Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Matemática

Curso Introdutório ao Matlab

Autor: Volmir Eugênio Wilhelm

Curitiba, 10 de Julho de 2006.

Sumário

1. Introdução ao Matlab

Matlab: Matlab é um programa versátil para computadores, tendo como parte central as ferramentas para álgebra linear. A sigla Matlab vem de *MATrix LABoratory*. O código do programa foi desenvolvido em linguagem C sendo melhorado à medida que se tornam disponíveis novas versões. [Bernard Kolman, Introdução a Álgebra Linear, 1996]

1.1. Básico

- Digitando help aparece na tela uma lista de sudiretórios do Matlab e outros diretórios contendo arquivos que correspondem a comandos e conjuntos de dados.

- Digitando help nome, onde nome é o nome de um comando, fornece informações sobre esse comando.

- Uma vez carregado o programa Matlab, aparecerá o logotipo do programa e o símbolo >>, indicando que o Matlab está aguardando um comando.

- Após digitar o comando deve-se apertar a tecla ENTER para ser executado pelo Matlab.

- À medida que são digitados comandos, Matlab salva um certo números de comandos utilizados numa pilha. Comandos anteriores podem ser recuperados utilizando-se a tecla ↑.

- Comandos que não cabem numa linha podem ser continuados em outra linha utilizando-se **...** (três pontos consecutivos).

- Para parar a execução de um comando utiliza-se Ctrl+C.

- Comandos finalizados por **;** (ponto e vírgula) indica que os resultados não deverão aparecer na tela.

- O Matlab difere letras minúsculas de minúsculas.

- O comando disp mostra na tela o valor de variáveis e também mostra mensagens.

- Qualquer comando do DOS pode ser executado digitando antes ! (exclamação).

- Quando o usuário inicializa uma sessão do Matlab, o diretório de trabalho será c:\matlab\bin. O usuário pode alterar a pasta de trabalho com os comandos cd.. e cd.

- Os comandos pwd e cd mostram em qual diretório o usuário está no momento.

- Executando o comando exit, o usuário encerra a sessão atual do Matlab.

- O comando diary lista.txt grava todos os comandos digitados na tela de trabalho do Matlab e os resultados gerados no arquivo lista.txt. Usar o comando diary off desativa a função diary.

1.2. Matemática simbólica

Matlab versão 6.08 apresenta recursos para trabalhar com matemática simbólica

```
>>f = 'sin(x)'
```

```
>>syms x              %transforma x em símbolo
>>g = sin(x)
>>h = x^2 + 2x - 1
>>fplot(g,[0,5*pi/2])
>>subs(g,'x',pi)      %cálculo de g(pi)
>>int(g,x)            %integração de g em relação a x
>>diff(h,x)           %derivada de h em relação a x
>>double
>>char
```

1.3. Digitando uma matriz

```
>>A=[1 2 3]           %vetor linha (separado por espaços ou vírgulas)
>>B=[1;2;3]           %vetor coluna (separado por ponto e vírgula)
>>C=[1 2 3; 4 5 6]    %matriz 2x3
>>disp('A matriz A é:'), disp(A)
```

1.4. Variáveis, expressões e alguns comandos básicos

workspace: espaço de trabalho do Matlab. Armazena as variáveis e seus valores.
who: lista todas as variáveis armazenadas na área de trabalho do Matlab.

```
Your variables are:
A          B          C
```

whos: lista as variáveis e a quantidade de *bytes* ocupados por cada uma.

```
 Name        Size     Elements    Bytes     Density    Complex
   A       1 by 3         3         24        Full        No
   B       3 by 1         3         24        Full        No
   C       2 by 3         6         48        Full        No
```

save: salva num arquivo *.mat a área de trabalho

```
>>save A.txt A -ascii;
>>save A.mat A -ascii;
```

load: carrega o que foi salvo num arquivo com extensão *.mat

```
>>load A.txt A -ascii;
>>load A;
```

variavel = expressão

```
>>raiz=sqrt(6)  %a variável raiz assume 2.4495
```

ans: omitindo o nome da variável (e o sinal de igualdade) Matlab cria automaticamente uma variável com nome ans. Esta variável pode ser acessada imediatamente após sua criação.

expressão

```
>>sqrt(-6)        %Matlab cria a variável ans com valor  (0 + 2,4495i)
```

clear: limpa o conteúdo de algumas ou de todas as variáveis do workspace

```
>>clear A         %deleta a matriz A
>>clear all       %deleta todas as variáveis do workspace
```

`inf or Inf`: expressa infinito
```
>>infinito = 1/0
```

`NaN`: is Not a Number
```
>>e=0/0
>>f=inf/inf
>>g=tan(inf)
```

`eps`: EPS é o nível de tolerância usada por várias funções do `Matlab`
```
>>eps              %mostra o valor default de eps (2.2204e-016)
>>eps = 1E^-5      %altera o valor de eps para 10^-15
```

`pi: 3,1415...` %valor de π
```
>>cos(2*Pi/3)      %calcula o co-seno de 120°
```

1.5. Formatos de apresentação

O Matlab possui diferentes formatos de visualização do valor das variáveis. Por exemplo, se a matriz contém apenas inteiros, todos os termos da matriz são apresentados com valores inteiros

```
A=[1 3 4; 7 3 70]
```

Se algum termo da matriz não pode ser representado como um inteiro, então toda a matriz é apresentada de modo conhecido como `format short`. Tal apresentação mostra quatro dígitos depois do ponto decimal, a exceção do zero.

```
A = [2 3.1234 1/4; 7.654321 0 pi]
```

Para ver mais de quatro dígitos, mude o formato de apresentação.
```
format long
format long e
format short e
format short
```

1.6. Operadores relacionais

```
<      % menor que
<=     % menor ou igual a
>      % maior que
>=     % maior ou igual a
==     % igualdade de comparação
~=     % diferente
(=     % igualdade de atribuição)
```

```
>>A=[4 2 1; 5 3 -1; 0 50 2]
>>X=A>3      %X=[1 0 0;1 0 0;0 1 0] indica onde estão os valores de A maiores que 3
```

1.7. Operadores lógicos

```
&          %e
|          %ou
~          %não
```

1.8. Funções elementares do Matlab

```
>>clear all
>>A=[-0.1, 0.1, 0; inf 2 10; -pi -5 9.78]  %inserir esta matriz

abs(X):                    %valor absoluto (positivo ou nulo)
>>abs(A)

cos(X):                    %co-seno de um ângulo expresso em radianos
sin(X):                    %seno de um ângulo expresso em radianos
>>cos(A)   >>sin(A)

log(X):                    %logaritmo natural
>>log(A)   >>log(abs(A))

log10(X):                  %logaritmo na base 10
>>log10(A)

exp(X):                    %função exponencial
>>exp(-1)  >>exp(A)

ceil(X):                   %arredonda para cima
>>ceil(A)

floor(X):                  %arredonda para baixo
>>floor(A)

round(X):                  %arredonda para o inteiro mais próximo
>>round(A)

polyval(P(X),b):           %Avalia o polinômio P(X) no ponto b
>>polyval([1 5 4],-1)

roots(P(X)):               %informa as raízes do polinômio P(x)
>>roots([1 5 4])

sqrt(x):                   %raiz quadrada de números positivos e negativo)
>>sqrt(A)
```

1.9. Operações envolvendo matrizes e vetores

```
clear('all')
>>A=[1 2 3]               %vetor linha (separado por espaços ou vírgulas)
>>A=[1;2;3]               %vetor coluna (separado por ponto e vírgula)

>>A=[1 2], B = [3 4]
>>C=[A B]                 %concatenação de vetores – forma vetor linha
>>C=[A;B]                 %formação da matriz 2x2

>>A=[1;2], B = [3;4]
>>C=[A B]                 %formação de matrizes - 1ª col. vetor A  e  2ª col.
   vetor B
```

```
>>C=[A;B]                %concatenação de vetores - forma vetor coluna

>>clear all
>>A=1:2:10               %A = [1 3 5 7 9]
>>A=1:10                 %B = [1 2 3 4 5 6 7 8 9 10]
>>A=1:-0.1:0             %C = [1  0.9  0.8  ...  0.1  0]
>>A=linspace(1,0,4)      %vetor com início em 1 e fim em 0 contendo 4
  elementos

>>A=1:3:11               %A = [1 4 7 10]
>>A(2)                   %ans = 4
>>A(2:3)                 %ans = [4 7]
>>A([2 4])               %ans = [4 10]
>>A + 1                  %ans = [2 5 8 11]

>>clear all
>>A=[1:3; 4:6]           %A = [1 2 3; 4 5 6]
>>A(1,2)                 %ans=2
>>A(1,:)                 %primeira linha: ans=[1 2 3]
>>A(:,2)                 %segunda coluna: ans=[2 5]

>>B = [-1 -2 -3]
>>B(:) + A(2,:)          %conta errada
>>B + A(2,:)             %conta correta
>>B(1,:) + A(2,:)        %conta correta

>>length(A(2,2:3))       %comprimento de um vetor  ans=2
>>size(A)                %números de linhas e colunas da matriz A, ans =
  [2 3]
>>[lin col] = size(A)
>>sum([1 5 4])           %soma os elementos de um vetor
>>sum(A)                 %soma as colunas da matriz A (resulta num vetor)
>>max(A(2,:))            %retorna o elemento máximo de um vetor
>>max(A)                 %retorna o máximo de cada coluna de A (resulta
  num vetor)
>>min(A(:,2))           %retorna o elemento mínimo de um vetor
>>[vet val]=eig(A)       %autovetores(vet) e autovalores(val) de A
>>inv(A)                 %cálculo da inversa de A
>>[Lin Col]=find(A>4)    %mostra as coordenadas dos elementos maiores
  que 4

>>A=[1 2 3 4 5 6]
>>B=[7 8 9 10 11 12]
>>A>3                    %ans = [0 0 0 1 1 1]
>>A([0 0 1 0 0 1])       %ans = [3 6]
>>X=A(A>3)               %Atribui a X os coeficientes de A que são maiores
  que 3
>>X=B(A>2 & A<5)%A>&A<5 = [0 0 1 1 0 0] X=[9 10]

>>A=[pi sqrt(3) 8; -1 0 4/9]
>>B=[3 2, 1/3; 2 3 4]
>>C=A.*B                 %multiplicação termo a termo (A e B de igual
  dimensão)
```

```
>>C=B./A              %divisão termo a termo
>>C=A.^B             %potenciação termo a termo
>>C=A.^2
>>C=2.^A

>>clear all
>>A = [1 0 2; 4 7 2; -1 0 1]      %A deve ser uma matriz quadrada
>>p = 3               %número qualquer
>>C=A^p               %eleva a matriz A à potência p
>>C=A^0.3             %utiliza autovalores e autovetores

>>clear all
>>A=[1 1; 1 -1;
>>B=[1 2]^T
>>X=A\B'              %resolve AX = B (A é quadrada não singular)
  (inv(A)*B)
>>X=A'/B             %resolve XA = B (B pode ser uma matriz)
  (B'*inv(A))
```

1.10. Matrizes especiais

```
>>A = [1:3; 4:6]
>>A`            %transposta da matriz A
>>eye(3)        %matriz identidade 3x3
>>diag([6 3 1]) %matriz 3x3 cuja diagonal é 6 3 1
>>rand(3)       %matriz 3x3 formada por elementos aleatórios
>>zeros(3,2)    %matriz 3x2 formada por zeros
>>zeros(2)
>>ones(2,1)     %matriz 2x1 formada por uns
>>A=[A ones(2)]
>>B(:,3)=ones(n,1)
>>A=[]          %matriz vazia, sem dimensão
>>A = 1:0.5

>>A=[3 2 0; 5 -3 1; -2 0 7]
>>A(:,2)=[]                %elimina a coluna 2 da matriz
>>A([1 3],:)               %elimina as linhas 1 e 2
>>flipud(A)                %troca as linhas de cima para baixo
>>A=flipud(A([1 2],:))     %troca somente as linhas 1 e 2
>>A=[A(2,:); A([1 3],:)]   %troca somente as linhas 1 e 2
>>fliplr(A)                %troca as colunas da esquerda à direita
>>A=fliplr(A(:,[1 2])      %troca somente as colunas 1 e 2
>>A=[A(:,[2 1]), A(:,3)]   %troca somente as colunas 1 e 2
>>sparse
```

2. Comandos de repetição e de condição

2.1. for

```
for j=1:n
    comandos
end
```

Exemplo (1)

```
>>for i=1:3, x(i) = -i;, end, disp(x) %gera o vetor x = [-1 -2 -3]

>>for i=1:4...
>>   for j=1:5...
>>       A(i,j) = 1/(i+j-1); ...
>>   end ...
>>end, A
```

2.2. while

```
while expressão
      comandos
end
```

Exemplo (2)

```
>>x =10;
>>n = 0;
>>while x>1e-100,
>>   x=x/2;
>>   n = n+1;
>>end, x, n
```

2.3. if, else, elseif

```
if expressão 1
   conjunto de comandos 1
elseif expressão 2
   conjunto de comandos 2
...
else
   conjunto de comandos
end
```

Exemplo (3)

```
>>%Problema da teoria dos números (Fonte: Matlab, User's
  Guide)
>>While 1
>>n = input('Digite um número inteiro, (se for negativo o
            programa será abortado.   ');
>>   if n <= 0, disp('O programa foi abortado!!'); break;
     end
>>         while n > 1
>>              if rem(n,2) == 0
>>                   n=n/2
>>              else
>>                   n=3*n+1
>>              end
>>         end
>>end
```

3. Arquivos de programas (*M-files*)

No Matlab, quando é dado um comando, este comando é processado e imediatamente o resultado é mostrado na tela. Matlab também pode executar seqüências de comandos armazenados em arquivos cuja extensão é ".m" (daí a denominação de M-*files*).

Um M-*file* consiste de uma seqüência de comandos do Matlab que possivelmente incluem referências a outros M-*files*. Um M-*file* pode-se chamar a si mesmo (recursividade ou auto referência) e podem ser criados em qualquer editor de texto.

Pode-se ter dois tipos de M-files: os *scripts* e as funções. Ambos são do tipo ASCII, porém os *scripts* contém uma longa seqüência de comandos do Matlab, enquanto que as funções são criadas para aumentar a lista de funções do Matlab (geralmente para resolver problemas específicos).

3.1. Arquivo *Script*

Quando um arquivo *script* é chamado, Matlab simplesmente executa os comandos contidos nele. As variáveis geradas permanecem na área de trabalho (*workspace*). Por exemplo, vamos gerar um arquivo *script* que contem os comandos do exemplo anterior.

Para iniciar, digite na área de trabalho do Matlab

Exemplo (4)

```
>>edit exemplo4.m
```

Aguarde o editor do DOS (ou outro) abrir o arquivo exe01.m e depois digite a seguinte seqüência de comandos.

```
clc;       %para limpar a tela
disp('Problema da teoria dos números-Matlab, User's Guide)');
While 1
   m = 0;  %zerar o contador do número de operações
   n = input('Digite um número inteiro positivo: ');
   disp('O numero digitado foi');
   disp(n);
   if n <= 0, break; end
   while n > 1
      m = m + 1;
      if rem(n,2) == 0
           n=n/2;
      else
           n=3*n+1;
      end
   end
   disp('A quantidade de operacoes foi');
   disp(m);
end
```

Após digitada a seqüência, feche o editor do DOS. O arquivo exe01.m foi salvo no diretório c:/matlab/nome, onde nome é o sub-diretório do Matlab em que o usuário está no momento. Na área de trabalho do Matlab digite o comando

```
>>exe01
```

Esta instrução faz com que o `Matlab` execute todos os comandos contidos no M-*file* `exe01.m`. Após o término da execução, digite

```
>>whos
>>clear all
>>whos
```

3.2. Arquivo Função

Um *M-file* que contem no início da primeira linha a palavra `function`, é denominado de arquivo função. A função difere do *script* em relação aos argumentos (parâmetros) que são passados. Por exemplo seja a seguinte função, que calcula o fatorial de números inteiros positivos.

Exemplo (5)

```
>>!edit exemplo5.m

function y = fatorial(x)

%FATORIAL Fatorial de números inteiros positivos
%Para constantes, FATORA retorna o fatorial de x
%Para vetores, FATORA retorna um vetor contendo as fatoriais
%dos elementos de x
%Para matrizes, FATORIAL retorna uma matriz contendo as
%fatoriais dos elementos de x

[m,n] = size(x)
if (m*n) == 1,
 if (x < 0) | ((ceil(x) - x) ~= 0),
      y = NaN;
 elseif (x == 0),
      y = 0;
 else,
      y = prod(x:-1:1);
 end;
else,
 for i = 1:m,
      for j = 1:n,
           y(i,j) = fatora(x(i,j);
      end;
 end;
end;
```

Alguns detalhes na geração de um M-*file* função

- A primeira linha declara o nome da função, o argumento de entrada, e o argumento de saída. Sem esta linha, o M-file é um arquivo script ao invés de um arquivo função.
- símbolo % indica que o restante da linha é um comentário e será ignorado.
- O primeiro conjunto de linhas, precedidas de %, serão mostradas quando for digitado `help fatora`.

- A primeira linha, precedida de %, é conhecida como "linha H1", e contem palavras chaves a respeito da função criada. Esta linha aparece quando for usado o comando `lookfor fatorial`, por exemplo.
- As variáveis m e n são locais e não existem na área de trabalho do Matlab após fatora ter sido executado.
- A execução de um M-*file* pode ser interrompida pelo usuário digitando `Ctrl+C` na área de trabalho. Se o usuário incluir o comando `return` no M-*file*, então Matlab pára (aborta) a execução dos comandos contidos no M-*file*.
- Utilizando o comando pause num M-*file*, `Matlab` pára a execução dos comandos contidos no M-*file* e aguarda o usuário pressionar qualquer tecla.
- O emprego do comando `disp()` é útil para encontrar erros no programa.

3.3. Exemplos

Exemplo (6)

Implementar a inversão de uma matriz quadrada `m×m`.

```
function y = inversa(x);

%INVERSA calcula a inversa de uma matriz quadrada
%        O metodo de calculo eh o do escalonamento

[m,n]=size(x);

if (m ~= n),
      disp('A matriz deve ser quadrada. O programa foi abortado!');
      break;
else,
      a = [x eye(m)];
      pivot = 1:m;
      for i=1:m,
            [lin, col] = find(abs(a(i:m,i))>1e-15); %primeiro nao nulo
            if lin == [],
                  disp('A matriz nao eh inversivel');
                  disp('O programa foi abortado!');
                  return;
            elseif (i-1+min(lin) ~= pivot(i)),   %o pivot eh nulo
                  aux1 = a(i,:);
                  aux2 = a(min(lin),:);
                  a(i,:) = aux2;
                  a(min(lin),:) = aux1;
            end
            a(i,:)= a(i,:)/a(i,i);
            for j = i+1:m
                  a(j,i:2*m) = a(j,i:2*m) - a(j,i)*a(i,i:2*m);
            end
            for j = 1:i-1
                  a(j,i:2*m) = a(j,i:2*m) - a(j,i)*a(i,i:2*m);
            end
      end
end

y = a(1:m,m+1:2*m);
```

Exemplo (7)

Implementar o método simplex.

```
function [primal, dual, fo] = simplex(c, a, b, tol)

%SIMPLEX resolve problemas de programacao linear
%        Os problemas sao de maximizacao com restricoes do tipo
%        menor ou igual.
%        Nao pode-se usar restricoes de igualdade e o vetor b deve
%        ser positivo.

if nargin < 4, tol = 1e-15; end;

[lin_c, col_c] = size(c);
[lin_a, col_a] = size(a);
[lin_b, col_b] = size(b);

if (lin_c > 1),
   disp('O programa serah abortado!! A funcao objetivo');
   disp('deve ser expressa por um vetor LINHA.');
   return;
end

if (col_a ~= col_c),
    disp('O programa serah abortado!! O numero de elementos do
vetor
    disp('funcao objetivo deve ser igual ao numero de colunas da');
    disp('matriz custo (A).');
    return;
end
if (lin_b ~= lin_a),
    disp('O programa serah abortado!! O numero de elementos do');
    disp('vetor COLUNA b (rhs) deve ser igual ao numero de
          linhas');
    disp('da matriz custo (A).');
    return;
end

C = [c zeros(1,lin_a) 0];
A = [a eye(lin_a) b];
continua = 1;
iteracao = 0;
base = col_a+1:col_a+lin_a;
while (continua == 1) & (iteracao < 51),
      iteracao = iteracao + 1;
      entra = max(C)
      if (max(C) >= tol),
            sai1 = find(A(:,entra)>tol);
            if (isempty(sai1) == 0),
                  menor = b(sai1)./A(sai1,entra);
                  sai2 = min(b(sai1)./A(sai1,entra));
                  sai = find(menor==sai2)
            else,
                  continua == 0;
                  disp('O problema eh ilimitado!!');
                  break;
            end;
            base(sai) = entra;
            A(sai,:) = A(sai,:)/A(sai,entra);
```

```
            zerar = find(abs(A(:,entra))>tol);
            for i = 1:sai-1,
                  A(i,:) = A(i,:) - A(sai,:)*A(i,entra);
            end
            for i = sai+1:lin_a,
                  A(i,:) = A(i,:) - A(sai,:)*A(i,entra);
            end
            C(1,:) = C(1,:) - A(sai,:)*C(1,entra);
      else,
            continua = 0;
            disp('A solucao encontrada eh otima!!');
            basees = base
            primal = A(:,col_a+lin_a+1)
            dual   = C(1+col_c:col_c+lin_a)'
            fo     = -C(col_c+lin_a+1)
      end
end
```

Exemplo (8)

Para evitar erros de digitação de seqüências de números de importância fundamental, como a matrícula de um aluno, o CPF, o número da conta bancária, geralmente se adiciona ao número um *dígito verificador*. Por exemplo, o número de matrícula 811057 é usado como 8110573, onde 3 é o dígito verificador calculado da seguinte maneira:

a) cada algarismo do número é multiplicado por um peso começando de 2 e crescendo de 1 unidade da direita para a esquerda:(8×7, 1×6, 1×5, 0×4, 5×3, 7×2, 3×2;
b) somam-se as parcelas obtidas: 56 + 6 + 5 + 0 + 15 + 14 = 96;
c) obtém-se o resto da divisão desta soma por 11: 96 ÷ 11 = 8×11 + 8 (o resto é 8);
d) subtrai-se de 11 o resto obtido: 11 – 8 = 3;
e) se o valor encontrado for 10 ou 11, o dígito verificador será 0; nos outros casos o dígito verificador é o próprio valor encontrado

Pede-se para escrever uma função, cujo parâmetro de entrada é o número de registro da matrícula (incluindo o dígito), que verifica se o dígito informado está correto ou não.

Exemplo (9)

Em um certo município, vários proprietários de imóveis estão em atraso com o pagamento do imposto predial. Implementar um programa de computador (em forma de função) que calcula e informa na tela do computador o valor da multa a ser paga pelo proprietário considerando que:

- o número de meses em atraso e o valor do imposto serão informados pelo usuário do programa;
- as multas devem ser calculadas a partir do valor do imposto e de acordo com a seguinte tabela:

Valor do imposto	% por mês em atraso
até R$ 10,00	1%
de R$ 10,01 a Cr$ 35,00	2%
de R$ 35,01 a R$ 110,00	4%
de R$ 110,01 a R$ 250,00	7%
Acima de R$ 250,00	10%

Deverá ser impresso na tela (i) o valor do imposto sem a multa, (ii) a quantidade de meses em atraso; (iii) o valor da multa; (iv) o total a ser pago pelo proprietário.

Exemplo (10)

Implementar uma função que classifica um NÚMERO inteiro positivo em PAR ou IMPAR. Se o NÚMERO é ímpar, também deverá ser informado na tela se é PRIMO ou não.

Exemplo (11)

Implementar a seqüência de Fibonacci

4. Ler, Salvar e Imprimir

Seja a seguinte matriz digitada no arquivo `matriz.txt`. Esta matriz deve ser digitada num arquivo utilizando o editor do DOS, o Excel, ou o WordPad, e deve ser gravado na área de trabalho do usuário.

```
 9   13   24   44
46   46   24   35
45   72    5   32
36   90    8   37
16   28   65   40
68   26   20   60
70   87   85   12
73   24   18    4
48   81   18   46
56   91  100   87
>>matriz = random('Discrete Uniform',100,10,4))
```

4.1. Salvar e carregar matrizes armazenadas em arquivos -ascii

```
save A.txt A -ASCII          %salva no arquivo A.txt a variável A
load A.txt                   %carrega dados contidos em arquivos
     >> A = random('unif',0,1,[3,4])  %gerar dados aleatórios
     >> save a.txt a - ascii          %salva num arquivo
     >> load a.txt           %carrega a.txt e cria a matriz a
     >> AB=load('a.txt')     %carrega a.txt e cria a matriz AB
```

4.2. Abrir um arquivo para ler e/ou gravar

Para que `Matlab` possa ler dados em arquivos (e armazená-los no `workspace`) e/ou gravar em arquivos os resultados oriundos da execução de *script files* ou *m-functions*, é necessário antes executar o seguinte comando:

```
fid = fopen(arq,permite)  % FID: File Identifier
```

Este comando "abre" o arquivo `arq` permitindo a leitura ou a gravação. O argumento `arq` é um *string* (p.ex. `'matriz.txt'`) que contem o nome do arquivo a ser aberto. O argumento `permite` indica a operação (de leitura ou de gravação) que será executada no arquivo aberto. Alguns dos possíveis valores do argumento `permite` são:

```
'r'          % para ler
'w'          % para escrever (cria se necessário)
'a'          % para anexar (cria se necessário)
'r+'         % para ler e escrever (não cria)
'a+'         % para ler e adicionar (cria se necessário)
```

Se ocorrerem erros ao executar o comando, `fopen` não abre o arquivo especificado no argumento arq, e retorna `FID = -1`. Se o comando for executado com sucesso, então FID assumirá um número inteiro maior ou igual a 3.

```
>>arq = fopen('matriz.txt','r')      %abre o arquivo matriz.txt
>>arq = fopen('result.txt','w')      %tenta abrir result.txt
```

4.3. Ler dados de um arquivo

Para ler, e carregar para o `workspace`, dados armazenados num arquivo do tipo –ascii, utiliza-se o comando`fscanf`, e se o arquivo for binário, emprega-se o comando `fread`. Suponhamos que o comando `fopen` já tenha sido usado, ou seja, o usuário digitou `>>fid = fopen('NomeArquivo','r')`. O comando de leitura para carregar os dados contidos no arquivo `NomeArquivo` (tipo –ascii) é:

```
[matriz quantos] = fscanf(fid, formato, tamanho)
```

`matriz`: é a variável na qual serão armazenados os dados lidos
`quantos`: é a quantidade de dados lidos
`formato`: é um *string* que contem o caracter %, tamanho do campo a ser lido e algum caracter do tipo d, i, o, u, x, e, f, g, s, c, dependendo dos dados a serem lidos. Por exemplo `'%s'`, `'%d'` e `'%f'`.
`tamanho`: é opcional e limita a quantidade de elementos que podem ser lidos do arquivo. Se não for especificado, o arquivo inteiro será ser lido. O argumento `tamanho` pode assumir os valores

- `N`: lê pelo menos N elementos num vetor coluna
- `Inf`: lê tudo, até o final do arquivo
- `[M,N]`: e uma matriz `M×N`, preenchendo pelo menos uma matriz `M` colunas por `N` linhas. `N` pode ser `inf` mas `M` não.

Se o uso do `fid` for desnecessário, deve-se utilizar o comando `fclose(fid)`. Se este comando não for utilizado o arquivo `fid` continuará disponível, ocupando desnecessariamente memória no `workspace`.

Exemplo: Vamos inicialmente criar a matriz `exp.txt` e salvá-la num arquivo –ascii.

```
>> a = 0:0.1:1, b = exp(a), exp=[a' b']
exp= 0.00   1.00000000000000
     0.10   1.10517091807565
     0.20   1.22140275816017
     0.30   1.34985880757600
     0.40   1.49182469764127
     0.50   1.64872127070013
     0.60   1.82211880039051
     0.70   2.01375270747048
     0.80   2.22554092849247
     0.90   2.45960311115695
     1.00   2.71828182845905
```

```
>>save exp.txt exp -ascii   %cria o arquivo exp.txt
>>fid = fopen('exp.txt')    %abre o arquivo exp.txt
>>a = fscanf(fid,'%g %g',[2 inf]) %A matriz gravada possui duas
                        colunas. A quantidade de linhas é dado como inf.
>>a = a'                          %transposta
>>a0 = fscanf(fid,'%g',[2 inf])  %leitura correta
>>a1 = fscanf(fid,'%4g',[2 11])  %leitura absurda
>>a2 = fscanf(fid,'%4g')         %transforma num vetor coluna
>>a3 = fscanf(fid,'%f',[2,inf])  %lê corretamente
>>a3 = fscanf(fid,'%f4',[2,inf]) %não lê corretamente
>>fclose(fid)     %fecha o arquivo aberto

>>fid = fopen('matriz.txt') %matriz de números inteiros
>>R0 = fscanf(fid,'%f')     %lê números reais como inteiros
>>R1 = fscanf(fid,'%4f')    %lê erradamente

>>S = fscanf(fid,'%s')      %lê (e retorna) um string
```

`[matriz, quantos] = fread(fid,tamanho,precisao)`: lê dados binários

4.4. Gravar dados num arquivo

O dois principais comando utilizados para gravar resultados em arquivos são os comandos `fprintf` e `fwrite`. O comando `fwrite` grava dados em formato binário. A sintaxe do comando `fprintf` é:

`fprintf(fid,format,A,...)`
Se `fid` for ignorado, então o resultado aparecerá na tela.

Exemplo

```
>>a = exp(random('unif',0,1,[2,2]))
>>a(1,:) = a(1,:)*1000
>>a(2,:) = a(2,:)/1000
>>format long
>>fprintf('%f', a)       %gera um vetor sem espaços
>>fprintf('%f ', a)      %gera um vetor separando os elementos
>>fprintf('%4.0f ', a) %imprime a parte antes da vírgula
>>fprintf('%3.2f ', a) %imprime os números com 2 dígitos
>>fprintf('A = %8.4f ', a) %reserva 8 posições para escrever o número,
   imprimindo 4 dígitos após a vírgula

>>for i=1:2, fprintf('a(%d,%d) = %9.4f, a(%d,%d) = %9.4f
  \n',i,1,a(i,1),i,2,a(i,2)), end
>>for i=1:2, fprintf('a(%f,%d) = %9.4f, a(%g,%d) = %9.4f
  \n',i,1,a(i,1),i,2,a(i,2)), end


>>fid = fopen('a.txt', 'w')
>>for i=1:2, fprintf(fid,'a(%d,%d) = %9.4f, a(%d,%d) = %9.4f
  \n',i,1,a(i,1),i,2,a(i,2)), end
>>type a.txt          %não funciona
>>fclose('all')       %fechar os arquivos abertos
>>type a.txt          %mostra o conteúdo do arquivo a.txt
```

```
>>fid = fopen('a.txt', 'w')
>>for i=1:2, fprintf(fid,'\nLinha %d',i);
  fprintf(fid,'%15.5f',a(i,1:2)), end
 ou
>>for i=1:2, fprintf(fid,'Linha %d',i);
  fprintf(fid,'%15.5f',a(i,1:2)); fprintf(fid,'\n'), end
 ou
>>for i=1:2, fprintf(fid,'Linha %d',i);
  fprintf(fid,'%15.5f',a(i,1:2)); fprintf(fid,'\n\n'), end
>>fclose('all')        %fechar os arquivos abertos
>>type a.txt           %mostra o conteúdo do arquivo a.txt
```

Resumo referente ao comando `'%a.bf'`

⇒ O argumento `a` é a quantidade de casas reservadas para escrever o número;

⇒ O argumento `b` é a quantidade de dígitos impressos após a vírgula;

⇒ Se o número que deverá ser escrito contiver mais dígitos válidos antes da vírgula do que a quantidade `a`, todos eles serão impressos, inclusive os `b` dígitos.

⇒ Se o usuário conhecer bem as quantias a serem impressas, poderá alinhá-las através de um controle correto sobre o valor de `a`.

5. Gráficos 2-D

Comando plot

`plot(x,y,s)` onde `s` é um *string* de 1, 2 ou 3 dos seguintes caracteres

`y`	yellow	`.`	point
`m`	magenta	`o`	circle
`c`	cyan	`x`	x-mark
`r`	red	`+`	plus
`g`	green	`-`	solid
`b`	blue	`*`	star
`w`	white	`:`	dotted
`k`	black	`-.`	dashdot
		`--`	dashed

```
>>x = linspace(0,2*pi,100) %plotar a função seno
>>y = sin(x);
>>plot(x,y)

>>fplot('sin(x)',[0, 2*pi])

>>fplot('sin(x)/x',[-50*pi,50*pi])

>>zoom on
>>zoom off
>>zoom out

>>fplot('-10*x^3+20*x^2+10*x',[-1 3 -5 30])

>>fplot('sin(x)/x',[-50*pi,50*pi])
>>grid on
>>zoom on          %qual o 1° vlr de x positivo ou negativo q/ anula a função?
>>grid off
```

5.1. Estilos de linha do gráfico

```
>>x = linspace(0,2*pi,100);
>>y = tan(x);
>>plot(x,y)
>>plot(x,y,':')
>>plot(x,y,'--')
>>plot(x,y,'.-')
>>plot(x,y,'o')
```

5.2. Diferentes cores do gráfico

```
>>plot(x,y,'r') %vermelho
>>plot(x,y,'rx')
>>fplot('sin(x)*x^3',[0,4*pi],'-.b')
```

5.3. Múltiplos gráficos numa única figura

```
>>x = linspace(0,2*pi,100);
>>y = tan(x);
>>z = cos(x);
>>plot(x,y,x,z);
>>plot(x,y,'r--',x,z,'b:o');

>>fplot('[sin(x),cos(x)]' [0,4*pi],'-.b')
>>fplot('[tan(x),sin(x),cos(x)]',2*pi*[-1 1 -1 1])
>>fplot('[tan(x),sin(x)]',2*pi*[-1,1,-1,1],'o',':') %:
>>fplot('[tan(x),x]',2*pi*[-1,1,-1,1],'bo','r--') % r–

>>fplot('tan(x)',2*pi*[-1,1,-1,1],'r--')
>>hold on
>>fplot('sin(x)',2*pi*[-1,1,-1,1],'b:')
>>grid on, zoom on, hold off
```

5.4. Inserção de legendas

```
>>xlabel('x');
>>ylabel('y');
>>title('Seno e Co-seno');
>>legend('seno','co-seno');
>>text(1.3, 4.2, 'AQUI');
>>gtext('MOUSE');
```

5.5. Eixos

```
axis([Xmin Xmax Ymin Ymax])
>>plot(x,y,'r-o',x,z,'b:.');
>>grid on
>>zoom on
>>axis([0 2*pi -10 10])
>>axis([0 2*pi -1 1])
```

5.6. Sub figuras numa única figura

```
>>subplot(2,1,1)
>>fplot('x.^2',[0 4],´y*´);
>>subplot(2,1,2)
>>fplot('-x.^2',[0 4],´g--´);
>>grid on
```

Limpar figura
```
>>clf;
```
Gráficos especiais

5.7. Gráficos especiais

```
>>x = linspace(0,10*pi,1000);
>>plot(x,z)
>>semilogx(x,z)  %log na base 10 – Erro!! contem dados negativos
>>semilogy(x,z)
>>polar(pi/6,3, 'ro')
>>polar([0:0.5:10*pi/2],[0:0.5:10*pi/2],'-')
>>polar([0:0.5:10*pi/2],[0:0.5:10*pi/2],'-')
bar
```

Exemplo (9)

Criar o arquivo exemplo9.m

```
x = [0:0.1:2*pi];
y = cos(x);
plot(x,y);
xlabel('x');
ylabel('y');
title('Curva Co-seno');
hold on;
pause;
z = sin(x);
plot(x,z);
grid on
hold off
```

Exemplo (10)

Vamos criar três arquivos. Um arquivo contem os dados, denominado de `dados.dat`, o segundo contem uma função denominado de `f.m` e o terceiro é o programa principal (`plot1.m`)

```
>>edit

[1 -1; 2 0; 3 1] %digite no arquivo dados.dat duas colunas e três linhas

function y = f(x)
y = exp(-x).*sin(x); %o arquivo f.m terá apenas duas linhas

load dados.dat   %gera a matriz dados
xx = dados(:,1);
yy = dados(:,2);
plot(xx,yy,'c-.');     %plota tt versus yy
disp('Pressione enter.');
pause;
clf;             %limpa o conteúdo da figura criada
t = [0:0.5:5];
y = f(t)
```

```
plot(t,y,'*-');
hold on;
plot(xx,yy,'c-');
hold off;
xlabel('Tempo');
ylabel('Valor a mostrar');
title('Tempo versus Valor');
legend('f(t)','dados.dat',0);
```

A legenda pode ser carregada (usando o mouse) para qualquer nova posição.

Exemplo (11)

%Fonte: Le Huy, H.: Introduction à Matlab et Simulink, Université Laval, Québec, Canada, 1988, pág. 14

```
t = 0:0.01e-3:0.06;
y = 10*exp(-60*t).*cos(120*pi*t);
z = 10*exp(-60*t).*sin(120*pi*t);
plot(t,y,'r',t,z,'g'); grid;
a = 10*exp(-60*t); hold;
plot(t,a,'b--'); plot(t,-a,'b--');
title('Função Senoidal');
xlabel('Temps , s'); ylabel('Tensao , V');
pause;
hold off
plot(y,z); grid;
axis('equal');
xlabel('y'); ylabel('z');
```

Exemplo (12)

%Fonte: Le Huy, H.: Introduction à Matlab et Simulink, Université Laval, Québec, Canada, 1988, pág. 15

```
w = logsace(0,3,1000);
s = j*w;
H = 225./(s.*s+3*s+225);
AdB=20*log10(abs(H));
fase=angle(H)*(180/pi);
subplot(2,1,1); semilogx(w,AdB); grid;
xlabel('w , rad/s'); ylable('Amplitude , dB');
subplot(2,1,2); semilogx(w,fase); grid;
xlabel('w , rad/s'); ylable('Fase , degre');
```

6. Gráficos 3-D

Matlab apresenta várias funções para plotar dados tridimensionais. Algumas plotam funções em três dimensões, enquanto que outros plotam superfícies. Alguns dos comandos são:

`plot3`	- plota linhas e pontos em 3D
`contour`, `contour3`	- cria contornos
`mesh`, `meshc`, `meshz`	- definição de superfícies a partir de malhas
`surf`, `surfc`, `surfl`	-
`fill3`	- cria polígonos tridimensionais preenchendo-os

6.1. Desenho de linhas

```
plot3(x,y,z)                    %x, y, z são matrizes de mesma dimensão
>>t = 0:pi/50:10*pi;
>>plot3(sin(t),cos(t),t);       %produz uma hélice

>>t=0:0.05:25;
>>x = exp(-0.05*t).*cos(t);
>>x = exp(-0.05*t).*sin(t);
>>z=t;
>>plot3(x,y,z), grid
```

6.2. Gráfico a partir de uma malha

Matlab define uma superfície a partir de uma malha. Primeiro deve-se ter dois vetores (x e y) de igual dimensão e que definem o domínio da função $z = f(x,y)$; em seguida usando o comando meshgrid gera-se duas matrizes X e Y cujos elementos correspondentes são todas as combinações possíveis (formando pares) dos elementos de x e de y. Formando os pares e ligando-os adequadamente, gerando uma malha no plano xy. A seguir aplica-se a função f em X e Y (nas interseções da malha) e determina-se z, que é o valor da função nos pares formados pelas matrizes X e Y. A variável z determina a altura do ponto a ser grafado. A função mesh plota os pontos z e une-os z conforme a malha gerada no plano xy, formando uma malha no espaço e que representa a função desejada.

```
>>x=-10:0.5:10;
>>y=x;
>>[X,Y]=meshgrid(x,y);          %cria a matriz que forma a malha no eixo xy
>>z=sqrt(X.^2 + Y.^2);
>>mesh(z);                      %gera um parabolóide
>>rotate3d on                   % pode rotacionar o gráfico

>>x=-8:0.5:8;
>>y=x;
>>[X,Y]=meshgrid(x,y);          %forma a matriz que gera a malha no eixo xy
>>z=sqrt(X.^2+Y.^2)+eps;
>>mesh(z);                      %gera o famoso sombrero
```

Criar o arquivo *script* surfit.m e digitar a seguinte seqüência de comandos

```
t=0:0.25:5;
```

Criar o arquivo *script* exemplo13.m e digitar a seguinte seqüência de comandos

```
b=1200*pi;
dt=50e-6;
for j=1:15
for i=1:150
      k(j)=j;
      a=(16-j)*50;
            t(i)=(i-1)*dt;
            y(j,i)=exp(-a*t(i).*sin(b*t(i));
      end
end
[K,T]=meshgrid(k,t);
mesh(T,K,y)
t=0:0.25:5;
a=0:0.5:10;
[T,A]=meshgrid(t,a);
Y=exp(T).*sin(T*A);
```

```
surf(T,A,Y)
xlabel('T')
ylabel('A')
zlabel('Y')
title('Y sobre T e A')
colormap(gray)
colorbar vert
rotate3d on
```

Executar o arquivo exemplo13.m.

```
>>shading interp
```

6.3. Desenho de contornos

Matlab pode traçar contornos tanto no espaço bidimensional como no tridimensional. Os comandos contorno e contorno3 geram plots compostos de linhas considerando valores constantes obtidos de uma matriz

```
>>x=-10:0.5:10;
>>y=x;
>>[X,Y]=meshgrid(x,y);        %malha no eixo xy
>>z=sqrt(X.^2 + Y.^2);
>>contour(z,2);               %contorno 2-D do parabolóide
>>contour3(z,5);              %contorno 3-D do parabolóide

>>x=-8:0.5:8;
>>y=x;
>>[X,Y]=meshgrid(x,y);        %matriz da malha no eixo xy
>>z=sqrt(X.^2+Y.^2)+eps;
>>contour(z,5);               %contorno 2-D do sombrero
>>contour3(z,10);             %contorno 3-D do sombrero
```

6.4. Pseudo gráficos através de cores

```
>>z=peaks;
>>pcolor(z)
>>colormap(hot)

>>colormap(hot);
>>pcolor(peaks)
>>shading flat                %elimina o grid (a malha)
>>hold on                     %mantem o gráfico ativo p/ plotar outro
>>contour(peaks, 20, 'k')     %´k´: linhas pretas para o contorno
>>hold off

>>A=zeros(32);
>>A(14:16,14:16)=ones(3);
>>Y=fft2(A);
>>surf(abs(Y));
>>shading interp;
>>title('Transformada discreta de Fourier de uma matriz');
```

6.5. Malhas e Superfícies

As funções `mesh` e surf mostram superfícies em três dimensões. Se *z* é uma matriz cujos elementos *z*(*i*,*j*) definem a altura da superfície acima de um grid (*i*,*j*), então `mesh(z)` gera uma malha no espaço. Similarmente `surf(z)` gera facetas coloridas, formando a superfície que representa a função *z*.

```
>>subplot(2,1,1);
>>mesh(peaks);
>>subplot(2,1,2);
>>surf(peaks);
```

Miscelânea

```
>>mesh(peaks(20)+7);
>>hold;
>>pcolor(peaks(20));
>>hiden off;
```

6.6. *Subplots*

```
>>t=0:pi/10:2*pi;
>>[X,Y,Z]=cylinder(4*cos(t));
>>subplot(2,2,1)
>>mesh(X)

>>subplot(2,2,2)
>>mesh(Y)

>>subplot(2,2,3)
>>mesh(Z)

>>subplot(2,2,4)
>>mesh(X,Y,X)
```

6.7. *Movies*

```
>>M = moviein(16);
>>for j = 1:16
>>    plot(fft(eye(j+16)))
>>    M(:,j) = getframe;
>>end
```

6.8. Mudando propriedades das superfícies

Abrir o arquivo surit.m e trocar surf(T,A,Y) por

```
h = surf(T,A,Y)
```

Execute surfit.m.

Digite na área do trabalho do Matlab

```
>>get(h)
>>get(h,'meshstyle')
>>set(h,'meshstyle')
>>set(h,'meshstyle','row')
>>set(h)
```

Abrir o arquivo surit.m e trocar h = surf(T,A,Y) por

```
h = mesh(T,A,Y)
set(h,'meshstyle','row')
```

Execute o programa surfit.m.

6.9. Mudando propriedades dos eixos

```
>>clear all
>>clf                    %limpa a figura atual
>>h1 =subplot(2,1,1)
>>plot1
>>h2 = subplot(2,1,2)
>>surfit
>>get(h1)                %mostra as propriedades do eixo xy da figura h1
>>get(h2)                % mostra as propriedades do eixo xy da figura h2
>>axes(h2)               %torna atual a figura h2
>>get(gca)               %mostra as propriedades do eixo atual (h2)
>>get(gca,'box')
>>set(gca,'box')
>>set(gca,'box','on')
```

6.10. Mudando as propriedades da figura

```
>>get(gcf)
>>get(gcf,'colormap')
>>set(gcf,'colormap',[0 0 0]
```

6.11. Mudando as propriedades da figura para impressão

```
>>get(gcf,'PaperUnits')
>>get(gcf,'PaperPosition')
```

O par (0.25 0.25) (em polegadas) é medido de baixo para cima da esquerda para a direita. A área de impressão terá 8 polegadas de largura e 6 polegadas de altura. Para trocar estas medidas use o seguinte comando

```
>>set(gcf,'PaperPosition',[0.25, 0.25, 6, 6])
>>print -deps myplots.eps        %salva o gráfico como PostScript
```

7. Funções de otimização

O *toolbox optimiza* contem algumas funções de otimização. Pode-se tanto determinar máximos e mínimos de funções como resolver problemas de programação linear e quadrática.

7.1. `attgoal`

Objetivo

Resolve problemas multi-objetivos, ou seja,

$$\underset{x,\lambda}{minimize}\ \lambda \qquad \text{sujeito a} \qquad F(x) - w\lambda \le objetivo$$

dados $F(x)$, w e *objetivo*, onde x, w, e *objetivo* são vetores, λ é uma variável escalar, e $F(x)$ é uma função que retorna um vetor de valores.

Comando
```
x = attgoal('fun',x0,goal,w,options,vlb,vub,'grad')
```
`fun`: `fun.m` é arquivo que contem as funções a serem minimizadas
`x0`: solução inicial
`goal`: vetor dos objetivos a serem alcançados
`w`: vetor de pesos
`options`: algumas opções (mudanças de parâmetros)
`vlb`: limite inferior de *x*
`vub`: limite superior de *x*
`grad`: usa o gradiente das funções gravados no arquivo `grad.m`

7.2. `conls`

Objetivo
Resolve o problema dos mínimos quadrados sujeito a um conjunto de restrições, ou seja,

$$\underset{x}{minimize}\frac{1}{2}\|Ax-b\|_2^2 \quad \text{sujeito a} \qquad C(x)\le d$$

onde *A* e *C* são matrizes e *b* , *d* e *x* são vetores.

Comando
```
[x, lambda, how] = conls(A,b,C,d,vlb,vub,x0,neqcstr,display)
```
`A,b`: matriz e vetor dos coeficientes do sistema linear $Ax = b$
`C,d`: matriz e vetor dos coeficientes das restrições $Cx \le d$
`vlb`: limite inferior das variáveis
`vub`: limite superior das variáveis
`x0`: solução inicial
`neqcstr`: as primeiras `neqcstr` restrições são de igualdade
`display`: mostra (`display=1`) ou não mostra (`display=-1`) avisos na tela
`lambda`: multiplicadores de Lagrange
`how`: indica erros ocorridos na última iteração

7.3. `constr`

Objetivo
Determina o mínimo de uma função (de várias variáveis) sujeito a um conjunto de restrições, ou seja,

$$\underset{x}{minimize}\, f(x) \text{ sujeito a} \qquad G(x)\le 0$$

onde *x* é um vetor, *G*(*x*) é uma função que retorna um vetor e *f*(*x*) é uma função que retorna um escalar. Ambas, *f*(*x*) e *G*(*x*) podem ser não linear. As restrições podem ser de igualdade.

Comando
```
[x,op ,lambda,hess]=constr('fun',x0,options1,vlb,vub,'grad')
```
`fun`: `fun.m` é arquivo que contem a função a ser minimizada e as restrições
`x0`: solução inicial
`options1`: algumas opções (mudanças de parâmetros)
`vlb`: limite inferior das variáveis
`vub`: limite superior das variáveis

`grad`: usa o gradiente das funções gravados no arquivo `grad.m`
`op`: retorna os parâmetros usados na minimização
`lambda`: retorna os multiplicadores de Lagrange
`hess`: retorna uma aproximação do hessiano na última iteração

Exemplo (14)

Achar x que minimiza $f(x) = -x_1x_2x_3$, iniciando pelo ponto x = [10 10 10], sujeito as restrições $-x_1 - 2x_2 - 2x_3 \le 0$ e $x_1 + 2x_2 + 2x_3 \le 72$.

7.4. `fmin`

Objetivo

Determina o mínimo de uma função (de uma variável) restrito a um intervalo fixo, ou seja,

$$\min_a imize\ f(a) \quad \text{sujeito a} \qquad a_1 < a < a_2$$

onde a, a_1 e a_2 são escalares e $f(a)$ é uma função que retorna um escalar.

Comando

```
a = fmin('fun',a1,a2,options)
```

`fun`: `fun.m` é arquivo que contem a função a ser minimizada
`a1,a2`: os limites inferior e superior tal que `a1<a<a2`
`options`: algumas opções (mudanças de parâmetros)

Exemplo (15)

```
a = fmin('sin',0,2*pi)
```

7.5. `fminu`, `fmins`

Objetivo

Determina o mínimo de uma função (de várias variáveis) sem restrições, ou seja,

$$\min_a imize\ f(x)$$

onde x é um vetor e $f(x)$ é uma função que retorna um escalar.

Comando

```
x = fminu('fun',x0,options,'grad')
```

`fun`: `fun.m` é o arquivo que contem a função a ser minimizada
`x0`: vetor que é o ponto inicial do processo de minimização
`options`: algumas opções (mudanças de parâmetros)
`grad`: usa o gradiente das funções gravados no arquivo `grad.m`

Exemplo (16)

Determine valores que minimizam $f(x,y) = 100\left(x - y^2\right)^2 + (1-x)^2$ iniciando no ponto (x,y) = [-1.2 1]. (Use um M-*file*)

7.6. `fsolve`

Objetivo

Resolve um sistema de equações não lineares, ou seja,

$$F(x) = 0$$

onde x é um vetor e $F(x)$ é uma função que retorna um vetor de valores.

Comando

```
x = fsolve ('fun',x0,options,'grad')
```

`fun`: `fun.m` é o arquivo que contem a função a ser minimizada
`x0`: vetor que é o ponto inicial do processo de minimização
`options`: algumas opções (mudanças de parâmetros)
`grad`: usa o gradiente das funções gravados no arquivo `grad.m`

Exemplo (17)

1) Determine um zero do sistema de duas equações e duas variáveis

$$2x - y = e^{-x}$$
$$-x + 2y = e^{-y}$$

partindo do ponto x0 = [-5 –5]. (Use um M-*file*)

2) Determinar uma matriz tal que

$X * X * X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ (Use um M-file e crie a função $F = x*x*x-[1\ 2;\ 3\ 4]$)

7.7. `fzero`

Objetivo

Determina o zero de uma função de uma variável, ou seja,

$$f(x) = 0$$

onde x é uma variável escalar e $f(x)$ é uma função que retorna um escalar.

Comando

```
x = fzero('fun',x0,tol,trace)
```

`fun`: `fun.m` é o arquivo que contem a função a ser minimizada
`x0`: vetor que é um ponto inicial ou intervalo
`tol`: precisão da soclução (o default é `eps`)
`trace`: mostra informações a cada iteração

Exemplo (18)

1) Determine o zero de seno mais próximo de 3

```
x = fzero('sin',3)
```

2) Determinar o zero de co-seno no intervalo [1 2]

```
x = fzero('cos',[1 2])
```
(pode ser calculado pois o sinal de cos(1) é diferente do sinal de cos(2))

7.8. `leastsq`

Objetivo

Resolve problemas de mínimos quadrados (não linear)

$$\min_x imize\ f(x) = f_1(x)^2 + f_2(x)^2 + f_3(x)^2 + \ldots + f_m(x)^2 + L$$

onde *L* é uma constante.

Comando
```
x = leastsq('fun',x0,options,'grad')
```
`fun`: `fun.m` é o arquivo que contem a função a ser minimizada
`x0`: vetor que é o ponto inicial do processo de minimização
`options`: algumas opções (mudanças de parâmetros)
`grad`: usa o gradiente das funções gravados no arquivo `grad.m`

Exemplo (19)

Achar *x* que minimiza $\sum_{k=1}^{10}\left(2+2k-e^{kx}-e^{ky}\right)^2$, iniciando no ponto *x* = [0.3 0.4]

7.9. `lp`

Objetivo
Resolve problemas de programação linear do tipo

$$\min_x imize\, cx^T \quad \text{sujeito a} \quad Ax \le b$$

onde *A* e *C* são matrizes e *b* , *d* e *x* são vetores.

Comando
```
[x, lambda, how] = lp(c,A,b,vlb,vub,x0,neqcstr,display)
```
`c`: vetor linha que contem dos coeficientes da função objetivo
`A,b`: matriz custo e o vetor *rhs* das restrições $Ax \le b$
`vlb`: limite inferior de *x*
`vub`: limite superior de *x*
`x0`: solução inicial
`neqcstr`: as primeiras `neqcstr` restrições são de igualdade
`display`: mostra (`display=1`) ou não mostra (`display=-1`) avisos na tela
`lambda`: multiplicadores de Lagrange
`how`: indica erros ocorridos na última iteração

Exemplo (20)
Determinar *x* que minimiza $f(x) = -5x - 4y - 6z$, sujeito ás restrições

$$y - y + z \le 20$$
$$3x + 2y + 4z \le 42$$
$$3x + 2y \le 30$$
$$x, y, z \ge 0$$

7.10. `minmax`

Objetivo
Resolve problemas de `minmax`, ou seja,

$$\min_x imize \max_{\{F_i\}} imize\{F_i\} \quad \text{sujeito a} \quad G(x) \le 0$$

onde *x* é um vetor e *F*(*x*) e *G*(*x*) são funções que retornam vetores de valores. $F_i(x)$ é o valor do *i*-ésima elemento do vetor retornado por *F*(*x*). *G*(*x*) pode conter restrições de igualdade.

, *w* e *objetivo*, onde *x*, *w*, e *objetivo* são vetores, λ é uma variável escalar, e *F*(*x*) é uma função que retorna um vetor de valores.

Comando
```
x = minmax ('fun',x0,options,vlb,vub,'grad')
```
`fun`: `fun.m` é arquivo que contem as funções
`x0`: solução inicial
`options`: algumas opções (mudanças de parâmetros)
`vlb`: limite inferior de *x*
`vub`: limite superior de *x*
`grad`: usa o gradiente das funções gravados no arquivo `grad.m`

Exemplo (21)
Determine o valor *x* que minimiza o valor máximo de $[f(x), g(x), h(x), v(x), w(x)]$, onde

$$f(x) = 2x^2 + y^2 - 48x - 40y + 304$$
$$g(x) = -x^2 - 3y^2$$
$$h(x) = x + 3y - 18$$
$$v(x) = -x - y$$
$$w(x) = x + y - 8$$

partindo do ponto *x* = [0.1 0.1]. (deve-se incluir no M-*file* de que g = [])

7.11. `nnls`

Objetivo
Resolve o problema dos mínimos quadrados não negativo, ou seja,

$$\underset{x}{minimize} \frac{1}{2}\|Ax - b\|_2^2 \quad \text{sujeito a} \qquad x \geq 0$$

onde *A* é uma matrizes e *b* e *x* são vetores.

Comando
```
[x, w] = nnls(A,b,tol)
```
`A,b`: matriz e vetor dos coeficientes do sistema linear *Ax* = *b*
`tol`: grau de precisão da solução final
`w`: retorno o vetor dos multiplicadores de Lagrange (dual)

7.12. `qp`

Objetivo
Resolve problemas de programação quadrática

$$\underset{x}{minimize} \frac{1}{2}x^T Hx + c^T x \qquad \text{sujeito a} \qquad Ax \leq b$$

onde *H e A* são matrizes e *c* , *b* e *x* são vetores.

Comando
```
[x, lambda, how] = qp(H,c,A,b,vlb,vub,x0,neqcstr,display)
```
`H, c`: A matriz Hessiana e o vetor c são os coeficentes da função objetiva quadrática. H deve ser simétrica

`A,b`: coeficientes das restrições (que são lineares)
`vlb`: limite inferior de *x*
`vub`: limite superior de *x*
`x0`: solução inicial
`neqcstr`: as primeiras `neqcstr` restrições são de igualdade
`display`: mostra (`display=1`) ou não mostra (`display=-1`) avisos na tela
`lambda`: multiplicadores de Lagrange
`how`: indica erros ocorridos na última iteração

Exemplo (22)

Determinar *x* que minimiza $f(x)=\frac{1}{2}x^2+y^2-xy-2x-6y$, sujeito ás restrições

$$x+y\le 2$$
$$-x+2y\le 2$$
$$2x+y\le 3$$
$$x,y\ge 0$$

(H = [1 –1, -1 2] c = [-2; –6], x = [x;y])

8. Funções estatísticas

O "*toolbox* de estatística" contem várias funções estatísticas. As seguir serão ilustradas algumas destas funções

`corrcoef`:.....Coeficientes de correlação
`cov`:.....Matriz covariância
`geomean`: .....Média Geométrica
`harmmean`:.....Coeficientes de correlação
`mean`: .....Média aritmética
`median`:.....Mediana
`std`:.....Desvio padrão
`var`:.....Variância
`hist`: .....Histograma
`regress`: .....Regressão linear
`normrnd`: .....Geração de números aleatórios usando a distribuição normal
`poissrnd`:.....Geração de números aleatórios usando a distribuição de Poisson
`unifrnd`: .....Geração de números aleatórios usando a distribuição uniforme contínua
`unidrnd`: .....Geração de números aleatórios usando a distribuição uniforme discreta

10. Construção de interface amigável com o usuário

O objetivo nesta seção é estudar a criação de menus e os diversos botões (*buttons*) disponíveis no Matlab. Este estudo será feito a partir de um exemplo.

10.1. Menus

Para gerar um menu deve-se inicialmente criar uma janela na qual serão anexadas as os itens de menu desejados.

10.1.1. Janela

Exemplo
Criar um arquivo denominado de `main.m`

```
clear all;                  %limpar as variáveis
clear('all');               %limpar as variáveis
clear global;               %limpar as variáveis globais
if exist('gcf'), close; end;      %fecha qualquer janela
%
global NUMFIG;              %NUMFIG é uma variável global
%
figurePos=get(0,'DefaultFigurePosition');
figurePos(3:4)=[560 420];
NUMFIG=figure( ...
   'Name','MULTIBENCH - Calculo de Medias', ...
   'NumberTitle','off', ...
   'Color', [.9 1 1],...
   'Visible','on', ...
   'Resize','on', ...
   'MenuBar','none', ...
   'Colormap',[1 1 1], ...
   'Position',figurePos, ...
   'Pointer','watch');
```

10.1.2. Item de menu

Exemplo (continuação do arquivo `main.m`)

```
arq = uimenu(NUMFIG,...
          'label','Arquivos ');
uimenu(arq,'Label','Editar Arquivo', ...
     'Callback','edit');
uimenu(arq,'Label','Fechar as Janelas', ...
     'Callback','close all',...
     'Separator','on');
uimenu(arq,'Label','Sair do MATLAB', ...
     'Callback','quit');

med = uimenu(NUMFIG,...
          'label','Medias');
ntt = uimenu(med,...
          'label','Noturno');
uimenu(ntt,...
     'label','Fisica',...
     'CallBack','medias(''noturno'',''fisica'')');
uimenu(ntt,...
     'label','Estatistica',...
   'CallBack','medias(''noturno'',''estat'')');
diu = uimenu(med,...
          'label','Diurno');
uimenu(diu,...
     'label','Fisica',...
     'CallBack','medias(''diurno'',''fisica'')');
uimenu(diu,...
```

```
            'label','Estatistica',...
            'CallBack','medias(''diurno'',''estat'')');
        aju = uimenu(NUMFIG,...
                  'label','Ajuda');
        uimenu(aju,...
            'label','Como calcular Medias',...
            'CallBack','ajuda(''como'')');
        uimenu(s,...
            'label','Sobre o MEDIAS',...
            'CallBack','ajuda(''sobre'')',...
            'Separator','on');
        watchoff;
```

Para executar main.m devemos ter os arquivos

`medias.m` com dois parâmetros: turno ('diurno', 'noturno') e curso ('fisica', 'estat')

`ajuda.m` com um parâmetro: `'como', 'sobre'`.

Sejam os seguintes arquivos contendo as notas das quatro turmas.

```
FisicaD.txt = tabela 4x3
EstatD.txt = tabela 5x3
FisicaN.txt = tabela 5x3
EstatN.txt = tabela 6x3
```

O arquivo `medias.m` poderia ser

```
function = media = medias(turno, curso);
%função que calcula as médias e as mostra na tela

if strcmp(turno,'diurno')
   if strcmp(curso,'fisica')
     fid = fopen('fisicad.txt, 'r');
     notas = fscanf(fid, 'f%',[4,Inf]);
   else,
     fid = fopen('estatd.txt, 'r');
     notas = fscanf(fid, 'f%',[5,Inf]);
if strcmp(turno,'diurno')
   if strcmp(curso,'fisica')
      fid = fopen('fisicad.txt, 'r');
      notas = fscanf(fid, 'f%',[4,Inf]);
   else,
      fid = fopen('estatd.txt, 'r');
      notas = fscanf(fid, 'f%',[5,Inf]);
   end,
else,
   if strcmp(curso,'fisica')
      fid = fopen('fisican.txt, 'r');
      notas = fscanf(fid, 'f%',[5,Inf]);
   else,
      fid = fopen('estatn.txt, 'r');
      notas = fscanf(fid, 'f%',[6,Inf]);
```

```
end
media = mean(notas);
for i = 1:size(media,2)
  fprintf(['Aluno' int2str(i)], '\n', media(i));
end
```

10.2. Objetos de gráficos e de controle

Os tipos de controle são

1. Push button — por exemplo, os botões de ok, cancel
2. Check box — escolher várias opções
3. Radio button — escolher apenas uma opção
4. Slider — semelhante as barras de rolagem do word
5. Pop-up menu — permite a escolha de um item de uma lista
6. Static text — mostra um texto numa única linha
7. Editable text — o usuário pode digitar um *string* para o programa
8. Frame — são caixas de enfeite que englobam os botões

10.2.1. Frames

```
Position = [Pos_H, Pos_V, comprimento,altura]
```

Criar o arquivo `bot00.m`

```
NUMFIG=figure(...
        'Name','CONTROLES - Push Button',...
        'MenuBar','none', ...
        'NumberTitle','off',...
        'Color', [1 .7 .2]);
engloba = uicontrol(NUMFIG,...
        'Style','frame',...
        'BackgroundColor',[.5 1 1],...
        'Position',[55 300 440 100]);
```

10.2.2. Static text

Criar o arquivo `bot01.m`

```
NUMFIG=figure(...
        'Name','CONTROLES - Static Text',...
        'MenuBar','none', ...
        'NumberTitle','off',...
        'Color', [1 .7 .2]);
uicontrol(NUMFIG,...
        'Style','frame',...
        'BackgroundColor',[.5 1 1],...
        'Position',[55 300 440 100]);
uicontrol(NUMFIG,...
        'Style','text',...
```

```
        'String','Digite a nota dos alunos',...
        'BackgroundColor',[1 1 1],...
        'Position',[57 381 435 17]);
```

10.2.3. Editable text

```
NUMFIG=figure(...
        'Name','CONTROLES - Editable Text',...
        'MenuBar','none', ...
        'NumberTitle','off',...
        'Color', [1 .7 .2]);
uicontrol(NUMFIG,...
        'Style','frame',...
        'BackgroundColor',[.5 1 1],...
        'Position',[55 300 440 100]);
uicontrol(NUMFIG,...
        'Style','text',...
        'String','Digite a nota dos alunos',...
        'BackgroundColor',[1 1 1],...
        'Position',[57 378 435 20]);
uicontrol(NUMFIG,...
        'Style','text',...
        'String','Margarete: ------------>',...
        'HorizontalAlignment','Left',...
        'Position',[57 350 130 20]);
nota(1) = uicontrol(NUMFIG,...
        'Style','edit',...
        'String','00',...
        'BackgroundColor',[1 1 .5],...
        'Position',[200 350 75 16]);
uicontrol(NUMFIG,...
        'Style','text',...
        'String','Pedro: ------------>',...
        'HorizontalAlignment','Left',...
        'Position',[57 330 130 20]);
nota(2) = uicontrol(NUMFIG,...
        'Style','edit',...
        'String','00',...
        'BackgroundColor',[1 1 .5],...
        'Position',[200 330 75 16]);
```

10.2.4. Push button

```
NUMFIG=figure(...
        'Name','CONTROLES - Editable Text',...
        'MenuBar','none', ...
        'NumberTitle','off',...
        'Color', [1 .7 .2]);
uicontrol(NUMFIG,...
```

```
        'Style','frame',...
        'BackgroundColor',[.5 1 1],...
        'Position',[55 300 440 100]);
uicontrol(NUMFIG,...
        'Style','text',...
        'String','Digite a nota dos alunos',...
        'BackgroundColor',[1 1 1],...
        'Position',[57 378 435 20]);
uicontrol(NUMFIG,...
        'Style','text',...
        'String','Margarete: ------------>',...
        'HorizontalAlignment','Left',...
        'Position',[57 350 130 20]);
nota(1) = uicontrol(NUMFIG,...
        'Style','edit',...
        'String','00',...
        'BackgroundColor',[1 1 .5],...
        'Position',[200 350 75 16]);
uicontrol(NUMFIG,...
        'Style','text',...
        'String','Pedro: ------------>',...
        'HorizontalAlignment','Left',...
        'Position',[57 330 130 20]);
nota(2) = uicontrol(NUMFIG,...
        'Style','edit',...
        'String','00',...
        'BackgroundColor',[1 1 .5],...
        'Position',[200 330 75 16]);
uicontrol(NUMFIG,...
        'Style', 'push',...
        'Position', [150 250 100 25],...
        'String', 'Iniciar Grafico',...
        'CallBack', 'figure(''Position'', [100 100
        550 200]); plot([1 2 3 4 5],[7 5 6 10 9],
        ''r'')');
uicontrol(NUMFIG,...
        'Style', 'push',...
        'Position', [300 250 100 25],...
        'String', 'Cancelar Grafico',...
        'CallBack', 'close all');
```

10.2.5. Check Boxes

Arquivo bot02.m

```
NUMFIG=figure( ...
        'Name','CONTROLES - Check Box', ...
        'NumberTitle','off', ...
        'Color', [1 .7 .2]);
uicontrol(NUMFIG,...
```

```
                'Style', 'push',...
                'Position', [100 10 100 25],...
                'String', 'Iniciar Grafico',...
                'CallBack', 'figure(''Position'', [100 250
                550 250]); plot([1 2 3 4],[1 4 9 16],
                ''r'')');
        uicontrol(NUMFIG,...
                'Style', 'push',...
                'Position', [250 10 100 25],...
                'String', 'Cancelar Grafico',...
                'CallBack', 'close all');
```

Atributos
Get Set
Botões radio frames
Callback
Menus
Arquivos de Help

11. Alguns *sites* na Internet

http://www.mathworks.com/products/matlab
http://www.stewart.cs.sdsu.edu/cs205/
http://www.engin.umich.edu/group/ctm/extras/function.html
http://www-stat.stanford.edu/~wavelab/
http://eulero.ing.unibo.it/
http://www.umassd.edu/SpecialPrograms/Atlast/welcome.html
http://www.cmsa.wmin.ac.uk/COURSE/matlab_intro.html
http://www.mathtools.net/

12. Referências Bibliográficas

1) Le-Huy, Hoang: *Introduction à Matlab et Simulink*. Département de génie électrique et de génie informatique, Université Laval, Québec, Canada, 1998

2) *Matlab: High-performance Numeric Computation and Visualization Software, User's Guide*. The Math Works Inc., ver. 4.0, 1992.

3) Leon, Steve J.: *Álgebra Linear com Aplicações*. 4ª edição, LTC, RJ, 1999

4) Marques, Jair M.: *Introdução ao Matlab*. Departamento de Estatística/UFPR, 1996