Introdução ao Octave

Fernando Lourenço de Souza

Junho – 2003

Sumário

CAPÍTULO
1

Introdução

O octave originalmente foi concebido para ser um software companheiro, como um livro texto de graduação no projeto de um reator químico que estava sendo escrito por James B. Rawlings, da Universidade Wisconsin-Wisconsin-Madison, e John G. Ekerdt da Universidade do Texas. Claramente, o octave é agora muito mais do que apenas um pacote destinado a sala de aula.

> *"Embora nossos objetivos iniciais fossem um tanto vagos, nós sabíamos que teríamos que criar algo que permitiria aos estudantes a resolver problemas realísticos, e que poderia ser usado para muitas coisas à exceção dos problemas químicos do projeto do reator."* *John W. Eaton*

Originalmente pretendia-se usar o Octave para ensinar o projeto do reator, mas ele acabou sendo usado também nas diversas disciplinas da graduação do departamento de engenharia química e do departamento da Matemática da Universidade do Texas, que tem usado este software para ensinar equações diferenciais e também álgebra linear. Todos são incentivados a compartilhar este software com o outro sob os termos gerais de licença pública do GNU. Você também é incentivado a ajudar fazer do octave um programa mais útil escrevendo e contribuindo com funções adicionais para ele, e relatando todos os problemas que você tiver.

Atualmente existem excelentes programas de livre distribuição e/ou de distribuição gratuita disponíveis para a realização das mais diversas atividades de pesquisa e ensino. Esses programas são desenvolvidos por milhares de pessoas no mundo inteiro e tornam-se cada vez mais de interesse da comunidade científica e do público em geral. Muitas das atividades cientificas e de ensino nos cursos de Cálculo e Física, são desenvolvidas utilizando programas proprietários(Por exemplo: MatLab®). Muitos desses programas possuem similares gratuitos que podem auxiliar a reduzir os custos de tais ferramentas computacionais, ao mesmo tempo em que dá acesso irrestrito a ferramentas robustas. Um desses programas é o GNU-Octave, utilizado para cálculos numéricos.

Como se pôde ver nesta apostila, o software Octave tem uma sintaxe semelhante à do MatLab®, oferecendo recursos suficientes à maioria dos usuários na resolução de problemas numéricos de álgebra linear, na manipulação de polinômios, integração de equações diferenciais ordinárias e equações algébrico-diferenciais, além de outros vários recursos que o software oferece.

Além do método interativo, o octave também aceita o modo batch, no MatLab® representado pelos ".m files"que ele também interpreta na sua maioria, bastando que se salve no local adequado (`octave_ files`), assim como foi visto no capítulo 6.

As tendências de utilização de Free software tem crescido cada vez mais, como o Linux comparado com o Windows®, ao qual muitas pessoas, empresas e Instituições Acadêmicas estão optando pelo Linux, não só por ser Free, mas também pela eficiência que esse tipo de software oferece.

Sendo assim, por esses e outros motivos, o Octave é a melhor alternativa Free quando comparado com o MatLab®.

1.1 Notas Sobre as diferenças entre MatLab e Octave

O Octave é 95% compatível com o Matlab (o número é dos seus criadores). A maioria das diferenças é ao nível das rotinas gráficas, como veremos no Capítulo 6. Abaixo apresentamos alternativas a certos comandos que estão presentes no MatLab e não estão implementadas no Octave.

MatLab	Significado	Octave
intersect(A,B)	$A \cap B$	`intersection(A,B)`
unique(A)	Ordenar e eliminar repetições	`create_set(A)`
setdiff(A,B)	$A \setminus (A \cap B)$	`complement(B,A)`
setxor(A,B)	$(A \cup B) \setminus (A \cap B)$	`complement(intersection(A,B),union(A,B))`

ismember(A,B)	TRUE onde $\{a \in A$ e $a \in B\}$	`t=zeros(1,length(A));`
		`for i=1:length(B),`
		`t(find(A==B(i)))=1;`
		`end;`

mod(a,b)	Resto da divisão inteira	`abs(rem(a,b))*sign(b)`

Note que a diferença entre MOD (que só existe no Matlab) e REM (que existe em ambos) só aparece se a*b<0.

1.2 Distribuição do Programa

O octave é um software livre. Isto significa que todos podem usá-lo adquirindo-o e redistribuindo-o em determinadas circunstâncias. Ele é registrado e há algumas limitações em sua distribuição, mas as limitações são projetadas para assegurar-se de que outras pessoas tenham a mesma liberdade para usar e redistribuir o Octave que você tem. As circunstâncias podem ser encontradas na licença geral pública de GNU que vêm com octave e que aparece também na licença geral publica da seção GNU. O Octave está disponível em cópia de CD-ROM com várias coleções de outros softwares livres, e da fundação do software livre. O octave está também disponível na Internet no site de ftp://ftp.che.wisc.edu/pub/octave , e a informação adicional está disponível no site de http://www.che.wisc.edu/octave.

Originalmente o octave foi implementado para o ambiente Linux. Todavia existe uma versão para windons, em que usa o programa cygwin para emular um ”ambiente linux ´´, onde o octave é executado.

1.3 Ambiente de trabalho

Ao executar o octave temos uma janela com a linha de comando da forma:

```
GNU Octave, version 2.1.36 (i686-pc-cygwin). Copyright (C) 1996,
1997, 1998, 1999, 2000, 2001, 2002 John W. Eaton. This is free
software; see the source code for copying conditions. There is
ABSOLUTELY NO WARRANTY; not even for MERCHANTIBILITY or FITNESS
FOR A PARTICULAR PURPOSE.  For details, type ‘warranty’.

Report bugs to <bug-octave@bevo.che.wisc.edu>.

>>
```

O símbolo >> representa o pront do octave e a partir dele que digitamos os comandos.

1.4 Como avaliar os possíveis erros no Octave

O Octave informa dois tipos de erros para programas inválidos. Um erro devido análise gramatical, que acontece se o Octave não entender algo que você digitou, por exemplo, se você escreve errado um determinado comando, tem-se:

```
>> functon y=f(x) y=x^2;endfunction
>>
```

O octave responderá com uma mensagem do tipo:

```
parse error:
>> functon y=f(x) y=x^2;endfunction
         ^
>>
```

Para a maioria dos erros, devido a análise gramatical, o Octave usa um sinal de intercalação, para marcar o ponto específico onde esta o erro. Neste exemplo, o Octave acusou um erro de comando mal escrito, functon em vez de function e como ficou faltando a letra 'i', o cursor apontou exatamente para a letra 'o', onde deveria conter também a letra 'i'. Estas mensagens facilitam muito a identificação e correção dos erros de compilação. Outra forma de erro é quando o Octave não reconhece uma variável ou a estrutura de um comando, como o exemplo abaixo, em que a variável xnão foi definida.

```
>> x
error: ‘x’ undefined near line 1 column 1
>>
```

No Octave, não é necessário redigitar comandos que já foram digitados anteriormente, basta apenas utilizar as teclas direcionais do computador (as setas $\uparrow\downarrow$) para que se possa obter comandos anteriores, sendo que estes são vizualizados no sentido inverso, ou seja, do último para o primeiro. E ainda, pode-se utilizar estas teclas para movimentar o cursor dentro de um comando, podendo este ser corrigido, deletando (usando a tecla delete ou backspace) ou inserindo novos caracteres.

CAPÍTULO
2

Operadores Aritméticos

O Octave, assim como outros softwares de programação, possui as operações obedecem uma precedência de acordo com a tabela abaixo:

Operador	**Ordem de precedência**
− Subtração	Potenciação
+ Adição	Multiplicação/Divisão
∗ Multiplicação	Adição/Subtração
^ Potenciação	
/ Divisão	

Abaixo, tem-se alguns exemplos de como o Octave obedece a precedência destes operadores:

```
>> 2*2^3
ans = 16
>>
```

Neste exemplo, pode-se verificar que primeiramente é realizado a operação de potenciação que precede sobre as demais, e posteriormente a multiplicação que possui precedência igual a divisão. Observe que o Octave atribui o resultado das operações realizadas em uma variável cujo nome é "ans". Esta variável "ans"pode ser guardada em uma outra variável que o programador pode escolher, por exemplo:

```
>> x=ans
x = 16
>>
>> fer=x
fer = 16
>> vc=fer
vc = 16
>>
```

E assim sucessivamente, podendo, posteriormente, executar operações com estas variáveis, tais como:

```
>> resultado=fer*vc+x
resultado = 272
>>
```

Observe que aqui o sinal de "="significa atribuição, no qual o resultado da operação executada a esquerda do sinal de "="é armazenado na variável cujo nome foi atribuído pelo próprio programador, no caso do exemplo acima foi "resultado".

Pode ser utilizado vírgula para separar comandos numa mesma linha, por exemplo:

```
>> x=12,y=15,t=x
x = 12 y = 15 t = 12
>>
```

E ainda ponto e vírgula para executar a operação, atribuir o resultado à variável designada e não mostrar o resultado da operação na tela, exemplo:

```
>> total=x*y+t*y;
>>
```

Este procedimento é indicado para programas longos e que não se tenha interesse nos resultados intermediários. O procedimento de mostrar o resultado na tela aumenta o custo computacional (tempo de execução + memória) Quando se executa estes tipos de comandos e que deseja-se visualizar o valor atribuído em uma determinada variável, ela poderá ser visualizada digitando simplesmente o seu nome, exemplo:

```
>> total
total = 360
>>
```

Além das precedências existentes entre os operadores $(+, -, *, /,)$, pode-se usar ainda parêntesis para alterar a ordem de operação, sendo os parêntesis mais internos avaliados antes dos mais externos.

CAPÍTULO
3

Variáveis

Assim como em outros programas, o Octave tem certas regras para nomear as variáveis. Estas variáveis devem ser nomes iniciados por letras e não podem conter espaços nem caracteres de pontuação. Além do que o Octave distingue letras maiúsculas de minúsculas.

Algumas variáveis já possuem um significado predefinido no Octave (palavras chaes), não podendo estas serem utilizadas para uma outra finalidade que não seja a sua própria. Algumas destas são:

- ans - variável usada para os resultados.
- pi - número pi.
- eps - menor número tal que, quando adicionado a 1, cria um número maior que 1 no computador.
- flops - armazena o número de operações em ponto flutuante realizadas (presente nas não implementada).
- inf - significa infinito.
- NaN ou nan - significa não é um número, por exemplo, 0/0.
- i e j - unidade imaginária de um número complexo.
- 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 em valor absoluto.
- realmax - maior número que o computador pode armazenar em valor absoluto.

Todas as variáveis, com excessão das predefinidas, podem ser renomeadas ou atribuído a elas qualquer valor, a todo instante.

Caso o usuário queira vizualizar as variáveis declaradas, apagar alguma delas, ou apagar todas elas, basta que se digite os comandos:

- who - mostra as variáveis declaradas pelo usuário.

- clear palavra - apaga a variável chamada palavra.
- clear - apaga todas as variáveis declaradas pelo usuário.

3.1 Formatos Numéricos do Octave

O Octave quando nos fornece algum resultado numérico, ele utiliza determinados critérios de apresentação deste resultado. Isto irá depender da forma de apresentação e do número de casas que estão sendo utilizados. Por exemplo:

```
>> 1/3
ans = 0.33333
```

- format short, utiliza-se 4 dígitos para mostrar o resultado.

```
>> format short
>> 1/3
ans = 0.333
```

- format bank, utiliza-se 2 dígitos para mostrar o resultado.

```
>> format bank
>> 1/3
ans = 0.33
```

- format long, utiliza-se 14 dígitos para mostrar o resultado.

```
>> format long
>> 1/3
ans = 0.333333333333333
```

- format short e, utiliza-se 3 digitos, mais o expoente.

```
>> format short e
>> 1/3
ans =  3.33e-01
```

- format long e - 16 dígitos e expoente,
- format hex - representa o número na base 16
- format + - Mostra o símbolo + nos elementos não nulos da matriz e deixa em branco os nulos,

Obs: O Octave possui a sua própria forma interna de representar os números, portanto, utilizando-se estes comandos, só irá alterar a forma de exibição na tela.

CAPÍTULO
4

Matrizes e Vetores

4.1 Como criar uma Matriz

Para criar uma matriz e armazená-la em uma variável de modo que se possa consultá-la mais tarde, basta digitar os elementos da matriz entre colchetes [...], sendo os elementos de uma mesma linha da matriz separados por vírgula ou espaço e as linhas separadas por ponto e vírgula. Por exemplo:

```
>> x=[2,5,8;5,6,7]
x =

        2        5        8
        5        6        7
>>
```

Ou

```
>> x=[2 5 8;5 6 7]
x =

        2        5        8
        5        6        7
>>
```

Caso o usuário queira criar uma matriz sem que ela seja mostrada na tela, basta colocar apenas ';' no final, como por exemplo:

```
>> x=[2,5,8;5,6,7];
>>
```

Obs: Este procedimento(;) pode ser aplicado em qualquer circunstância.

Estes três casos fornecem o mesmo resultado, que é o de armazenar estes elementos em forma de matriz (2 linhas por 3 colunas) numa variável cujo nome é 'x'.

Para gerar uma matriz com elementos igualmente espaçados, utiliza-se o seguinte comando:

```
>> 0:2:10
ans =

   0   2   4   6   8  10

>>
```

O Octave possui ainda funções para geração de matrizes especiais, tais como:

- eye - matriz identidade

```
>> I=eye(3)
I =
     1     0     0
     0     1     0
     0     0     1
>>
```

- ones - matriz cujos todos os elementos são 1

```
>> Uns=ones(3)
Uns =

        1        1        1
        1        1        1
        1        1        1

>>
```

- zeros - matriz nula

```
>> Z=zeros(3)
Z =

        0        0        0
        0        0        0
        0        0        0

>>
```

- rand - matriz com valores aleatórios

```
>> Aleat=rand(3)
Aleat =

  0.00207  0.59451  0.58122
  0.07802  0.25870  0.62960
  0.47869  0.01718  0.48988

>>
```

4.2 Como deletar uma linha ou uma coluna de uma Matriz

Para deletar alguma linha ou coluna de uma Matriz, deve-se utilizar o seguinte comando:

```
C =

        2         5         8         4
        8         9         7         6
        3        15        12        11

>>
>> C(3,:)=[]
C =

        2         5         8         4
        8         9         7         6

>> C(:,2)=[]
C =

        2         8         4
        8         7         6

>>
```

- C(3,:), deleta a linha 3 da Matriz C;
- C(:,2), deleta a coluna 2 da Matriz C resultante.

4.3 Como acessar os elementos da Matriz

Para acessar cada elemento da matriz utiliza-se a notação de matrizes B(i,j), onde 'i' é a linha, e 'j' é a coluna onde o elemento da matriz 'B' está. Por exemplo:

```
>> B=[2 3 4;5 6 7]
B =

        2         3         4
        5         6         7

>> B(1,2)
ans = 3
>> B(2,2)
ans = 6
>> B(1,:)
ans =

        2         3         4

>> B(:,2)
```

```
ans =

        3
        6
>> B(:,1:2)
ans =

        2        3
        5        6

>>
```

Observe que:

- B(1,2), este comando fornece o elemento da linha 1 e coluna 2;
- B(2,2), este comando fornece o elemento da linha 2 e coluna 2;
- B(1,:), este comando fornece todos os elementos da linha 1;
- B(:,2), este comando fornece todos os elementos da coluna 2;
- B(:,1:2), este comando fornece os elementos de todas as linhas das colunas 1 até a 2.

4.4 Como acrescentar linhas ou colunas numa dada Matriz

```
>> A=[5 6 7;5 8 9]
A =

        5        6        7
        5        8        9

>> B=[A,[15;12]]
B =

        5        6        7       15
        5        8        9       12

>> C=[B;[16 17 18 19]]
C =

        5        6        7       15
        5        8        9       12
       16       17       18       19

>>
```

- B=[A,[15;12]], este comando, acrescenta-se a Matriz A, a quarta coluna formada pelos elementos 15 e 12;
- C=[B;[16 17 18 19]], este comando, acrescenta-se a Matriz A, a terceira linha formada pelos elementos 16, 17, 18 e 19.

4.5 Operações Matriciais

As operações matriciais são executadas de forma semelhante a que são executadas as operações escalares, como por exemplo:

```
>> A=[5 6;4 3]
A =

        5        6
        4        3

>> B=[5;3]
B =

        5
        3

>> C=[4 -3;7 8]
C =

         4        -3
         7         8

>> A+C
ans =

        9        3
       11       11

>> 4*B
ans =

       20
       12


>> A*C
ans =

       62       33
       37       12

>> C^2
ans =

        -5       -36
        84        43

>> C.^3
ans =
```

```
        64       -27
       343       512

>> (A*C)^2
ans =

     5065     2442
     2738     1365

>> (A+C)*B
ans =

       54
       88

>>
```

As operações Matriciais podem ser resumidas em:

Operador	Operação	Exemplo
$+$	Adição	$A + B$
$-$	Subtração	$A - B$
$*$	Multiplicação	$A * B$ ou $k * A$
^	Potenciação (A^k)	A^k
.^	Potenciação elemento a elemento (a_{ij}^k)	A.^k
'	Transposição da Matriz A	A'
[v,x]=eig(A)	Retorna os autovetores em v e os autovalores em x	
	Visualização da coluna j da matriz A	A(:,j)
	Visualização da linha i da matriz A	A(i,:)
	Elemento a_{ij} da matriz A	A(i,j)

CAPÍTULO
5

Funções

5.1 Funções elementares

O Octave possui várias funções científicas prè-definidas, cujo cálculo destas funções, é semelhante a obtenção através das calculadoras, por exemplo:

```
>> x=-4
x = -4
>> y=-20
y = -20
>> t=5
t = 5
>> k=lcm(x,y,t)
ans = 20
>> z=atan(k)
z = 1.5208
>> z_graus=(k*180)/pi
z_graus = 1145.9
>> anglo_efetivo=1145.9/360
anglo_efetivo = 3.1831
>>who

*** currently compiled functions:

findstr  gcd      lcm      rows

*** local user variables:

t  x  y  z  k  z_graus
>>
```

Lembrete: O comando who, lista as varíaveis declaradas pelo o usuário.

Algumas destas funções pré-definidas estão listadas abaixo:

- 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.
- cos(x) - cosseno de x.
- exp(x) - exponencial ex.
- 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.
- sin(x) - seno de x.
- sqrt(x) - raiz quadrada de x.
- tan(x) - tangente de x.

5.2 M-arquivos

As funções podem ser trabalhadas de duas formas, uma delas é programar a função no prompt do Octave e utilizá-la, ou programar e salvá-la em um arquivo como a extensão ".m"(recurso este conhecido como programação em lote, ou script(roteiro)), ao qual deve ser feito num editor de texto puro, ou seja aquele que salva o arquivo em código ASCII. Neste segundo tipo, para que a versão Windows do Octave, reconheça estas funções, deve-se salvar estes programas na pasta que o programa cria no momento de instalação, cujo endereço é:

```
Diret{\'o}rio de instala\c{c}{\~a}o\Arquivos de programas\GNU Octave 2.1.36
\octave_files
```

Neste caso, depois de ter salvado estas funções neste arquivo, para utilizá-las, basta apenas que se digite no prompt do Octave o nome e os parâmetros, ao qual foi salvo a função.

Exemplos:

```
>> function y=f2(x)
b=0.01; a0=10; c=1000;
y=a0*exp(b*x)*sin(2*pi/c*x)
endfunction
```

Para que o Octave reconheça esta função, cujo programa foi feito em um outro software, ela deverá ter sido salva na pasta `octave_files` com extensão ".m", na qual o nome da função tem que ser "f2.m". E para executar esta função, basta digitar-se no prompt do Octave da seguinte maneira:

```
>> y=f2(3)
y = 0.19422
```

Outro exemplo:

```
function X=fourier2(x,N)
% Calcula a N pontos da transformada de fourier de uma sequencia(vetor x)
% implementado com 1 loop 'for'

X=zeros(1,N);
L=length(x);
for k=0:(N-1)
     w= -j*2*pi/N*k;
     X(k+1)=sum(x .* exp(w .* (0:(L-1)) ) );
end
```

Neste caso, a função deverá ser salva como "fourier2.m", e para executá-la, basta digitar:

```
>> x=fourier2(2,4)
x =
        2        2        2        2
>>
```

5.3 Solução de sistemas lineares Ax=b

Por exemplo:

```
>> A=rand(3,3)
A =
  0.16808  0.53037  0.13528
  0.84642  0.09465  0.97188
  0.77846  0.88364  0.02581

>> b=rand(3,1)
b =
  0.82943
  0.59267
  0.31844
>> x=A\b
x =
  -1.44673
   1.58480
   1.71545
>>
```

x=A\b, é conceitualmente equivalente a usar $(A^{-1})b$, o que da a resolução deste tipo de problema.

5.4 Solução de um conjunto de equações não-lineares

Neste caso, tem-se:
Sejam:

$$y_1 = -2x_1^2 + 3x_1x_2 + 4sen(x_2) - 6$$
$$y_2 = 3x_1^2 - 2x_1x_2^2 + 3cos(x_1) + 4$$

Com as condições iniciais $y_1 = 1, y_2 = 2$. Então no Octave, a solução é implementada como:

```
>> function y=f(x)
y(1)=-2*x(1)**2+3*x(1)*x(2)+4*sin(x(2))-6;
y(2)=3*x(1)**2-2*x(1)*x(2)**2+3*cos(x(1))+4;
endfunction
>> [x,info]=fsolve('f',[1;2])
x =
  0.57983
  2.54621
info = 1
>>
```

Obs: O valor $info = 1$, indica que a solução converge.

5.5 Cálculo da integral definida em um intervalo para uma variável

Para isso, utiliza-se a função **quad**, em uma operação estabelecida da seguinte forma:

[v,ier,nfun,err]=quad("f",a,b,tol,sing)

Onde **f** do lado direito da expressão, é o nome da função a ser chamada para calcular o valor do integrando(deve ter a forma $y = f(x)$, sendo **y** e **x** escalares); os argumentos a e b são os limites da integração(que podem ser +-infinito); o argumento tol é um vetor que especifica a precisão desejada do resultado; e o argumento $sing$ é um vetor de valores em que o integrando é conhecido como singular. Quanto ao lado esquerdo da expressão, têm-se que n é o resultado da integração e ier é um código de erro(0 indica que a operação foi bem sucedida); o valor $nfun$ indica quantas iterações foram necessárias; e o valor de err é uma estimativa do erro na solução. Por exemplo:

Seja:

$$\int_0^3 xsen\left(\frac{1}{x}\right)\sqrt{|1-x|}\,dx$$

então, no Octave, a solução é implementada como:

```
>> function y=f(x)
y=x.*sin(1./x).*sqrt(abs(1-x));
endfunction
>> [v, ier, nfun, err]=quad("f", 0, 3)
```

```
  v = 1.9819
  ier = 1
  nfun = 5061
  err =  1.1522e-07
>>
```

Obs: Embora prescindível para o cálculo de $quad$, a forma $ponto$ dos operadores usados no GNU/Octave torna mais fácil criar um conjunto de pontos para gerar gráficos(pois isso possibilita utilizar argumentos vetoriais para produzir resultados vetoriais).

CAPÍTULO
6

Gráficos

6.1 Plotando em 2 dimensões

Todas as funçoes de plotagem do octave usam o Gnuplot para o manuseio dos gráficos. Existem dois níveis de funções, gplot(manuseio de gráficos 2D) e gsplot(manuseio de gráficos 3D)., que comporta-se quase exatamente quanto o correspondente das funções do MatLab, que é o plot e o splot. Um exemplo simples em que usamos o comando plot para gerar o gráfico de duas curvas.

```
>> t = 0:0.1:6.3;
>>plot(t, cos(t), "-b;cos(t);", t, sin(t), "+3r;sen(t);");
```

Os termos entre aspas indica o formato da linha e a legenda a ser apresentada pelo gráfico. No exemplo indicamos que a curva do cosseno deve ser sólida (-), na cor azul (b) e na legenda a curva é identificada por cos(t) (;cos(t);). No caso do seno indicamos que a curva será indicada pelo sinal de + a cada três pontos, na cor vermelho (r), com legenda sen(x) (Ver Figura 6.1).

Comando gplot

Você pode plotar expressões mútiplas com um simples comando pela separação delas por vírgulas. Cada expressão deve ter seu próprio ajuste de requisitos. A expressão para ser plotada não deve conter qualquer matriz literal([1,2;3,4]), por exemplo:

```
> gplot [-11:11] [-1.1:1.1]  \
```

Obs: Ele aceitará o comando mas não irá plotar nada.

- **Comando: data with lines**, Desenha o gráfico como uma linha reta.
- **Comando: gset***opções*, Opções de ajuste dos parâmetros de plotagem.
- **Comando: gshow** *opções*, Mostra os parâmetros de plotagem.
- **Comando: replot**, Plota o gráfico novamente.
- **Comando: closeplot**, Fecha a janela do gráfico.
- **Comando: axis**, Limites dos eixos.

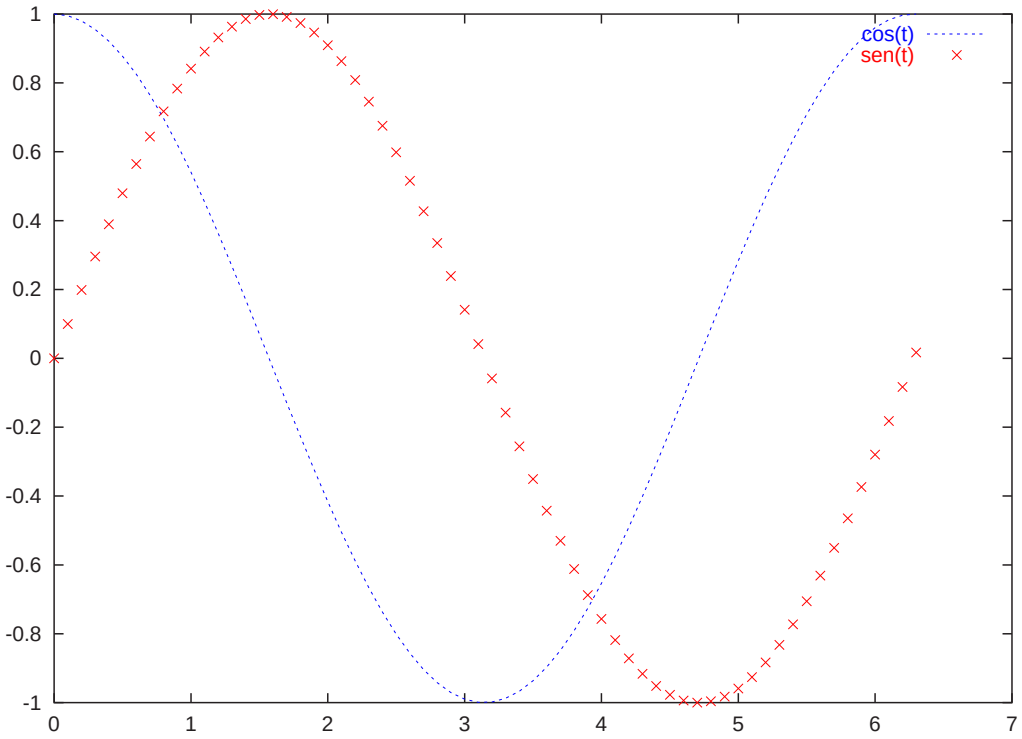

Figura 6.1: Exemplo de gráfico

6.1.1 Funções Especiais de Plotagem em 2 Dimensões

- **Comando: bar(x,y)**, Dados 2 vetores de dados x e y, é produzido um gráfico de barras. Se somente um argumentmo é dado, pegamos os valores de y, e as coordenadas x são colocadas para ser o índice dos elementos.

  Bar(x,y) equivalente a:

  [xb,yb] = bar(x,y); plot(xb,yb);

- **Comando: loglog(**$args$**)**, Plota em escala logaritmica ambos os eixos.
- **Comando: polar(**θ, ρ**)**, Faz um gráfico de acordo com as coordenadas polares θ e ρ.
- **Comando: semilogy(**$args$**)**, Gráfico logarítmico somente no eixo y.
- **Comando: semilogx(**$args$**)**, Gráfico logarítmico somente no eixo x.
- **Comando: stairs(**x, y**)**, Gráfico em degrau.
- **Comando: loglog(**$args$**)**, Plota em escala logaritmica ambos os eixos.

6.2 Plotando em 3 dimensões

Há muitas formas de se criar um gráfico 3D. Primeiro é necessário gerar a malha de dados a serem visualizados(usando o recurso meshgrid). Para executar a visualização 3D pode-se utilizar o comando mesh e para geração de contornos, o comando contour.

- **Comando: gsplot**, Plota em 3 dimensões.

```
>> x=[1 2 3]
x =
        1        2        3
>> y=[2 5 6]
y =
        2        5        6
>> z=[x;y]
z =
        1        2        3
        2        5        6
>> gsplot z
```

Gráfico 3D

- **Comando: mplot(x1,y1,x2,y2)**, Os vários valores de x e y são plotados num mesmo gráfico. Este comando é necessário quando deseja-se comparar diferentes valores.
- **Comando: plot border(args)**, Especifica o tipo de borda a ser usada, de acordo com os seguintes argumentos:

  blank;
  all;
  north;
  south;
  east;
  west.

- **Comando: subplot(rows,cols,index)**
  Rows, número de linhas no subplot grid;
  Cols, número de colunas no subplot grid;
  Index, index de subplot onde se faz a proxima figura.

6.2.1 Acéssórios para plotagem em 3D

- **Comando: grid**, Coloca uma grade na figura plotada.
- **Comando: title(string)**, Especifica um título para a figura.
- **Comando: xlabel(string)**, Especifica um nome, para o eixos x.
- **Comando: ylabel(string)**, Especifica um nome, para o eixos y.
- **Comando: zlabel(string)**,Especifica um nome, para o eixos z.

Outra forma é usar o comando `mesh` em que temos que definir a malha, pelo comando `meshgrid` para obter os pontos de plotagem

Exemplo de gráficos em 3D:

```
>> [x,y]=meshgrid(-4:0.6:4,-3:0.6:3);
>> z=x.^2+y.^2;
>> grid;
```

```
>> xlabel('eixo x'); ylabel('eixo y'); zlabel('eixo z')
>> title('Exemplo de Mesh');
>> mesh(x,y,z);
>> gset term postscript color
>> gset output "gset3D.ps"
>> replot
```

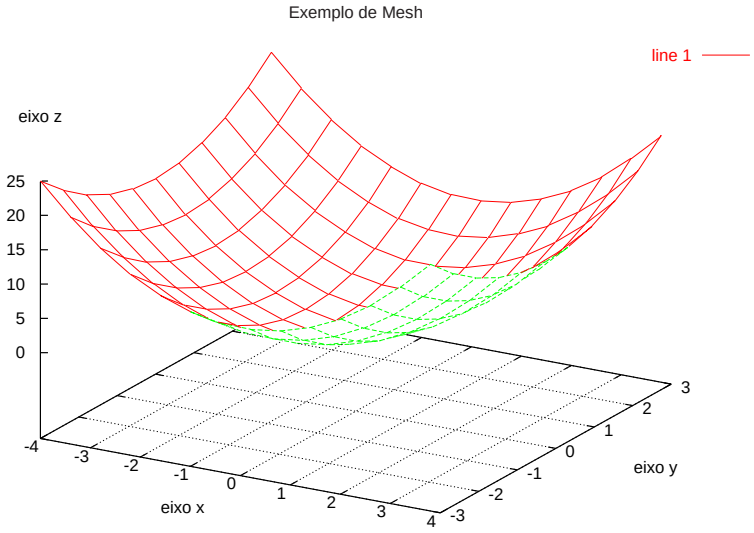

Figura 6.2: Gráfico utilizando a função mesh.

6.3 Como Imprimir ou Salvar os Gráficos

No Octave os processos de impressão e gravação de um determinado gráfico, utiliza a seguinte sequência de comandos:

Para salvar os gráficos:

```
>> gset term postscript eps 22
>> gset output "nome do grafico.ps"
>> replot
```

Onde o primeiro comando define o formato do arquivo, o segundo define o nome do arquivo e o terceiro gera o gráfico de acordo com os parâmetros estabelecidos. Alguns tipos de formato são:

- `postscript`, PostScript graphics language;
- `emf`, Enhanced Metafile Format;
- `epslatex`, LaTex(Text) and encapsulated PostScript;
- `latex`, LaTex picture environment;

- `texdraw`, LaTex texdraw environment.

Para imprimir o gráfico em ambiente Windows®:

```
>> gset term windows
>> gset output "PRN"
>> replot
```

CAPÍTULO
7

Equações Diferenciais

O octave possui variáveis específicas para a resolução de equações diferenciais. A função Lsode pode ser usada para resolver Equações Diferenciais Ordinarias na forma de Problema de Valor Inicial:

$$\begin{cases} \frac{dx}{dt} = f(x,t), \\ x(0) = x_0 \end{cases}$$

7.1 Função: lsode (fcn, x_0, t, t_{crit})

Esta função retorna uma matriz de x elementos como função de t, dado um estado inicial do sistema x_0. Cada linha da matriz resultante, corresponde para um dos elementos do vetor t. O 1° elemento de t corresponde para o valor inicial do estado x_0 então a 1^{a} linha da saída é x_0. O 1^{o} argumento, fcn, é uma string que nomeia a função para chamar para o computador o vetor de correto aspecto para o ajuste de equações.

E deve ter a forma: xdot = f(x,t)
Onde xdot e x são vetores e t está na forma escalar. Ex:

```
>> function xdot=f(x,t)
xdot=zeros(3,1);
xdot(1)=77.27*(x(2)-x(1)*x(2)+x(1)/-8.375e-06*x(1)^2);
xdot(2)=(x(3)-x(1)*x(2)-x(2))/77.27;
xdot(3)=0.161*(x(1)-x(3));
endfunction
>> x0=[4;1.1;4];
>> t=linspace(0,500,1000);
>> y=lsode('f',x0,t)

y =
  4.00000  1.10000  4.00000
     .        .        .
     .        .        .
     .        .        .
  0.99999  0.97619  1.06809
```

7.2 Equações Diferenciais Algébricas

Função: Dassl

É usada para resolver equações diferenciais algébricas. Esta função Retorna uma matriz de estados e sua primeira derivada em relação a t. O $1^{\diamond}$ elemento de t corresponde ao estado inicial x_0 e a derivada $xdot_0$.

[x,xdot] = dassl (fcn, x0 , xdot0, t , tcrit)

O primeiro argumento, *fcn*, é uma *string* que nomeia a função a ser chamada para computar o vetor de residuos para o ajuste de equações. E deve ter a seguinte forma:

res = f(x,xdot,t)

Onde *x*, *xdot* e *res* são vetores, e t é um esacalar.

CAPÍTULO
8

Estatística

Neste capítulo são encontradas funções para cálculo de média, desvio padrão, entre outras.

Função: mean(x)

Se x é uma matriz, esta função irá computar o significado para cada coluna e retorná-las em um vetor linha.

mean(x) = sum_i x(i)/ N

Função: median(x)

Esta função calcula o valor mediano do vetor x.

median(x) = (x(N/2) + x((N/2)+1))/2

Função: std(x)

Esta função calcula o valor mediano do vetor x.

Std(x) = sqrt(sumsq(x- mean(x)) / (x-1))

Função: cov(x,y)

Cada linha de x e y é uma observação e cada coluna é uma variável. A entrada (i,j) ordem de cov(x) é a covariância entre a variável de ordem i em x, e a variável de ordem j em y.

Função: corrcoef(x,y)

Esta função faz a correlação entre a variável de ordem i em x, e a variável de ordem j em y.