

**Centro Brasileiro de
Pesquisas Físicas**

INTRODUÇÃO AO MATLAB

AUTORES:

Alan Tavares Miranda
Márcio Portes de Albuquerque
Marcelo Portes de Albuquerque

SUMÁRIO

1.	Introdução	4
2.	Interface	4
3.	Operações mais freqüentes	6
4.	Gráficos 2D	7
5.	Gráficos 3D	7
6.	Linhas de Dados	8
7.	Matrizes e Vetores	9
7.1	Geração de Vetores.....	9
7.2	Geração de matrizes	9
7.3	Selecionando elementos de uma matriz	9
7.4	Operações com Matrizes e alguma funções de geração	10
8.	Sistema de equações lineares	10
9.	Funções	11
10.	Operadores Relacionais.....	12
11.	Operadores Lógicos	12
12.	Funções Úteis (“any” e “all”).....	13
13.	Polinômios.....	13
14.	Máximo e Mínimo absolutos de uma função	13
15.	Controladores de Fluxo (flow control).....	14
16.	Variáveis Simbólicas	15
16.1	Somatórios	16
16.2	Algumas Funções de Calculo.....	16

16.3	Algumas funções de simplificação ou modificação de expressões simbólicas.....	16
16.4	Substituindo valores numéricos em expressões simbólicas.....	16
16.5	Raízes de um polinômio	16
16.6	Resolução de sistemas quaisquer.....	16
17.	Formatação de dados	17
17.1	PRINT	17
17.2	Gravando dados num arquivo : FID,FPRINTF,FCLOSE	17
17.3	Alguns formatos	18

1. Introdução

O **MATLAB** é um software destinado a fazer cálculos com matrizes. Os comandos do MATLAB são muito próximos da forma como escrevemos expressões algébricas, tornando mais simples o seu uso, podendo ser incorporados às rotinas pré-definidas, pacotes para cálculos específicos.

Esta nota técnica objetiva dar ao leitor uma visão geral do software MATLAB (*Matrix Laboratory*) como uma ferramenta matemática e uma linguagem de programação de alto nível, mostrando suas principais funções básicas como, por exemplo: construção de gráficos e compilação de funções. Algumas ferramentas um pouco mais avançadas como, por exemplo, funções específicas de cálculo e variáveis simbólicas também serão apresentadas no decorrer desta nota técnica.

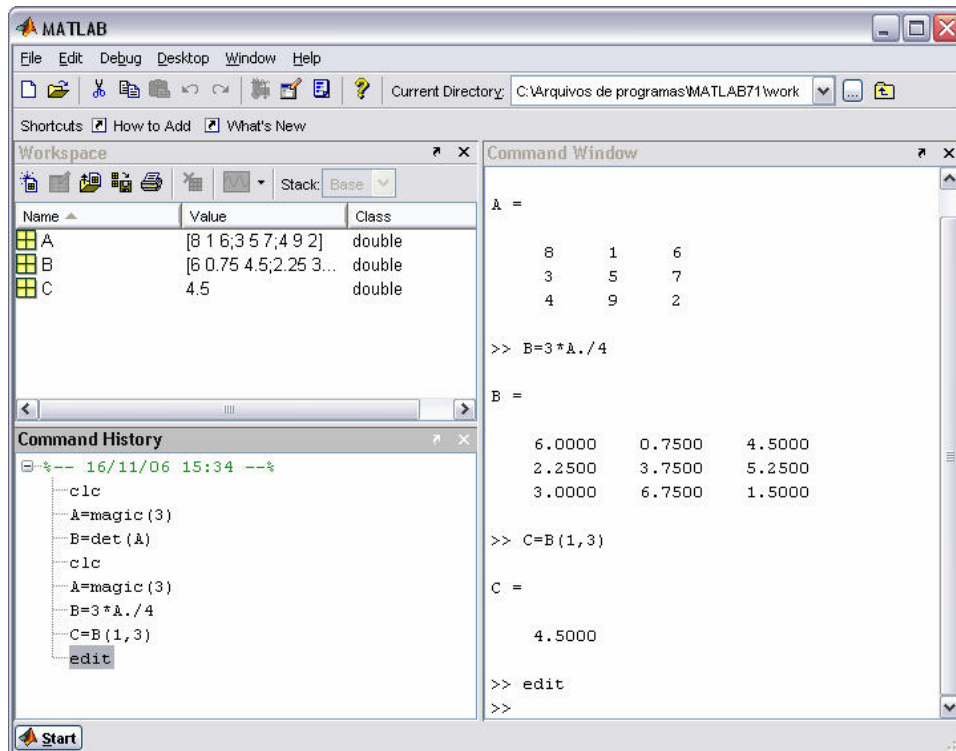
A grande vantagem que o MATLAB possui em relação à outras linguagens como, por exemplo, o C e o Fortran consiste no fato de que no MATLAB as informações são facilmente armazenáveis em matrizes o que proporciona uma fácil e rápida manipulação de uma grande quantidade de informações. Além disso, o MATLAB possui uma grande quantidade de bibliotecas auxiliares ("*Toolboxes*") que otimizam o tempo gasto para realizar tarefas, uma vez que, o usuário poderá utilizar muitas funções já definidas, poupando o tempo de criá-las.

Por outro lado, infelizmente, os programas feitos são difíceis de serem executados num ambiente fora do MATLAB. Criar um compilador que torne-o executável numa plataforma onde não exista o MATLAB não é uma tarefa fácil e geralmente os programas feitos são dependentes comercialmente devido a isso, uma vez que, junto à venda do aplicativo, indiretamente estará a venda do MATLAB.

A grande idéia deste artigo é ser um ponto inicial para leitores não familiarizados com a ferramenta computacional **MATLAB** e servir de base para o estudo de outras áreas mais específicas como o **processamento digital de imagens**.

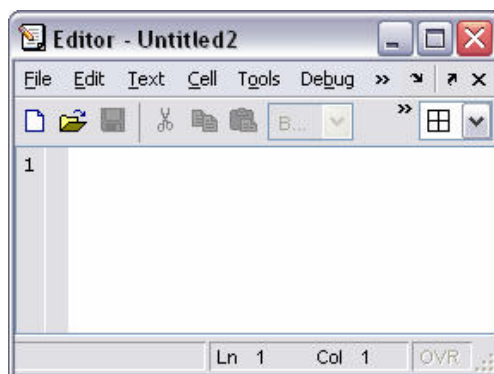
2. Interface

O primeiro passo para iniciarmos nosso estudo do MATLAB é nos familiarizarmos com a interface do programa.



- Command Window: Local onde as operações podem ser diretamente feitas.
- Workspace: espaço destinado às variáveis que estão salvas na memória, onde é possível visualizar o nome, valor e classe da mesma.
- Command History: Lista de comandos realizados, organizados por data de execução, permitindo o comando ser realizado novamente com duplo clique.

Caso se deseje criar procedimentos lógicos à serem realizados de forma que estes fiquem salvos em arquivo deveremos fazer uso do editor através do comando edit no command window.



**prompt do editor*

Para maiores informações basta teclar F1 para abrir o help do MATLAB.

3. Operações mais freqüentes

O MATLAB, além de uma poderosa ferramenta matemática, é uma prática e otimizada linguagem de programação. Abaixo temos algumas funções importantes:

abs(x) – Valor absoluto de x.

log(x) – logaritmo de x na base e.

loga(x) – logaritmo de x na base a.

rem(x,y) – resto da divisão de x por y.

eps – menor número tal que, quando adicionado a 1, cria um número maior que 1 no computador.

* É somado a denominadores de frações para evitar que se tornem zero, eliminando problemas com restrição.

inf – infinito

help – descrição de uma função .

clc- limpa a tela.

whos – mostra e descreve as variáveis q estão no workspace.

clear – remove todas as variáveis do workspace.

std(x) – desvio padrão.

ceil(x) – numero inteiro que corresponde ao arredondamento para cima.

floor(x) - numero inteiro que corresponde ao arredondamento para baixo.

fix(x) – arredondamento para o inteiro mais próximo de zero.

round(x) – arredondamento para o inteiro mais próximo.

format rat – números mostrados na forma de fração.

format short - números mostrados com 5 dígitos

format long - números mostrados com 15dígitos.

format compact - mostra separadamente parte inteira e parte decimal.

load – abre no MATLAB um arquivo de dados

* Funções são escritas na forma função(argumentos), onde os argumentos variam de acordo com a função, incluindo o uso dos operadores.

4. Gráficos 2D

ezplot('x(t),y(t)') - plota função 2D parametrizada ou não sem intervalo definido.

ezplot('x(t),y(t)'),[a,b],[b,c] - função 2D parametrizada ou não com intervalo definido.

* A função deve ser escrita de forma implícita ou explícita não escrevendo, neste caso, a variável dependente.

ezpolar('h(t)') - plota gráfico em coordenadas polares; precisa de uma única função.

fplot(['f(x),g(x)'],[a,b]) - plota vários gráficos num único plano cartesiano. É necessário especificar um intervalo de domínio.

5. Gráficos 3D

ezplot3('x(t)', 'y(t)', 'z(t)') – plota curva 3D parametrizada.

ezsurf('f(x,y)') - plot de superfície

Argumentos opcionais: *colorbar* – escala de cores

shading interp – retira linhas

shading faceted – retorna as linhas

* *Ezsurf* também pode plotar gráfico de funções parametrizadas.

ezsurf('f(x,y)') - plota gráfico e curva de contorno.

ezmesh('f(x,y)') – plota gráfico da superfície com linhas.

ezmeshc('f(x,y)') - plota gráfico da superfície com linhas + curva de contorno.

ezcontour('f(x,y)') - curva de contorno.

surf([matriz]) – Utiliza plotagem através da informação de uma matriz de pixels.

6. Linhas de Dados

Sendo x, y, z e w vetores temos :

plot(x,y, 'x') – plota pontos (não sob a forma de gráfico)

plot(x,y) - plota gráfico dos pontos

plotyy(x,y,z,w) – plota gráfico tendo-se referencia num eixo coordenado à esquerda e outro a direita.

plot(y) – plota um eixo imaginário e um real de um número complexo.

*Se y for função *plot* também poderá ser usado para plotar gráficos, porém devemos estar atentos aos operadores de divisão, multiplicação e exponenciação para evitar problemas de dimensões da matriz.

.- multiplicação elemento por elemento.

./ - divisão elemento por elemento.

.^- exponenciação elemento por elemento.

semilogx – escala logarítmica no eixo x .

semilogy - escala logarítmica no eixo y .

loglog – escala logarítmica nos eixos x e y .

legend('L1','L2'...'LN') – coloca caixa de legenda.

axis([a,b],[c,d]) – estipula os valores máximos e mínimos dos eixos.

axis tight – os limites dos eixos são os limites dos dados.

axis auto – restaura os limites padrões do MATLAB.

hold on – deve ser usado na linha de comando de uma plotagem para manter o gráfico plotado anteriormente e colocar um novo no mesmo plano cartesiano.

figure(n) – coloca um novo gráfico em um outro plano cartesiano, figura n .

subplot(a,b,c) – plota vários gráficos numa mesma figura, sendo a altura, b largura e c posição do gráfico.

7. Matrizes e Vetores

Definição: Vetor é um elemento que armazena valores, podendo ser um vetor linha ou coluna. Matriz é um conjunto de vetores especialmente disposto.

7.1 Geração de Vetores

O operador ':' indica da onde ate aonde

$v=[a:b]$ – gera vetor com valores de a até b espaçados de 1.

$v2=[a:c:b]$ - gera vetor com valores de a até b espaçados de c.

$v3=linspace(a,b,n)$ – gera vetor linearmente espaçado de a até b com n elementos.

7.2 Geração de matrizes

Sendo a matriz $A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$,

Para criar a matriz a fazemos: $A=[a,b,c;d,e,f;g,h,i]$

Transposta de A: $TA=[a,b,c;d,e,f;g,h,i]'$

7.3 Selecionando elementos de uma matriz

Considerando a matriz A selecionamos elementos da seguinte forma:

$B=A(i:m;j:n)$ – seleciona elementos da matriz a que estejam da linha i ate a linha m e da coluna j até a coluna n e armazena em B.

$C= A(:,n)$ – seleciona todos os elementos contidos na coluna n e armazena em C

$D=A(m,:)$ - seleciona todos os elementos da linha m e armazena em D.

$E=A(end,end-n)$ – seleciona o elemento que esteja na última linha e na última coluna menos n.

$v=A(:,)$ – dispõe os elementos da matriz A em um vetor , coluna em baixo de coluna.

7.4 Operações com Matrizes e alguma funções de geração

$A*B$ – multiplica a matriz A pela B.

$n*A$ – multiplica a matriz A pelo escalar n.

$\det(A)$ – determinante de A.

$\text{inv}(A)$ – inversa de A.

$\text{eye}(i,j)$ – cria matriz identidade.

$\text{zeros}(i,j)$ – cria matriz de zeros.

$\text{ones}(i,j)$ - cria matriz de uns.

$A.*B$ – multiplicação de matrizes elemento por elemento.

$\text{randn}(i,j)$ - gera matriz aleatória.

$\text{rand}(i,j)$ – gera matriz randômica com elementos variando entre 0 e 1.

$\text{magic}(N)$ – gera matriz quadrada de ordem N de números inteiros que têm a soma de cada linha, coluna ou diagonal principal iguais.

$S=\text{sum}(A(:))$ – soma todos os elementos da matriz A.

$S=\text{sum}(A(x:y))$ – soma todos os elementos da matriz A indo de x ate y.

* A contagem é feita de cima para baixo, da esquerda para a direita pelas colunas da matriz.

$S=\text{sum}(A)$ – soma todos os elementos de cada coluna separadamente e os coloca numa matriz linha.

8. Sistema de equações lineares

Sendo
$$\begin{cases} x + 2y - z = 1 \\ -2x - 6y + 4z = -2 \\ -x - 3y + 3z = 1 \end{cases}$$
 podemos facilmente resolver da seguinte forma:

$$A = \begin{bmatrix} 1 & 2 & -1 \\ -2 & -6 & 4 \\ -1 & -3 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \quad \text{e} \quad w = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Fazemos $w = \text{inv}(A) * b$ ou $w = A \setminus b$ onde w é a matriz com as raízes.

* Note que o operador ' \setminus ' serve para calcular a inversa de A .

9. Funções

Escrever funções possui grande aplicabilidade caso esta seja utilizada varias vezes em alguma aplicação.

Escrevendo uma função

Primeiramente deve-se abrir um M-file (no MATLAB, File → New → M-file).

No editor utilizamos a função *function* para criar uma função.

Ex.

```

Editor - Untitled*
File Edit Text Cell Tools Debug Desktop
function[y]=f(x)
    Term1=1./x;
    Term2= x.^2;
    y= term1+term2-6;
Ln 5 Col 1 OVR
  
```

```
function[y]=f(x)
```

```
    Term1=1./x;
```

```
    Term2= x.^2;
```

```
    y= term1+term2-6;
```

* Note que o operador ';' usado no fim de uma linha suprime a saída da tela.

* y é dado de saída e x é dado de entrada.

* no command window podemos chamar essa função exemplo facilmente chamando $f(n)$ onde n é um valor de x a ser colocado na função.

Depois de feita a função devemos salvá-la num arquivo de extensão `.m`.

Outra forma de definirmos uma função é através do comando inline diretamente no 'command window', da seguinte forma:

$z = inline('x+y')$, onde a função estará funcionando na forma $z(x,y)=x+y$

10. Operadores Relacionais

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

- Expressões do tipo $3+4 == 7$, terão como resultado 1, pois a sentença é verdadeira. Caso fosse falsa a resposta seria 0.

11. Operadores Lógicos

Símbolo	Operador
&	E
	Ou
~	nao

“~” inverte uma determinada condição lógica.

12. Funções Úteis (“any” e “all”)

Para mostrar como usá-las vamos observar o exemplo abaixo:

`x= [1,2,3,4,5,6]`

`any(x>5)` % algum maior que 5 ?

`ans = 1` (sim)

`all(x>5)` % Todos os valores de x são maiores que 5 ?

`ans = 0` (não)

* Observe que o símbolo “%” é utilizado para fazer comentários, não interferindo em nada no processamento.

13. Polinômios

Achando raízes

1º passo : Escrever os coeficientes do polinômio em forma de vetor

2º passo: Usar o comando `roots(p)`, onde p é o vetor coeficiente.

`polyval(p,[a,b,c])` – calcula o valor do polinômio p nos pontos de abscissa a,b e c.

`conv(p1,p2)` – multiplica o polinômio p1 pelo p2.

`deconv(p1,p2)` – divisão de polinômios

14. Máximo e Mínimo absolutos de uma função

`xmin=fminbnd('f',a,b)` – abscissa do ponto de mínima da função no intervalo [a,b].

* Deve-se abrir o arquivo salvo e trocar o sinal de todos os termos da função para depois aplicar `fminbnd` caso deseje calcular o ponto máximo.

* Não existe a função `fmax`, por isso utilizamos o procedimento acima.

`fzero('f',p)` – acha o ponto mais próximo onde a função muda de sinal tendo como referencia um ponto p dado.

15. Controladores de Fluxo (flow control)

O MATLAB, além de uma poderosa ferramenta matemática, também é uma linguagem de programação. Estão listados abaixo algumas estruturas lógicas utilizadas.

A sintaxe do IF é

```
if ...teste
.....
elseif
.....
else
.....
end
```

* O *elseif* e o *else* não são obrigatórios, mas o *end* é.

A sintaxe do FOR é

```
for ...variável=vetor
.....
end
```

* O valor de *variável* recebe o conteúdo de cada coluna de *vetor* seqüencialmente e, para cada conteúdo que receba, executa o corpo do FOR.

A sintaxe do WHILE é simples;

```
while....teste,
.....
end
```

* Onde teste é uma condição lógica que restringe o processo.

INPUT

É usado para inserção interativa de dados, sua sintaxe é a seguinte.

```
f=input('escrever qual dado de entrada:');
```

O MATLAB também possui outros recursos para facilitar o controle do fluxo do algoritmo :

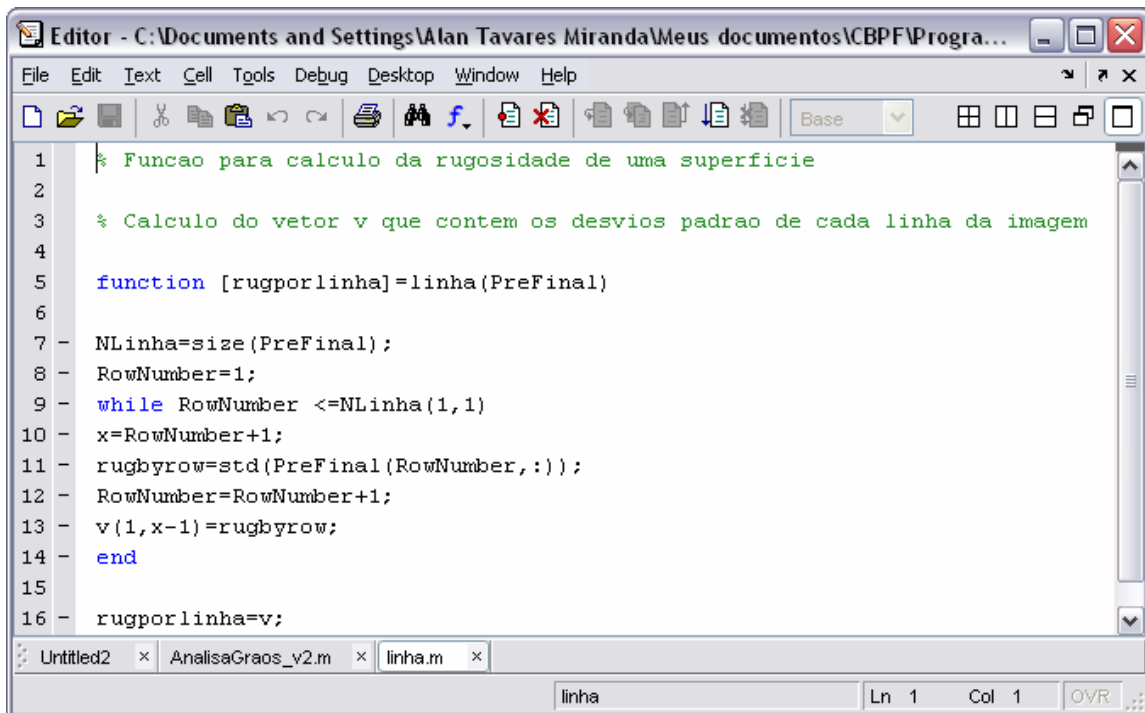
Break – determina uma saída imediata de uma estrutura

Continue – interrompe o fluxo do programa e recomeça um loop

switch...case – parecido com o IF, mas significa mais uma alternância. É um teste para múltiplos casos.

return – causa a saída imediata de uma função.

EXEMPLO



```

1  % Funcao para calculo da rugosidade de uma superficie
2
3  % Calculo do vetor v que contem os desvios padrao de cada linha da imagem
4
5  function [rugporlinha]=linha(PreFinal)
6
7  - NLinha=size(PreFinal);
8  - RowNumber=1;
9  - while RowNumber <=NLinha(1,1)
10 - x=RowNumber+1;
11 - rugbyrow=std(PreFinal(RowNumber,:));
12 - RowNumber=RowNumber+1;
13 - v(1,x-1)=rugbyrow;
14 - end
15
16 - rugporlinha=v;

```

16. Variáveis Simbólicas

O MATLAB usa objetos simbólicos para representarem variáveis e expressões. Uma expressão simbólica é uma expressão que contém objetos simbólicos e um objeto simbólico é uma estrutura de dados do tipo cadeia de caracteres (string).

Os objetos simbólicos podem ser criados através das funções *sym* e *syms* da seguinte forma:

$v = \text{sym}(\text{'expressão'})$ – cria uma variável simbólica para representar a expressão.

$\text{syms } x, y, z, \dots$ – cria as variáveis simbólicas x, y, z, \dots

$\text{double}(x)$ – retorna o valor numérico da expressão simbólica x com dupla precisão.

* Depois de criar uma variável simbólica, esta poderá ser usada em operações da mesma forma que as variáveis numéricas.

16.1 Somatórios

$\text{symsum}(f,x,a,b)$ – retorna $\sum_a^b f(x)$

16.2 Algumas Funções de Calculo

$\text{limit}(f,x,a)$ – retorna o limite de $f(x)$ quando $x \rightarrow a$

$\text{limit}(f,x,a, 'right')$ - retorna o limite de $f(x)$ quando $x \rightarrow a^+$

$\text{limit}(f,x,a, 'left')$ - retorna o limite de $f(x)$ quando $x \rightarrow a^-$

$\text{diff}(f,x,n)$ – retorna a n-ésima derivada de $f(x)$

* A função “*diff*” também opera em matrizes.

16.3 Algumas funções de simplificação ou modificação de expressões simbólicas

$\text{simple}(p)$ – simplifica uma expressão simbólica p .

$\text{expand}(P)$ – expande todos os termos de p .

$\text{factor}(p)$ – tenta representar p como produto de polinômios.

16.4 Substituindo valores numéricos em expressões simbólicas

$g=\text{subs}(f,'a')$ – substitui um valor a na expressão simbólica f .

$f1=\text{eval}(g)$ – transforma a expressão simbólica g numa expressão numérica $f1$.

16.5 Raízes de um polinômio

1º passo - definir variável simbólica. Ex. $\text{syms } x$

2º passo - definir uma variável qualquer para a expressão simbólica. Ex. $p1$

3º passo- usar $\text{solve}(p1)$

16.6 Resolução de sistemas quaisquer

Utiliza os mesmos passos da obtenção das raízes de um polinômio. As diferenças estão em definir-se como simbólica todas as variáveis das equações e escrever `solve(e1,e2,e3....en)`

* O processamento simbólico é muito mais preciso que o processamento numérico, porque as operações com valores números produzem erros de arredondamento e que vão se acumulando em operações sucessivas, enquanto que as expressões simbólicas não geram esses erros, pois não efetuam cálculo numérico.

* Os erros de arredondamento ocorrem, pois a cada operação que gera um valor com número de casas maior que o estabelecido, ocorrerá arredondamento ou truncamento caso as operações sejam feitas normalmente, porém se forem feitas com variáveis simbólicas, o mesmo não ocorrerá, uma vez que, as operações serão acumuladas e feitas ao fim do processamento, se assim for desejado, fazendo que ocorra apenas uma perda de precisão.

17. Formatação de dados

17.1 PRINT

Para gravar um dado diretamente sob forma de arquivo usamos o comando `print` :

```
'cd destino' , print('nome','dextensao do arquivo')
```

* Usamos o comando `print` para enviar para arquivo dados já formatados, como por exemplo, gráficos.

* Se escrevermos `print` sozinho e tiver algum gráfico na memória, este será enviado para a impressora diretamente.

17.2 Gravando dados num arquivo : FID,FPRINTF,FCLOSE

1º passo: Criação do identificador de arquivo – `fid` - .

```
fid=fopen('nome do arquivo.extensão', 'modalidade')
```

*modalidade: `w` – para gravação; `r` – para leitura.

* na modalidade de gravação, se o arquivo não existir, ele será criado.

* consultar o manual do MATLAB para obter outras modalidades.

2º passo: Determinação de formato – `fprintf`

```
fprintf(fid, formato,dados)
```

17.3 Alguns formatos

Especificação	Descrição
%c	caractere único
%d	notação decimal
%e	notação exponencial
%s	caractere string

* /n – pula linha

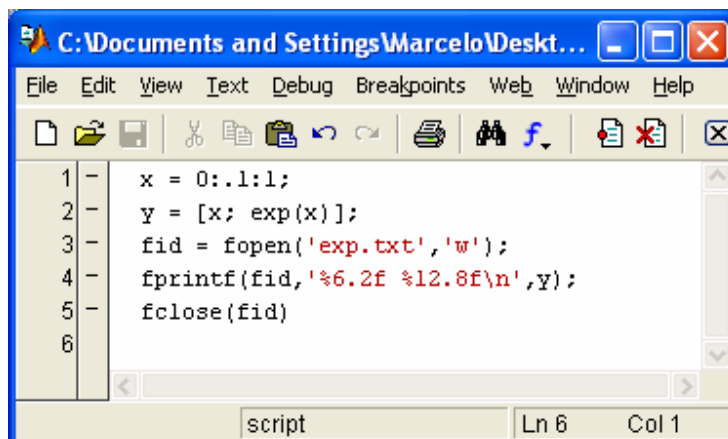
* para obter outros formatos consultar um manual de C.

3º passo : fechar identificador de arquivo

fclose(fid) – fecha um identificador de arquivo.

fclose(all) – fecha todos os identificadores abertos.

EXEMPLO



The screenshot shows a Notepad window titled "C:\Documents and Settings\Marcelo\Desk...". The menu bar includes File, Edit, View, Text, Debug, Breakpoints, Web, Window, and Help. The toolbar contains icons for file operations and editing. The text area contains the following C code:

```

1 - x = 0:.1:1;
2 - y = [x; exp(x)];
3 - fid = fopen('exp.txt','w');
4 - fprintf(fid,'%6.2f %12.8f\n',y);
5 - fclose(fid)
6

```

The status bar at the bottom indicates "script" and "Ln 6 Col 1".

O arquivo de saída estará disposto da seguinte maneira

0.00 1.00000000

0.10 1.10517092..