Octave

Introdução

O GNU Octave é um software livre sob licença GPL para cálculos matemáticos. Sua linguagem é de alto nível e bastante compatível com o MATLAB. Foi desenvolvido em C++ e possui um interpretador para executar os scripts. Funciona em diversos sistemas operacionais.

Chamando o programa através de um terminal obtemos o sinal “>” que indica que o octave está pronto para receber comandos. Para inserir comentários nos códigos, devemos iniciar a linha com % ou # para que ela seja ignorada pelo interpretador

Primeiros Comandos

Usando o Octave como uma calculadora:

Podemos digitar comandos matemáticos no octave usando-o como se fosse uma calculadora, por exemplo:

octave:1> 6+4
ans = 10

A resposta fica armazenada na variável *ans* (do inglês, answer)
Outras operações aritméticas: -, *, /, ^ (potência)

Algumas funções matemáticas:

abs(x): Módulo de x
acos(x): Arco cosseno é x
cos(x): Cosseno de x
cosh(x): Cosseno hiperbólico
round(x): Arredonda o valor de x
sinh (x): Seno hiperbólico
tan(x): Tangente de x
exp(n): Função exponencial
log10(x): Logaritmo de x na base 10
sum(x): Somatória de x
prod(x): Produtória de x
sumsq(x): Somatória dos quadrados dos elementos de x

Para as funções trigonométricas devemos expressar o ângulo em radianos.

Algumas constantes:

octave:2> e
ans = 2.7183
octave:3> pi
ans = 3.1416

Pata imprimir mensagens na tela podemos usar a função *disp()*:

octave:4> disp("Olá mundo!")
Olá mundo!

Variáveis

O Octave tem certas regras para nomear as variáveis. Os nomes de variáveis devem ser iniciados por letras, não podem conter espaços nem caracteres de pontuação. O Octave diferencia letras maiúsculas de minúsculas. Alguns nomes são de uso restrito, como *pi*, *inf*, *ans*, etc. Fazemos a atribuição com o sinal de igual (=). Por exemplo:

octave:5> a = 3
a = 3

Isto significa que agora a variável *a* tem valor 3. Para que o interpretador não repita o comando recebido, basta colocar ponto e vírgula (*;*) no final.

Para colocar vários comandos em uma mesma linha podemos separá-los por vírgulas ou ponto e vírgulas. Por exemplo:

octave:6> a = 1, b = 2; c = 3.0
a = 1
c = 3

Também podemos usar números imaginários através da variável pré definida i:

octave:7> num_imag = 3 + 4i
num_imag = 3 + 4i

Podemos também usar j no lugar de i.

É importante notar que devemos evitar sobrescrever o valor de variáveis (ou funções) pré definidas como *e*, *pi*, *i*...

- Para saber quais variáveis foram decaradas podemos usar o comando *who*.
- Para apagar uma variável usamos *clear <variavel>* ou *clear all* para apagar todas.

Formatação e precisão numérica

O Octave normalmente exibe os números com seis algarismos significativos.

Apesar de exibí-los dessa forma, o Octave trabalha internamente com uma precisão bem maior, por isso, é bom guardar os resultados em variáveis, no lugar de digitar novamente os valores mostrados, para evitar erros nos resultados.

O comando *format* permite selecionar a forma com que os algarismos são mostrados. Digitando format long o Octave passa a exibir os valores com pelos menos 16 algarismos significativos nas respostas dos cálculos efetuados.

O comando *format* sem parâmetros faz o programa retornar ao seu modo normal de exibição, o comando *format long* mostra o número com 16 casas decimais e o *format bank* com apenas 2 casas decimais:

Além dos números reais e complexos mostrados, outros números são reconhecidos e calculados pelo Octave:

-Infinito (*Inf*) Resultado da divisão de um número por zero. Esta é uma resposta válida e pode ser usada nos cálculos e atribuída a uma variável, assim como outro número qualquer.

-Not a Number (*NaN*) Resultado da divisão de zero por zero e outras operações que geram resultados indefinidos. Novamente, os resultados podem ser tratados como qualquer outro número, apesar dos resultados dos cálculos com seu uso gerarem sempre a resposta *NaN*.

Exemplo:
octave:8> 1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2
ans = 5.5511e-17

O resultado fornecido é bem pequeno, mas não chega a ser exatamente zero, que é a resposta correta. A razão para isto é que 0.2 não pode ser representado em binário usando um número finito de dígitos (em binário fica 0.0011001100...).

Os programas de computador procuram representar esses números da forma mais próxima possível, mas o uso repetido dessas aproximações pode causar problemas.

Comandos úteis

- Podemos recuperar os comandos digitados anteriormente no octave através da seta para cima
- Ajuda: *help nome_do_comando*
- Para interromper a execução de comandos: teclas Ctrl-C.
- Data: *date*
- Data e hora: *clock*

Gráficos bidimensionais

O comando básico para o traçado de gráficos bidimensionais é o plot(x,y). Os parâmetros x e y são as coordenadas a serem traçadas.

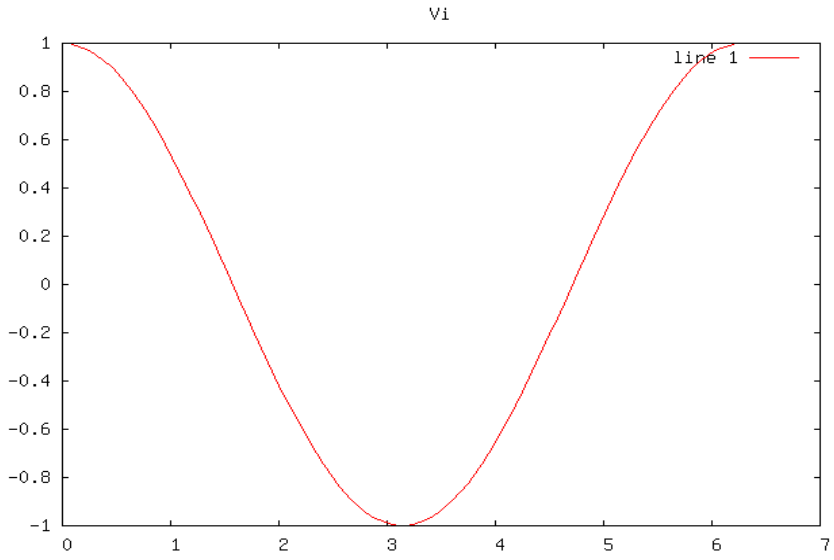
O primeiro passo para o traçado de gráficos 2D é formar uma tabela de coordenadas (x, y) da função a ser plotada.

Para construir um vetor x, pode-se usar o comando:
*octave:9> x=linspace(0, 2*pi);*

A função *linspace(a, b, n)* cria *n* elementos igualmente espaçados entre *a* e *b* (inclusive). O valor padrão onde *n* é 100. Para espaçar os elementos em escala logarítmica poderíamos usar *logspace*.

Para completar a tabela de coordenadas, determina-se o vetor y correspondente aos valores em x. Isso é feito simplesmente invocando a função com o comando, por exemplo:
octave:10> y=cos(x);

Uma vez construída a tabela com as coordenadas (x, y), pode-se usar a função *plot*:
octave:11> plot(x,y)



Linhas de grade podem ser adicionadas ao gráfico com o comando
octave:12>grid on

A aparência do gráfico pode ser modificada com os parâmetros adicionais, colocados entre aspas simples;

Símbolo	Cor	Símbolo	Marcador	Símbolo	Tipo de linha
b	azul	+	Sinal positivo	-	Linha cheia
w	branco	*	Asterisco	..	pontilhado
c	ciano	o	Diamante		
k	preto	x	Letra x		
g	verde				
r	vermelho				
m	magenta				

Os comandos *title*, *xlabel* e *ylabel* permitem escrever um título para o gráfico e um rótulo em cada um dos eixos. Deve-se passar como parâmetro para esses comandos uma seqüência de caracteres entre aspas.

Neles também podemos usar uma formatação de texto especial para símbolos como pi,delta e alfa, existe uma seleção com mais de 75 desses símbolos alguns mais utilizados:

alpha	*theta*
beta	*lambda*
gamma	*pi*
delta	*int*
epsolon	*infty*
omega	*rho*

Para salvar o gráfico obtido usamos o comando *print*:
octave:13>print -dpng graf.png
-d é o dispositivo que irá imprimir

Gráficos tridimensionais

O comando *mesh(x,y,z)* permite traçar a malha para gráficos tridimensionais, da forma z=f(x, y).

Para mudar o ângulo de visão, pode-se clicar com o botão direito do mouse sobre a figura, e arrastá-la para uma nova posição.

Enquanto o comando *mesh(x,y,z)* representa o gráfico por meio de uma malha, o comando *surf(x,y,z)* representa a função tridimensional como uma superfície, adicionando à malha efeitos de cores e profundidade. Uma vez que essa função é acionada, as chamadas subseqüentes à função *mesh* irão também mostrar uma superfície com os mesmos efeitos de profundidade, a não ser que o comando *clf* seja usado antes para limpar a janela gráfica, ou então o comando *close* seja usado para fechá-la.

Funções

Uma função no Octave tem a forma geral:

function [lista-saida] = nome(lista-entrada)
comandos
endfunction

Onde

- *lista-saída* é uma lista de parâmetros de saída da função, separados por vírgula;
- *lista-entrada* é uma lista de parâmetros de entrada, separados por vírgula;
- *nome* é o nome dado à função.

Uma função pode ser criada digitando-a no ambiente de trabalho, ou criando um arquivo com a função e salvando-o no diretório de trabalho. O arquivo deve ter o mesmo nome dado à função e a extensão ".m". Dessa forma o octave pode localizá-la quando essa função for chamada dentro do programa sem necessidade de passar nenhum comando adicional.

Passando o comando

ignore_function_time_stamp ("all")

estamos falando para o octave que nossas funções não serão modificadas em tempo de execução e com isso ganhamos um pouco de desempenho (já que o interpretador não precisa ficar verificando). Para que essa configuração volte ao normal passamos "*system*" como parâmetro do comando anterior.

É importante saber que para passar dois vetores como argumento e aplicar a multiplicação ou divisão elemento a elemento (e não como operação de matrizes) devemos usar ".*" e "./".

É possivel passar valores default na criação da função

Scripts

Um script pode conter diversos comanos que não pertençam a uma única função. Ele não deve começar com a palavra *function* (para não ser interpretado como um arquivo de função). Uma das maneiras de evitar que isso aconteça é colocar um "*1;*" no começo do arquivo.

Para executar um arquivo podemos dar s*ource(arquivo)*, sendo que o arquivo não precisa necessariamente ter a extenção .m

Matrizes

Entrada de dados

As matrizes têm os elementos de uma linha separados por vígula e as linhas separadas por ponto e vírgula por exemplo:

octave:14> A=[0,2,0,1;2,2,3,2;4,-3,0,1.;6,1,-6,-5]

A =

```
  0   2   0   1
  2   2   3   2
  4  -3   0   1
  6   1  -6  -5
```

Para gerar uma matriz com valores aleatórios podemos fazer *rand(<linhas>, <colunas>)*. Para que os números sigam uma outra distribuição aleatória podemos usar *rande*, *randg*, *randp*...

Para gerar uma matriz identidade de dimensão nxn usamos *eye(n)*. Caso não queiramos quadrada, podemos passar as dimensões *m* e *n* como argumento sendo que se n>m as colunas excedentes são completadas com zero e se n<m as colunas excedentes são removidas

Analogamente podemos criar uma matriz só de uns: *ones(m, n)* ou de zeros: *zeros(m, n...)*.

Exemplo: Para criar uma matriz só de 5:
*octave:15> mat = 5*ones(7)*
mat =

```
 5  5  5  5  5  5  5
 5  5  5  5  5  5  5
 5  5  5  5  5  5  5
 5  5  5  5  5  5  5
 5  5  5  5  5  5  5
 5  5  5  5  5  5  5
 5  5  5  5  5  5  5
```

A função *repmat(A, m, n)* copia a matriz *A m* vezes na vertical e n na horizontal

Para criar uma matriz diagonal usamos diag(vec, d) onde vec é o vetor com os elementos que queremos na diagonal, e d é um paramentro opcional de qual subdiagonal queremos colocar o vetor (o padrão é zero, a diagonal principal)

Acessando elementos da matriz:
A(i,j): Elemento aij da matriz A
A(i,:): Linha i da matriz A
A(:,j): Coluna j da matriz A

Removendo elementos da matriz
A(i,:) = [] Remove linha i da matriz A
A(:,j) = [] Remove coluna j da matriz A

Para multiplicar matrizes pordemos usar o operador * normalmente. Para transpor uma matriz podemos usar *'*. Por exemplo:
*octave:16> A'*A*
ans =

```
  56   -2  -30  -22
  -2   18    0   -2
 -30    0   45   36
 -22   -2   36   31
```

Resulução de sistema linear
Para resolver o sistema Ax = b usamos o seguinte operador: '\':
A\b
Usar dessa maneira é mais eficiente que calcular primeiro a inversa de A e depois fazer a multiplicação.

Se a matriz for singular é emitida uma mensagem de warning e a solução que minimiza o resíduo da operação.

Algumas funções para matrizes

det(A)	Determinante de A
norm(A, p)	Norma p (padrão 2) de A
inv(A)	Matriz inversa de A
chol(A)	Fatoração de Cholesky de A (retorna r: r'*r = A)
lu(A)	Fatoração lu de A (retorna l, u e p)
qr(A)	Fatoração QR de A (retorna q, r e p)
eig(A)	Retorna os autovelores e autovalores da matriz A
horzcat(A1, A2, ...)	Concatena as matrizes na horizontal
vertcat(A1, A2, ...)	Concatena as matrizes na vertictal

Controle de Fluxo:

```
if (condition)
     then-body
   elseif (condition)
     elseif-body
   else
     else-body
   endif

switch expression
     case label
       command_list
     case label
       command_list
     ...

     otherwise
       command_list
   endswitch
```

se o *label* for um vetor, entra se satisfizer qualquer um dos elementos dele
o *if* não vale para string, o *switch* vale

```
while (condition)
     body
   endwhile

do
     body
   until (condition)
```

for var = expression
body
endfor

break: sai do for ou while mais interno
continue: leva ao proximo ciclo do for ou while sem sair dele

Referências

http://en.wikipedia.org/wiki/GNU_Octave
http://www.math.uic.edu/~hanson/Octave/OctaveLinearAlgebra.html
http://www.gnu.org/software/octave/doc/interpreter/
http://www.ime.unicamp.br/~marcio/tut2005/octave/042565Cassia.pdf
http://www.castilho.prof.ufu.br/cn/Octave.pdf