilizando o algoritmo de solução de sistemas lineares que seja mais barato. O seu algoritmo deverá escolher o método de solução de sistemas lineares mais barato.

6.2. Faça as seguintes interpolações utilizando o algoritmo implementado na questão anterior

- a) Para cada função geradora você determinará o y através da função. Com o conjunto de pares ordenados (x,y) você interpolará o valor que se pede utilizando o algoritmo de solução de sistemas lineares de ordem n. E então determinar o erro relativo para cada valor interpolado.

Função Geradora	x					Valor a Interpolat
i	0	1	2	3	4	
$f(x)=e^{\frac{x}{2}}x^2-10$	-8,154	-4,919				-5,00
	-6,166	-3,950	-1,871			-4,76
	-1,455	0,0	0,762	1,178		-0,83
$f(x)=5x^2+3x-10$	0,5	1,0				0,250
	-1,5	0,5	1,5			0,0
	-2,0	-1,5	0,0	2,0		1,5
	-4,0	-2,5	2,0	3,0	4,0	1,0
$f(x)=\frac{4}{e^{5x^2}}$	-3,11	5,196				2,0
	-3,11	-2,50	2,15			1,150
	-3,11	-2,00	0,0	3,150		-1,65
	-4,53	-2,56	-1,76	3,6	5,13	2,25

- b) Utilizando o algoritmo de Lagrange construa um algoritmo que retorne os coeficientes do polinômio interpolador de Lagrange.

- c) Utilizando a resposta da prática b determine os coeficientes dos polinômios do exercício a.

6.3. O que acontece com a precisão do método quando o grau do polinômio aumenta. Justifique a resposta baseado nos dados.

2. Polinômios de Lagrange

A precisão do método aumenta com aumento do grau do polinômio. Além do que este método é mais barato computacionalmente do que outros que utilizam interpolação utilizando a solução de sistemas lineares.

3. Atividades

Questão 1) Implemente o algoritmo de Lagrange para interpolação.

Questão 2) Faça as seguintes interpolações utilizando o algoritmo implementado na questão anterior

- a) Para cada função geradora você determinará o y através da função. Com o conjunto de pares ordenados (x,y) você interpolará o valor que se pede utilizando o algoritmo de Lagrange. E então determinar o erro relativo para cada valor interpolado.

Função Geradora	x					Valor a Interpolar
	0	1	2	3	4	
$f(x) = \frac{x}{e^2}$	-1,178	-0,6236				-1,00
	-1,178	-0,3236	-0,3464			-0,50
	0,485	0,9007	1,594	2,564		1,2
$f(x) = 3x^2 + 1$	-2,148	0,9007				-0,485
	-2,148	0,9007	0,00			-0,485
	-2,148	0,0	0,9007	1,732		-0,485
	-2,148	-1,532	-0,607	0,0	1,732	-0,485
$f(x) = \frac{\sin(x^2)}{e^x} - 1$	-2,564	-1,316				-2,286
	-2,564	-2,148	-1,316			-2,286
	-2,564	-2,148	-1,316	-0,9007		-1,732
	-2,564	-2,148	-1,316	-0,9007	0,6236	-0,2079

- b) Utilizando o algoritmo de Lagrange construa um algoritmo que retorne os coeficientes do polinômio interpolador de Lagrange.
- c) Utilizando a resposta da prática b determine os coeficientes dos polinômios do exercício a.

Questão 3) O que acontece com a precisão do método quando o grau do polinômio aumenta. Justifique a resposta baseado nos dados.

Questão 4) Resolva as questões 1, 2 e 3 do método de interpolação de Lagrange utilizando o polinômio de Newton.

AULA PRÁTICA 7 - INTERPOLAÇÃO UTILIZANDO SPLINES

INTRODUÇÃO

O objetivo da interpolação é determinar o valor da função em um ponto. Este ponto deverá estar entre os limites dos valores utilizados na interpolação. Nas técnicas vistas até agora o grau do polinômio aumenta à medida que o número de pontos cresce já para técnicas de splines, o grau do polinômio interpolador é sempre igual a 3, o que varia é a quantidade de polinômios que é utilizado para interpolação. O polinômio para interpolação é da forma da equação abaixo.

$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

No caso dos splines para cada par de pontos é construído um polinômio como o da equação mostrada acima que deverá obedecer as seguintes relações:

$$s_{i+1}(x_i) = s_{i+1}(x_{i+1})$$

$$s'_i(x_{i+1}) = s'_i(x_{i+1})$$

$$s''_i(x_{i+1}) = s''_i(x_{i+1})$$

Utilizando a equação e as relações acima é possível construir um sistema de equações que será da seguinte forma:

Como é possível observar o sistema acima possui 2 incógnitas a mais do que número de equações, tornando o assim o sistema subdeterminado. Utilizaremos duas técnicas para que seja possível diminuir o número de incógnitas no sistema acima, que é a técnica dos splines cúbicos naturais e splines cúbicos extrapolados.

Splines cúbicos naturais

Observando-se o comportamento dos pontos utilizados nas interpolações é possível verificar que este pontos, nas bordas da interpolação, que é para o intervalo entre o primeiro e o segundo ponto e entre o penúltimo e o último ponto, quase sempre apresentam um comportamento linear. Então teremos que $s''_0(x_0) = s''_n(x_n) = 0$ o sistema que determina as derivadas segundas da função ficará da forma:

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & 0 & 0 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & 0 & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \begin{bmatrix} s''_1(x_1) \\ s''_2(x_2) \\ s''_3(x_3) \\ \vdots \\ s''_{n-1}(x_{n-1}) \end{bmatrix} = 6 \begin{bmatrix} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \Delta y_3 - \Delta y_2 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{bmatrix}$$

Restando assim o mesmo número de equações e incógnitas, o que torna o sistema possível e determinado.

Splines cúbicos extrapolados

Os splines cúbicos extrapolados são utilizados quando $y = f(x)$, possui um ponto de inflexão nas extremidades. O sistema é obtido então fazendo $s'''_0(x_0) = s'''_1(x_1)$ e $s'''_{n-2}(x_{n-2}) = s'''_{n-1}(x_{n-1})$. Este sistema fica da seguinte forma:

$$\begin{bmatrix} \frac{(h_0+h_1)(h_0+2h_1)}{h_1} & \frac{h_1^2-h_0^2}{h_1} & 0 & 0 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & 0 & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \frac{h_{n-2}^2-h_{n-1}^2}{h_{n-2}} & \frac{(h_{n-1}+h_{n-2})(h_{n-1}+2h_{n-2})}{h_{n-2}} \end{bmatrix} \begin{bmatrix} s''_1(x_1) \\ s''_2(x_2) \\ s''_3(x_3) \\ \vdots \\ s''_{n-1}(x_{n-1}) \end{bmatrix} = 6 \begin{bmatrix} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \Delta y_3 - \Delta y_2 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{bmatrix}$$

ATIVIDADES

- 7.1. Implemente o algoritmo que faça interpolação utilizando splines cúbicos naturais e/ou extrapolados. O seu algoritmo deverá ainda criar um gráfico mostrando os pontos utilizados para construir os polinômios interpoladores, os pontos interpolados e o gráfico de cada polinômio interpolador no intervalo de interpolação. Os parâmetros de entrada deverão ser apenas os vetores x e y nada mais.
- 7.2. Utilize o algoritmo implementado acima para avaliar as funções na tabela 1. Utilize os dois métodos de splines cúbicos naturais e extrapolados para determinar a solução. Avalie qual dos métodos apresentou melhor resposta e porquê.

Função geradora	x					Valores a interpolar			
i	0	1	2	3	4				
$f(x) = e^{\frac{x}{2}} x^2 - 10$	-8,154	-4,919	-1,45	0	3,25	-6,00	-4,53	-0,67	2,5
	-6,166	-3,950	-1,871	1,23	4,5	-4,76	-2,5	1,45	3,76
	-1,455	0,0	0,762	1,178	3,5	-0,83	0,54	1,96	3,35
$f(x) = 5x^2 + 3x - 10$	0,5	1,0	4,6	5,7	8,2	0,250	1,56	4,98	7,37
	-1,5	0,5	1,5	4,76	7,34	0,0	1,34	2,98	6,67
	-2,0	-1,5	0,0	2,0	6,7	1,5	-1,3	0,43	5,38
	-4,0	-2,5	2,0	3,0	4,0	1,0	-3,87	1,35	3,56
$f(x) = \frac{4}{e^{5x^2}}$	-3,11	-2,196	-0,43	1,9	2,65	-2,0	-2,35	-0,025	2,40
	-3,11	-2,50	2,15	4,13	6,24	1,150	2,16	5,3	6,15
	-3,11	-2,00	0,0	3,150	4,52	-1,65	-2,12	0,42	2,35
	-4,53	-2,56	-1,76	3,6	5,13	2,25	-3,2	1,63	4,79

- 7.3. Utilize o algoritmo implementado para resolver o problema abaixo.

Seja a função

$$f(x) = \begin{cases} e^x, & -2 \leq x \leq 0 \\ x \sin(5x) + 1, & 0 \leq x \leq 4 \end{cases}$$

definida por n pontos distintos $(-2, f(-2)), (-2+h, f(-2+h)), (-2+2h, f(-2+2h)), \dots, (4, f(4))$, sendo $h = (4 - (-2))/(n-1)$. Deseja-se reconstruir a curva de $y=f(x)$ por meio dos splines cúbicos, com $m=121$ pontos. Faça a avaliação dos seus resultados, comparando splines cúbicos naturais e extrapolados.

AULA PRÁTICA 8 - Ajuste de curvas

INTRODUÇÃO

Quando se realiza medidas de um dado experimento o que se espera obter é informações sobre como medidas de variáveis diferentes se relacionam e a partir daí poder então decidir se estas medidas estão corretas ou não. Para fazer isto podemos utilizar um modelo já pré existente que relaciona as variáveis medidas quando isto acontece dizemos que o modelo é determinístico isto é as variáveis estão relacionadas entre si através de uma lei que normalmente é expressa por uma fórmula matemática. Algumas vezes estas medidas estão relacionadas por uma forma específica do conjunto de variáveis, por exemplo linear, quadrática ou exponencial, no entanto os parâmetros que relacionam as variáveis não são conhecidos. Quando se tem somente os valores das medidas e a partir daí quer se obter os parâmetros e a forma da relação entre as medidas dizemos que este modelo é empírico. O que faremos então é tentar encontrar um modelo ou fórmula matemática que relaciona as variáveis resposta (dependente) com as variáveis explicativas(independentes). O processo de obtenção destas medidas está sujeitos a erros de fontes e formas variadas isto torna o processo de inferir esta relação complicado então o que veremos são técnicas para determinar este modelo mesmo considerando este erro e qual o melhor modelo que se ajusta ao conjunto de dados obtidos.

REGRESSÃO LINEAR SIMPLES

O que se procura através de um modelo linear simples é determinar os parâmetros β_0 e β_1 de um modelo do tipo onde as variáveis resposta y estão relacionadas as variáveis explicativas x por uma relação do tipo $y = \beta_0 + \beta_1 x$.

ATIVIDADES

8.1. Com os dados das tabelas abaixo construa um gráfico para cada par x e y utilizando o Octave.

x	y	x	y	x	y	x	y
8,1710	39,2053	7,6784	0,2369	3,7796	0,3356	1,2867	16,4650
6,8664	30,7299	3,1249	-1,2767	1,8787	0,0450	-0,5849	23,2295
1,3960	11,2640	2,1721	-1,2050	-1,0432	0,0535	-0,8381	22,8824
0,8992	2,8130	1,3425	-2,8443	-4,1964	-0,0329	-0,9192	43,1092
-2,7101	-4,1482	-5,2367	-8,4783	-6,7271	-0,0214	-4,1909	77,7867

8.2. Utilizando apenas o primeiro e último dado de cada tabela determine um modelo linear para cada conjunto de dados. Desenhe o gráfico contendo o seu modelo sobre o gráfico de cada conjunto.

8.3. Determine o erro, determinando a distância entre cada ponto e a reta encontrada no seu modelo linear.

8.4. Implemente o algoritmo de regressão linear múltipla

8.5. Utilize algoritmo implementado acima para ajustar um modelo linear a cada conjunto de dados acima. Desenhe o gráfico contendo seu modelo quadrático sobre o gráfico de cada conjunto de dados e seu primeiro modelo, cada reta deverá ter uma cor diferente. Calcule também a variância residual e o coeficiente de determinação de cada modelo. Compare os resultados com os obtidos no item 2.

AULA PRÁTICA 9 - Regressão linear multivariáveis e ajuste polinomial

OBJETIVO

- ✓ Aprender utilizar o método de Regressão linear multivariáveis.

INTRODUÇÃO

O ajuste de curvas se justifica quando há a necessidade de um modelo que generalize a relação entre variáveis. Pode ocorrer um caso onde seja necessário relacionar mais de uma variável ao mesmo tempo. Quando há uma relação entre mais uma variável explicativa com uma mesma variável resposta dizemos que estas variáveis são correlacionadas e então é possível determinar uma modelo de correlação, o que não é o objetivo deste curso.

Uma das técnicas de determinação deste modelo é a regressão linear múltipla que consiste em minimizar a equação:

$$D(\beta_0, \beta_1, \dots, \beta_p) = \sum_{i=1}^n (y_i - u_i)^2$$

O que irá produzir o sistema:

$$\begin{bmatrix} n & \sum x_{i1} & \sum x_{i2} & \cdots & \sum x_{ip} \\ \sum x_{i1} & \sum x_{i1}x_{i1} & \sum x_{i2}x_{i1} & \cdots & \sum x_{ip}x_{i1} \\ \sum x_{i2} & \sum x_{i1}x_{i2} & \sum x_{i2}x_{i2} & \cdots & \sum x_{ip}x_{i2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_{ip} & \sum x_{i1}x_{ip} & \sum x_{i2}x_{ip} & \cdots & \sum x_{ip}x_{ip} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_p \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_{i1}y_i \\ \sum x_{i2}y_i \\ \sum x_{i3}y_i \\ \vdots \\ \sum x_{ip}y_i \end{bmatrix}$$

Através da solução deste sistema é possível encontrar os parâmetros de ajuste do modelo dado por **b**. A partir deste modelo é possível encontrar o coeficiente de determinação e a variância para o modelo.

ATIVIDADES

- 9.1. Implemente o algoritmo para regressão linear múltipla em qualquer linguagem de programação, seu algoritmo deverá determinar o condicionamento da matriz das variáveis explicativas.
- 9.2. Utilize o algoritmo implementado para responder a questão abaixo.

Uma empresa que vende por correio componentes de computadores pessoais, software e hardware possui um depósito geral para a distribuição dos produtos. Atualmente, a administração está examinando o processo de distribuição deste depósito e está interessado em estudar os fatores que afetam os custos de distribuição do depósito. Atualmente um pequeno encargo de manipulação se adiciona ao pedido, independentemente da quantidade pela que se fizeram. Foram coletados dados correspondentes a 24 meses em relação aos custos de distribuição, de depósito, as vendas e número de pedidos. A continuação apresenta os resultados:

Mês	Y	X ₁	X ₂	Mês	Y	X ₁	X ₂
1	52,95	386	4.015	13	62,98	372	3.977
2	71,66	446	3.806	14	72,30	328	4.428
3	85,56	512	5.309	15	58,99	408	3.964
4	63,69	401	4.262	16	79,38	491	4.582
5	72,81	457	4.296	17	94,44	527	5.582
6	68,44	458	4.097	18	59,74	444	3.450
7	52,46	301	3.213	19	90,50	623	5.079
8	70,77	484	4.809	20	93,24	596	5.735
9	82,03	517	5.237	21	69,33	463	4.269
10	74,39	503	4.732	22	53,71	389	3.708
11	70,84	535	4.413	23	98,18	547	5.387
12	54,08	553	2.921	24	66,80	415	4.161

- a) Ajuste os dados a um modelo de regressão com duas variáveis e interprete as estimativas dos parâmetros do modelo.
- b) Estime o custo de distribuição do depósito mensal da empresa quando as vendas são \$ 400.000 dólares e o número de pedidos é de 4.500.

9.3. Utilize a tabela abaixo para responder a próxima questão.

x	y
-6,0866	-509,9181
-5,9294	-493,0127
-5,2795	-291,7017
-5,0197	-160,6647
-4,7357	-182,8648
-1,8721	44,4050
-0,2061	-85,4323
0,1073	-33,8353
0,2787	-19,6766
7,3624	679,0813

- a) Utilizando os algoritmos implementados, ajuste o modelo de regressão linear e polinomial de graus 2 até 5 para os dados acima.
- b) Determine qual ajuste apresentou a melhor solução.
- c) Implemente o algoritmo de decomposição em valores singulares. Para o ajuste que apresentou a matriz de variáveis explicativas mais malcondicionada utilize o método de decomposição em valores singulares para estimar os parâmetros de ajuste do modelo.

AULA PRÁTICA 10 - Integração numérica

OBJETIVO

- ✓ Aprender utilizar técnicas de integração numérica

INTRODUÇÃO

Algumas vezes uma função que se quer integrar torna-se tão difícil de fazer a integração analítica que é preciso fazer a integração numérica. A integração numérica nada mais é do que aplicar o conceito de integral que é construir pequenas figuras que se conhece a área e somar a área destas figuras individuais para se obter a área total. Este método é chamado de integração numérica. A integração se torna tão melhor quando maior é o número de figuras que é utilizado no método ou quanto maior é o polinômio utilizado para integração.

Fórmulas de Newton-Cotes

É um método de integração onde a função que se quer integrar é aproximada por um polinômio interpolador de grau n . Esta função deve ser contínua no intervalo $[a,b]$ de integração. Para melhorar a resposta da integral pode-se aumentar o grau do polinômio e/ou dividir o intervalo de integração em intervalos menores.

ATIVIDADES

10.1. Implemente o algoritmo de Newton-Cotes para integração numérica. O algoritmo deverá fazer um gráfico da função junto com o polinômio interpolador para cada intervalo.

10.2. Utilize o algoritmo implementado acima para encontrar as integrais abaixo de acordo com as condições dadas.

Função	Graus do polinômio interpolador	Intervalo de integração	Quantidade de subintervalos
$f(x) = -3x^3 + \frac{3}{2}x^2 + 5$	1, 2 e 3	$[-1, 2]$	1 e 18
$f(x) = \frac{4x^2}{e^{5x}}$	3, 4 e 5	$[-0.5, 0.5]$	1 e 32
$f(x) = \frac{1}{x} 20 \sin(2x)$	1, 2 e 3	$[1, 7]$	3 e 15

10.3. Para cada função acima verifique se a regra que diz que quanto maior o grau do polinômio e a quantidade de intervalos melhor o resultado da integração é válida. Para isto determine o valor analítico da integral, o valor analítico pode ser determinado utilizando funções do Octave.

AULA PRÁTICA 11 - Integração numérica método de Gauss-Legendre

OBJETIVO

- ✓ Aprender utilizar técnicas de integração numérica utilizando a quadratura de Gauss-Legendre.

INTRODUÇÃO

O fato de escolher pontos igualmente espaçados simplifica consideravelmente os cálculos, no entanto, a determinação da integral poderia obter um resultado melhor se não tivéssemos esta imposição. A idéia da quadratura de Gauss-Legendre que não seja necessário dividir o intervalo de integração então é procurada uma função que minimize a diferença entre o valor real da integral e o valor determinado pelo método.

ATIVIDADES

- 11.1. Implemente os algoritmos necessários para resolver uma integral simples utilizando o método da quadratura de Gauss-Legendre.
- 11.2. Utilize o algoritmo implementado acima para encontrar as integrais abaixo de acordo com as condições dadas.

Função	Número de pontos	Intervalo de integração
$f(x)=4x^2+5$	1,2 e 3	[1,3]
$f(x)=3x^3e^{x^2}$	3,4 e 5	[1,3]
$f(x)=3x\cos(\pi x)$	1,2 e 3	$[-\pi/4, \pi/2]$

- 11.3. Para cada função acima verifique se a regra que diz que quanto maior o grau do polinômio e a quantidade de intervalos melhor o resultado da integração é válida. Para isto determine o valor analítico da integral, o valor analítico pode ser determinado utilizando funções do Octave.
- 11.4. Utilizando o algoritmo implementado para integração pelo método de Newton-Cotes resolva as integrais da questão 2 utilizando polinômios de grau 1 até 3 e o número de subintervalos iguais ao número de pontos utilizados na integração pelo método da quadratura de Gauss-Legendre. Compare os resultados com os obtidos na questão 2. Elabore uma expectativa da relação entre os métodos baseada nos resultados.

AULA PRÁTICA 12 - Integração dupla pelo método de Newton-Cotes

OBJETIVO

- ✓ Aprender utilizar técnicas de integração numérica dupla utilizando o método de Newton-cotes.

ATIVIDADES

- 12.1. Implemente o algoritmo necessário para resolver uma integral dupla utilizando o método de Newton-Cotes
- 12.2. Elabore e implemente um algoritmo simples para integração tripla. É possível extrapolar o método para integração n-upla?
- 12.3. Utilize o algoritmo implementado acima para encontrar as integrais duplas em relação a x e y abaixo de acordo com as condições dadas.

Função	Número de pontos	Intervalo de integração
$f(x, y) = -x^2 y + 5x$	1,2 e 3	$X=[2,3], y=[0,2]$
$f(x, y) = -2x^2 y + e^{(x+y)}$	3,4 e 5	$X=[10,11.5], y=[5,8]$
$f(x, y) = \cos(x+y)$	1,2 e 3	$x=[\pi/2, \pi], y=[\pi/4, \pi/2]$

- 12.4. Implemente o algoritmo necessário para resolver uma integral dupla utilizando o método de Newton-Cotes composta.
- 12.5. Utilize o algoritmo implementado acima para encontrar as integrais duplas em relação à x e y abaixo de acordo com as condições dadas.

Função	Número de pontos	Intervalo de integração
$f(x, y) = -x^2 y + 5x$	1,2 e 3	$X=[2,3], y=[0,2]$
$f(x, y) = -2x^2 y + e^{(x+y)}$	3,4 e 5	$X=[10,11.5], y=[5,8]$
$f(x, y) = \cos(x+y)$	1,2 e 3	$x=[\pi/2, \pi], y=[\pi/4, \pi/2]$

AULA PRÁTICA 13 - Raízes de Equações

OBJETIVO

- ✓ Aprender utilizar técnicas de determinação de raízes de equações.

INTRODUÇÃO

O problema de determinar em que ponto(s) uma função assume valor zero é recorrente em vários de problemas científicos e de engenharia. Este ponto é chamado de raízes da função. Algumas vezes as raízes podem ser determinadas de forma analítica, onde temos uma expressão que relaciona a função com suas raízes. Nem sempre isto é possível para funções com grau maior do que 4 ou funções transcendentais isto não é possível até mesmo porque estas raízes podem não existir. Neste caso podemos utilizar um método numérico para aproximação do valor destas raízes. Para tanto os algoritmos numéricos necessitam de 2 passos fundamentais que são:

- 1) Garantir que dentro de um intervalo fechado $[a,b]$ exista uma, e somente uma, raiz.
- 2) Produzir intervalos cada vez menores que irão convergir para o valor da raiz.

Método de Horner

Para que possamos determinar uma raiz então será preciso avaliar o polinômio em um determinado ponto. Fazer isto da forma convencional utilizando a forma exponencial é caro computacionalmente. Um método alternativo é utilizar o algoritmo de Horner para avaliar o polinômio. Este algoritmo utiliza a expressão abaixo para avaliar a função em um ponto. Este algoritmo possui complexidade de tempo igual a n para multiplicações e subtrações, que é bem menor que o método comum onde as multiplicações tem complexidade $n(n+1)/2$.

Limites das raízes reais

Um método computacional comum para resolver problemas computacionais é o método onde se testa todas as soluções de um problema até encontrar a solução procurada. Este método pode ser usado para encontrar as raízes de um polinômio, onde pode-se substituir valores neste polinômio até se encontrar um valor onde o polinômio é igual a zero. Um problema que surge para adotar esta solução é quais valores devem ser utilizados? O número de raízes encontradas é igual ao número de raízes do polinômio? Na tentativa de resolver estas questões foram propostos métodos de determinar os limites das raízes, isto não resolve o problema de determinar as raízes mas diminui o espaço de busca o que já é um grande passo na solução deste problema.

ATIVIDADES

- 13.1. Implemente o algoritmo de Horner para avaliação de um polinômio. Os parâmetros de entrada deverão ser apenas o vetor dos coeficientes e o valor do ponto onde se quer avaliar a função.
- 13.2. Faça uma modificação no algoritmo para que o algoritmo informe quantas raízes o polinômio tem e se ele tem pelo menos uma raiz real.
- 13.3. Faça a avaliação dos seguintes polinômios no ponto igual a 2, utilizando o algoritmo implementado acima.
 - a. $4x^3 + 2x^2 - 3$
 - b. $5x^4 - 10x^3 + 4x^2 - x^2 + 250x - 5$
 - c. $25x^{11} + 7x^2 - 32x + 4$
- 13.4. Implemente o algoritmo para determinar os limites das raízes reais de um polinômio.
- 13.5. Faça uma modificação no algoritmo acima para que faça o gráfico da função no intervalo onde estão as raízes.

13.6. Utilize o algoritmo implementado acima para determinar os limites das raízes das funções abaixo:

a. $x^3 - 3x^2 - 10x + 24$

b. $x^4 - 2x^3 - 13x^2 + 14x + 24$

c. $4x^3 + 25x^2 + 14x - 1$

13.7. Faça a análise dos resultados apresentados no item anterior considerando o gráfico apresentado.

13.8. Para os polinômios abaixo determine a raiz na proximidade do ponto que se pede utilizando os algoritmos dos métodos de aproximação linear visto.

Polinômio	Ponto	
$f(x) = 25x^2 + 3x - 4$	-1	1
$f(x) = x^5 + 3x^4 - 4x^3 - 8x^2 + 2x - 10$	-1	-10

13.9. Implemente os algoritmos baseados em aproximação quadrática e em tangente e resolva os exercícios da questão 13.8. e 13.6..

AULA PRÁTICA 14 - Equações Diferenciais Ordinárias

OBJETIVO

- ✓ Determinar a solução de um problema de equações diferenciais ordinárias utilizando métodos numéricos.

INTRODUÇÃO

Alguns problemas em engenharia e outras ciências exatas podem ser formulados em termos de equações diferenciais. Por exemplo, trajetórias balísticas, estudos de redes elétricas, curvatura de vigas e etc. De uma forma geral as equações diferenciais são utilizadas para fenômenos que são descritos através de uma taxa de variação.

A equação:

$$y' = f(x, y)$$

é chamada equação diferencial de primeira ordem. Nesta equação, f é uma função real dada, de duas variáveis reais x e y , sendo y uma função incógnita da variável independente x . As equações diferenciais podem ser classificadas em ordinárias e parciais. Uma equação diferencial é dita ordinária se a função dependente apenas de uma variável dependente. Uma equação diferencial é dita parcial quando depende de mais do que uma variável como no exemplo abaixo

$$\frac{\partial^2 u(x, y)}{\partial x^2} + \frac{\partial^2 u(x, y)}{\partial y^2} = 0$$

As equações diferenciais ordinárias podem ser classificadas de acordo com a ordem da derivada utilizada na determinação da função. Para esta aula prática utilizaremos EDO de primeira ordem.

ATIVIDADES

- 14.1. Implemente o algoritmo de Euler para solução de problemas de valor inicial (PVI).
- 14.2. Determine a solução para os seguintes PVIs utilizando o programa implementado na questão 14.1.:
 - a. $y' = \sqrt{x}$, $y(0) = 0$, $x \in [0, 2]$ e $m = 2, 3, 4, 5$
 - b. $y' = x^2 + y^2$, $y(1) = 0$, $x \in [1, 2]$ e $m = 2, 4, 6, 8$
 - c. $y' = xy$, $y(0) = 1$, $x \in [0, 1]$ e $m = 4, 6, 8, 10$
- 14.3. Determine a solução analítica dos problemas 14.2.. Compare o resultado obtido com a solução analítica. Faça o gráfico da solução analítica.
- 14.4. Para cada um dos problemas 14.2. determine a equação da reta que passa pelos pontos $[x_i, y_i]$ e $[x_{i+1}, y_{i+1}]$ e imprima junto com o gráfico da questão 14.3.. Avalie se a aproximação foi boa para cada um dos valores de m .
- 14.5. Implemente o algoritmo de Runge-Kutta de quarta ordem para solução de PVI.
- 14.6. Determine a solução dos seguintes PVIs utilizando o programa implementado na questão 14.5.:
 - a. $y' = x^2 + y$, $y(0) = 1$, $x \in [0, 2.0]$ e $h = 0.2, 0.4$ e 0.5
 - b. $y' = y(x - y) + 1$, $y(0) = 1$, $x \in [0, 1.5]$ e $h = 0.15, 0.3$ e 0.5
 - c. $y' = -2xy$, $y(0) = 1$, $x \in [0, 0.5]$ e $h = 0.1$ e 0.05 .
- 14.7. Determine a solução analítica dos problemas 14.6.. Compare o resultado obtido com a solução analítica. Faça o gráfico da solução analítica.
- 14.8. Para cada um dos problemas 14.6. determine a equação da reta que passa pelos pontos $[x_i, y_i]$ e $[x_{i+1}, y_{i+1}]$ e imprima junto com o gráfico da questão 14.7.. Avalie se a aproximação foi boa para cada um dos valores de m .
- 14.9. Resolva os problemas 14.2. e 14.6. utilizando os métodos de Dormand-Prince e Adams. Faça uma comparação para determinar qual o melhor método de solução para cada um dos casos.

Bibliografia

- Ascêncio, A. G., & de Campos, E. V. (2007). *Fundamento de programação de computadores: Algoritmos, Pascal, C/C++ e Java* (2ª ed.). São Paulo: Pearson Prentice Hall.
- Campos, F. F. (2007). *Algoritmos Numéricos* (3ª ed.). Belo Horizonte, Minas Gerais, Brasil: LTC.
- Franco, N. B. (2006). *Cálculo Numérico*. São Paulo, SP, B