Matlab

Histórico

Matlab, de Matrix Laboratory, é um ambiente interativo para computação envolvendo matrizes. Matlab foi desenvolvido no inicio da década de 80 por Cleve Moler, no Departamento de Ciência da Computação da Universidade do Novo México, EUA.

Matlab coloca à disposição do usuário, num ambiente interativo, as bibliotecas desenvolvidas nos projetos LINPACK e EISPACK. Estes projetos elaboraram bibliotecas de domínio público para Álgebra Linear. LINPACK tem rotinas para solução de sistemas de equações lineares, e EISPACK tem rotinas para cálculo de autovalores. Os manuais destes projetos são portanto documentação complementar à documentação do Matlab.

Versões posteriores de Matlab, atualmente na versão 4.0, foram desenvolvidas na firma comercial MathWorks Inc., que detêm os direitos autorais destas implementações. As versões recentes do produto Matlab melhoram significativamente o ambiente interativo, incluindo facilidades gráficas de visualização e impressão; todavia a "Linguagem Matlab" manteve-se quase inalterada. Existem vários interpretadores da linguagem Matlab em domínio publico, como Matlab 1.0, Octave e rlab. Existem também outros interpretadores comerciais de Matlab, como CLAM. Existem ainda várias Tool Boxes, bibliotecas vendidas pela MathWorks e por terceiros, com rotinas em Matlab para áreas específicas.

Usaremos a grafia de nome próprio, **Matlab**, como referência a linguagem, o nome em maiúsculas, **MATLAB**, como referência ao produto comercial da MathWorks, e a grafia em minúsculas, **matlab**, como referência a um interpretador genérico da linguagem Matlab.

O Ambiente

Para entrar no ambiente matlab, simplesmente digite "matlab". O prompt do matlab é `>>` , que espera por comandos. Para sair use o comando `quit`. Dentro do ambiente matlab, um comando precedido do bang, `!`, é executado pelo sistema operacional, assim: usando `!dir` ou `!ls` ficaremos sabendo os arquivos no diretório corrente, e usando `!edit` , `!vi` ou `!emacs` , seremos capazes de editar um arquivo. Normalmente Matlab distingue maiúsculas de minúsculas.

O comando `help` exibe todos os comandos e símbolos sintáticos disponíveis. O comando `help nomecom` fornece informações sobre o comando de nome *nomecom*. O comando `diary nomearq` abre um arquivo de nome *nomearq*, e ecoa tudo que aparece na tela para este arquivo. Repetindo o comando `diary` fechamos este arquivo. O formato dos números na tela pode ser alterado com o comando `format`.

Os comandos `who` e `whos` listam as variáveis em existência no espaço de trabalho. O comando `clear` limpa o espaço de trabalho, extinguindo todas as variáveis. O comando `save nomearq` guarda o espaço de trabalho no arquivo *nomearq*, e o comando `load nomearq` carrega um espaço de trabalho previamente guardado com o comando `save`.

Em Matlab há dois terminadores de comando: a vírgula, `,` , e o ponto-e-vírgula, `;` . O resultado de um comando terminado por vírgula é ecoado para a tela. O terminador ponto-e-vírgula não causa eco. Resultados ecoados na tela são atribuídos a variável do sistema `ans` (de answer, ou resposta). O terminador vírgula pode ser suprimido no último comando da linha. Para continuar um comando na linha seguinte devemos terminar a linha corrente com três pontos, `...` , o símbolo de continuação. O sinal de porcento, `%` , indica que o resto da linha é comentário.

Matrizes

O tipo básico do Matlab é uma matriz de números complexos. Uma matriz real é uma matriz que tem a parte imaginária de todos os seus elementos nula. Um vetor linha é uma matriz $1 \times n$, um vetor coluna uma matriz $n \times 1$, e um escalar uma matriz 1×1.

As atribuições
`A = [1, 2, 3; 4, 5, 6; 7, 8, 9];` ou
`A = [1 2 3; 4 5 6; 7 8 9];` ,
são equivalentes, e atribuem à variável A o valor

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Matrizes são delimitadas por colchetes, elementos de uma mesma linha são separados por vírgulas (ou apenas por espaços em branco), e linhas são separadas por ponto-e-vírgula (ou pelo caracter nova-linha). O apóstrofe, `'` , transpõem uma matriz. É fácil em Matlab compor matrizes blocadas, desde que os blocos tenham dimensões compatíveis! Por exemplo:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \;,\; B = \begin{bmatrix} 5 \\ 6 \end{bmatrix} \;,\; C = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \;,\; \texttt{D=[A,B;C'];} \;\; D = \begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Cuidado ao concatenar matrizes com os espacos em branco, pois estes são equivalentes a vírgulas, separando elementos. Assim: `[1,2+3]==[1 5]` mas `[1,2 +3]==[1,2,3]`.

Há várias funções para gerar matrizes e vetores especiais: `zeros(m,n)`, `ones(m,n)` e `rand(m,n)` são matrizes de dimensão $m \times n$ com, respectivamente, zeros, uns, e números aleatórios em [0,1]. O vetor `i:k:j` é o vetor linha $[i, i+k, i+2k, \ldots i+nk]$, onde $n = \max m \mid i+mk \leq j$. Podemos suprimir o "passo" $k=1$, escrevendo o vetor `i:1:j` simplesmente como `i:j`. Se v é um vetor, `diag(v)` é a matriz diagonal com diagonal v, se A é uma matriz quadrada, `diag(A)` é o vetor com os elementos da diagonal principal de A. A matriz identidade de ordem n é `eye(n)`, o mesmo que `diag(ones(1,n))`.

`A(i,j)` é o elemento na i-ésima linha, j-ésima coluna de A, $m \times n$. Se vi e vj são vetores de índices em A, i.e. vetores linha com elementos inteiros positivos, em vi não excedendo m, e em vj não excedendo n, `A(vi,vj)` é a sub-matriz do elementos de A com índice de linha em vi e índice de coluna em vj. Em particular `A(1:m,j)`, ou `A(:,j)` é a j-ésima coluna de de A, e `A(i,1:j)`, ou `A(i,:)`, é a i-ésima linha de A. Exemplo:

$$A = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix} \quad vi = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \quad vj = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad A(vi,vj) = \begin{bmatrix} 32 & 33 \\ 12 & 13 \end{bmatrix}$$

As operações de adição, subtração, produto, divisão e potência, `+ - * / ^`, são as usuais da álgebra linear. O operador de divisão à esquerda, `\`, fornece em `x = A\b;` uma solução $x \mid A*x = b$. O operador de divisão a direita, `/` , fornece em `a = b/A;` uma solução $x \mid x*A = b$. Quando o sistema é bem determinado isto é o mesmo que, respectivamente, `inv(A)*b` ou `b*inv(A)`. (Quando o sistema é super-determinado x é uma solução no sentido dos mínimos quadrados.) Os operadores aritméticos de produto, potência e divisão tem a versão pontual, `.* .^ ./ .\` , que são executados elemento a elemento. Exemplo:

$$x = \begin{bmatrix} 5 & 5 & 5 \end{bmatrix}, \quad y = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad x.*y = \begin{bmatrix} -5 & 0 & 5 \end{bmatrix}$$

Matlab é uma linguagem declarativa (em oposição a imperativa) onde as variáveis são declaradas, dimensionadas e redimensionadas dinamicamente pelo interpretador. Assim, se A presentemente não existe, `A=11;` declara e inicializa uma matriz real 1×1. Em seguida, o comando `A(2,3)=23;` redimensionaria A como a matriz 2×3 `[11, 0, 0; 0, 0, 23]`. A nome da matriz nula é `[ ]`. A atribuição `A(:,2)=[];` anularia a segunda coluna de A, tornado-a a matriz 2×2 `[11, 0; 0, 23]`.

Controle de Fluxo

Os operadores relacionais da linguagem são `< <= > >= == ~=`, que retornam valor 0 ou 1 conforme a condição seja verdadeira ou falsa. Os operadores lógicos, *não e ou*, são, respectivamente, `~ & |` .

Matlab possui os comandos de fluxo *for – end*, *while – end* e *if – elseif – else – end*, que tem a mesma sintaxe do Fortran, exemplificada a seguir. Lembre-se de **não** escrever a palavra `elseif` como duas palavras separadas.

```
for i=1:n
  for j=1:m
    H(i,j)=1/(i+j-1);
  end
end
```

```
if(x<0)
  signx=-1;
elseif(x>0)
  signx=1;
else
  signx=0;
end
```

```
fatn=1;
while(n>1)
  fatn=fatn*n;
  n=n-1;
end
```

Uma consideração sobre eficiência: Matlab é uma linguagem interpretada, e tempo de interpretação de um comando simples pode ser bem maior que seu tempo de execução. Para tornar o programa rápido, tente operar sobre matrizes ou blocos, evitando loops explícitos que operem numa matriz elemento por elemento. Em outras palavras, tente evitar loops que repetem um comando que atribui valores a elementos, por atribuições a vetores ou matrizes. As facilidades no uso de vetores de índices, os operadores aritméticos pontuais, e funções como `max` , `min` , `sort` , etc., tornam esta tarefa fácil, e os programas bem mais curtos e legíveis.

Scripts e Funções

O fluxo do programa também é desviado pela invocação de subrotinas. A subrotina de nome *nomsubr* deve estar guardada no arquivo `nomsubr.m`; por isto subrotinas são também denominadas M-arquivos (M-files). Há dois tipos de subrotinas: Scripts e Funções.

Um script é simplesmente uma seqüência de comandos, que serão executados como se fossem digitados ao prompt do matlab. Subrotinas podem invocar outras subrotinas, inclusive recursivamente.

Um M-arquivo de função em Matlab começa com a declaração da forma

```
[ps1, ps2,... psm] = nomefunc( pe1, pe2,... pen )
```

A lista entre parênteses é a lista de parâmetros de entrada da função, e a lista entre colchetes são os parâmetros de saída. Parâmetros são variáveis locais, assim como são variáveis locais todas as variáveis no corpo da função.

Ao invocarmos a função *nomefunc* com o comando

```
[as1,... asm] = nomefunc(ae1,... aen);
```

Os argumentos de entrada, $ae1, \ldots aen$, são passados por valor aos (i.e. copiados nos) parâmetros de entrada, e ao fim do M-arquivo, ou ao encontrar o comando `return` , os parâmetros de saída são passados aos argumentos de saída. Exemplos:

```
function [mabel, mabin] = vmax(v)
% procura o maior elemento, em valor absoluto,
% dentro de um vetor, linha ou coluna.
%Obs: Esta funcao NAO segue os conselhos
```

```
%para operar sobre blocos, e nao elementos!

[m,n]=size(v);
if( m ~= 1  &  n ~= 1 )
  erro;
else
  K=max([m,n]);
  mabel= abs(v(1)); mabin= 1;
  for k=2:K
    if( abs(v(k)) > mabel )
      mabel= abs(v(k)); mabin= k;
    end%if
  end%for
end%else
```

Para referir-mo-nos, dentro de uma função, a uma variável externa, esta deve ter sido declarada uma variável global com o comando `global`. A forma de especificar uma variável como global muda um pouco de interpretador para interpretador, e mesmo de versão para versão: Diga `help global` para saber os detalhes de como funciona a sua implementação.

Bibliografia

[Colema-88] T.F.Coleman and C.F.van Loan. *A Matrix Computation Handbook.* SIAM Publications, Philadelphia.

[Moler-81] C.B.Moler. *Matlab Manual.* Department of Computer Science, University of New Mexico.

[Dongar-79] J.J.Dongarra, J.R.Bunch, C.B.Moler, G.W.Stewart. *LINPACK Users' Guide.* Society for Industrial and Applied Mathematics, Philadelphia.

[Smith-76] B.T.Smith, J.M.Boyle, J.J.Dongarra, B.S.Garbow, Y.Ikebe, V.C.Klema, C.B.Moler. *Matrix Eigensystem Routines: EISPACK Guide.* Lecture Notes in Computer Science, volume 6, second edition, Springer-Verlag.

[Garbow-77] B.S.Garbow, J.M.Boyle, J.J.Dongarra, C.B.Moler. *Matrix Eigensystem Routines: EISPACK Guide Extension.* Lecture Notes in Computer Science, volume 51, Springer-Verlag.

[Moler-92] C.B.Moler, J.N.Litte and S.Bangert. *PC-MATLAB User's Guide.* The MathWorks Inc. Sherborn, Massachusetts.

[Eaton-92] J.W.Eaton. *Octave Manual.* Chemical Engineering Department, University of Texas at Austin. Austin, Texas.