Capítulo 6 - Vetores multidimensionais

Vetores multidimensionais

- O **MATLAB** suporta vetores de dimensões arbitrárias.
- A terceira dimensão é denominada de *página (page em inglês)*. Dimensões de ordens superiores não possuem nomes específicos.
- Não há limite para o número de dimensões, mas trabalharemos com vetores tridimensionais porque são mais fáceis de visualizar. As idéias são extensivas a vetores de dimensões maiores.
- Considerando as operações elemento a elemento que vimos antes há as seguintes considerações a serem feitas:
  - Aritmética de vetor e escalar permanece inalterada.
  - Aritmética de vetor e vetor permanece inalterada desde que *todas* as dimensões correspondentes nos operandos tenham o mesmo tamanho.

Construção

- Seguindo a filosofia do **MATLAB** há várias formas de se construir vetores multidimensionais. Mostraremos estas formas por meio de exemplos:

```
>> A = zeros(4,3,2)
A(:,:,1) =
     0     0     0
     0     0     0
     0     0     0
     0     0     0

A(:,:,2) =
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

- Vetor com 4 linhas, 3 colunas e 2 páginas.
- A exibição é feita por páginas.
- As funções *ones*, *rand* funcionam da mesma forma que com vetores bidimensionais, basta acrescentar um parâmetro referente ao número de páginas.

Construção

- Podemos construir um vetor multidimensional partindo de dimensões menores e acresentando novas dimensões conforme os exemplos abaixo.

```
>> A = zeros(2,3)
A =
     0     0     0
     0     0     0

>> A(:,:,2) = ones(2,3)
A(:,:,1) =
     0     0     0
     0     0     0
A(:,:,2) =
     1     1     1
     1     1     1
```

```
>> A = [0 0 0]
A =
     0     0     0

>> A(2,:) = 1
A =
     0     0     0
     1     1     1

>> A(:,:,2) = 2
A(:,:,1) =
     0     0     0
     1     1     1
A(:,:,2) =
     2     2     2
     2     2     2
```

Construção - *reshape*

- A função *reshape* também pode ser usada para construir vetores n-dimensionais. Considere os exemplos a seguir:

```
>> A = zeros(2,3); A(:,:,2)= ones(2,3); A(:,:,3)=4;
>> B = reshape(A,2,9)
% equivalente: B = [A(:,:,1) A(:,:,2) A(:,:,3)];
B =
  0     0     0     1     1     1     4     4     4
  0     0     0     1     1     1     4     4     4

>> reshape(B,2,3,3) %equiv.: reshape(B,[2 3 3]);
ans(:,:,1) =
     0     0     0
     0     0     0
ans(:,:,2) =
     1     1     1
     1     1     1
ans(:,:,3) =
     4     4     4
     4     4     4
```

Construção - *reshape*

```
>> A = zeros(2,4); A(:,:,2)= ones(2,4); A(:,:,3)=4
A(:,:,1) =                    A(:,:,2) =
  0     0     0     0         1     1     1     1
  0     0     0     0         1     1     1     1
          A(:,:,3) =
                4     4     4     4
                4     4     4     4

>> reshape(A,[3 2 4])
ans(:,:,1) =                ans(:,:,2) =
     0     0                     0     1
     0     0                     0     1
     0     0                     1     1

ans(:,:,3) =                ans(:,:,4) =
     1     1                     4     4
     1     4                     4     4
     1     4                     4     4
```

Construção - *repmat*

- A função *repmat* também pode ser usada como auxiliar para construção de vetores multidimensionais. Exemplo:

```
>> C = ones(2,3);

>> repmat(C,[1 1 3])
ans(:,:,1) =
     1     1     1
     1     1     1
ans(:,:,2) =
     1     1     1
     1     1     1
ans(:,:,3) =
     1     1     1
     1     1     1

>> repmat(C,1,1,3)
??? Error using ==> repmat
Too many input arguments.
```

Construção - *cat*

- A função *cat* cria vetores a partir de dimensões menores.

```
>> A=zeros(2); B=ones(2); C=repmat(2,2,2);

>> D = cat(3,A,B,C) % conCATena A,B,C em 3 dim.
D(:,:,1) =       D(:,:,2) =       D(:,:,3) =
     0   0             1   1             2   2
     0   0             1   1             2   2

>> D = cat(4,A,B,C) % conCATena A,B,C em 4 dim.
D(:,:,1,1) =     D(:,:,1,2) =     D(:,:,1,3) =
     0   0             1   1             2   2
     0   0             1   1             2   2

>> size(D)
ans =
     2   2   1   3
```

- Note que A, B e C são matrizes bi-dimensionais. No segundo uso do *cat*, o **MATLAB** interpretou que a terceira dimensão deveria ficar unitária.

Funções úteis

- $squeeze(A)$: Elimina as dimensões unitárias, isto é, de tamanho um. Exemplo:

```
>> D %relembrando D
D(:,:,1,1) =    D(:,:,1,2) =    D(:,:,1,3) =
     0   0           1   1           2   2
     0   0           1   1           2   2
>> size(D)
ans =
     2   2   1   3

>> D = squeeze(D)
D(:,:,1) =    D(:,:,2) =    D(:,:,3) =
     0   0           1   1           2   2
     0   0           1   1           2   2
>> size(D)
ans =
     2     2     3
```

Funções úteis

- $sub2ind(size(A), i, j, k)$: Retorna o índice do elemento $A(i, j, k)$. A ordem de contagem é $i \rightarrow j \rightarrow k$.
- $[r, c, p] = ind2sub(size(A), ind)$: É o inverso da função anterior.

```
>> F = cat(3,2+zeros(2,4),ones(2,4),zeros(2,4))
F(:,:,1) =    2     2     2     2
              2     2     2     2

F(:,:,2) =    1     1     1     1
              1     1     1     1

F(:,:,3) =    0     0     0     0
              0     0     0     0

>> sub2ind(size(F),1,2,3)
ans =  19

>> [r,c,p]=ind2sub(size(F),19)
r =  1     c =  2     p =  3
```

Funções úteis

- $flidim(A, i)$: "intercambia elementos ao longo da i-ésima dimensão"

```
M = reshape(1:32,2,4,4);
M(:,:,1) =                          M(:,:,2) =
  1     3     5     7                 9    11    13    15
  2     4     6     8                10    12    14    16
M(:,:,3) =                             M(:,:,4) =
  17    19    21    23                25    27    29    31
  18    20    22    24                26    28    30    32


>> flipdim(M,2)
M(:,:,1) =                          M(:,:,2) =
  7     5     3     1                15    13    11     9
  8     6     4     2                16    14    12    10
M(:,:,3) =                          M(:,:,4) =
  23    21    19    17                31    29    27    25
  24    22    20    18                32    30    28    26
```

Funções úteis

- $shiftdim(A, i)$: Rotaciona as dimensões do vetor A, de acordo com o parâmetro i.
  - Se i for positivo, desloca no sentido anti-horário, i vezes. Por exemplo, suponha que A possua 3 dimensões. Então
    - $shiftdim(A, 1)$ fará com que o elemento $A(i, j, k)$ torne-se o elemento $A(j, k, i)$.
    - $shiftdim(A, 2)$ fará com que o elemento $A(i, j, k)$ torne-se o elemento $A(k, i, j)$.
  - Se i for negativo, cria i novas dimensões unitárias. Assim
    - $shiftdim(A, -1)$ fará com que o elemento $A(i, j, k)$ torne-se o elemento $A(1, i, j, k)$.

Funções úteis

- $permute(A, [ordem])$: permuta os elementos de A de acordo com a ordem especificada.

```
>> A = reshape(1:18,2,3,3);
>> permute(A,[2,3,1]) % a(j,k,i) <- a(i,j,k)
ans(:,:,1) =                    ans(:,:,2) =
     1     7    13                    2     8    14
     3     9    15                    4    10    16
     5    11    17                    6    12    18

>> permute(A,[4,1,2,3]) %a(1,i,j,k) <- a(i,j,k)
ans(:,:,1,1) =                  ans(:,:,2,1) =
     1     2                         3     4
ans(:,:,3,1) =                  ans(:,:,1,2) =
     5     6                         7     8
ans(:,:,2,2) =                  ans(:,:,3,2) =
     9    10                        11    12
ans(:,:,1,3) =                  ans(:,:,2,3) =
    13    14                        15    16
              ans(:,:,3,3) =
                   17    18
```

Funções úteis

- $ipermute(B, [ordem])$: é usada para desfazer as operações da função $permute$. Assim, se foi feito $B = permute(A, [ordem])$ então $A = ipermute(B, [ordem])$.

```
>> A = reshape(1:18,2,3,3);

>> permute(A,[2,3,1]) % a(j,k,i) <- a(i,j,k)
ans(:,:,1) =                    ans(:,:,2) =
     1     7    13                    2     8    14
     3     9    15                    4    10    16
     5    11    17                    6    12    18

>> ipermute(ans,[3,2,1])
ans(:,:,1) =                    ans(:,:,2) =
     1     3     5                    7     9    11
     2     4     6                    8    10    12
               ans(:,:,3) =
                   13    15    17
                   14    16    18
```