2007 COBENGE
XXXV Congresso Brasileiro de Educação em Engenharia

ENSINO DO MANUSEIO DE VETORES E MATRIZES POR MEIO DE FUNÇÕES

Edson Almeida Rego Barros – prof_edson@mackenzie.com.br
Lincoln Cesar Zamboni – lincoln.zamboni@mackenzie.com.br
Melanie Lerner Grinkraut – mlgrinkraut@mackenzie.com.br
Osvaldo Ramos Tsan Hu – oshu@yahoo.com
Sergio Vicente Denser Pamboukian – sergiop@mackenzie.com.br
Universidade Presbiteriana Mackenzie, Escola de Engenharia
Rua da Consolação, 930 – Prédio 6
01302-907 – São Paulo – SP

Resumo: *Este trabalho detalha a metodologia empregada pelos professores da Escola de Engenharia da Universidade Presbiteriana Mackenzie (EEUPM) no ensino das técnicas de programação, em linguagem C/C++ e o software MatLab, para manuseio de estruturas de dados homogêneas, também conhecidas como vetores e/ou matrizes.*

Palavras-chave: *computação, programação, funções, matrizes, vetores.*

1 INTRODUÇÃO

O manuseio de uma massa de dados numéricos de grande volume, nos segmentos relacionados ao ensino de engenharia, normalmente envolve o entendimento e a compreensão das ferramentas matemáticas conhecidas como Vetores ou Matrizes. Analogamente, a operação informatizada destas ferramentas, demanda dos estudantes universitários um grau de domínio apropriado de linguagens de programação científicas.

O presente artigo tem o objetivo de ilustrar etapas que são abordadas no ensino de tais ferramentas nos cursos de engenharia.

2 BREVE HISTÓRICO

O grupo de professores que ministra disciplinas relacionadas à Computação e Programação na Escola de Engenharia da Universidade Presbiteriana Mackenzie (EEUPM) tem desenvolvido um trabalho ao mesmo tempo inovador e tradicional, conforme já foi relatado em BARROS *et al* (2004).

Inovador no sentido que sempre se procurou adaptar os conteúdos pedagógicos das disciplinas às tendências de mercado nos segmentos de hardware e software. Em um prazo de aproximadamente vinte anos, evoluiu-se de ambientes computacionais que usavam cartão perfurado, passando pelas diversas gerações de computadores, até a situação atual. Durante esse período os temas do curso foram sendo debatidos e ensinados, enquanto as linguagens empregadas nas disciplinas de computação e programação foram se alternando entre: FORTRAN (FARRER *et al*., 1992), BASIC (STEINBRUCH, 1981), Algorítmica (FARRER *et al*., 1999), Pascal (FARRER *et al*., 1999), Delphi (CANTÙ, 2005; BARROS *et al,* 2006),

C++ Builder (LEÃO, 1998; BARROS *et al,* 2003), C/C++ (DEITEL; DEITEL, 2007; ZAMBONI *et al,* 2006) e MatLab (CHAPMAN, 2006; MATHWORKS, 1997, 1999, 2001a, 2001b).

Tradicional quando se considera toda a fundamentação teórica sobre a qual o conteúdo pedagógico tem sido amparado e desenvolvido. Independente da época ou ambiente computacional, sempre foram ensinados os seguintes conteúdos: conceitos da lógica booleana, séries matemáticas, arranjos vetoriais e matriciais, cálculos complexos entre tantos outros temas relevantes pertinentes às disciplinas de computação.

3 AS ESTRUTURAS DE TRABALHO

No artigo (BARROS *et al*., 2006), apresentado no *World Congress on Computer Science, Engineering and Technology Education*, desenvolveu-se toda a contextualização do emprego de funções como ferramenta de ensino para estudantes de engenharia. Partindo-se da premissa que a função é a essência do pensamento lógico, procurou-se transformar o computador em uma forma de auxilio aos estudantes no sentido de descobrir o que há de fundamental no pensamento do engenheiro.

Com a mesma filosofia, no artigo (ZAMBONI *et al*., 2005), apresentado no XXXIII Congresso Brasileiro de Ensino de Engenharia, o foco do texto se voltou ao uso de classes de objetos para a programação dos problemas de engenharia. Afinal, nas classes são programados os métodos de cálculo, que na verdade são funções dentro das classes, ou seja, uma forma de serviço encapsulado.

4 ARMAZENAMENTO DE DADOS HOMOGÊNEOS

Um dos assuntos relevantes para os engenheiros refere-se ao domínio e manuseio de dados em forma de arranjos homogêneos. Quando um arranjo homogêneo possui uma dimensão é chamado de vetor, duas dimensões de matriz ou tabela, e três ou mais dimensões de matriz multidimensional, ou mesmo outro nome mais apropriado ao problema que venha a ser empregado (por exemplo, Volume, Base de dados, e outros).

A relevância do emprego de vetores ou matrizes para os alunos de engenharia é notória, no sentido que estes podem ser usados como ferramentas de cálculos em vários contextos complexos, tais como: cálculos estruturais, levantamentos estatísticos, estudos de origem-destino, distribuição de carregamentos, campos elétricos, elementos finitos, computação gráfica, processamento de imagens entre tantas outras possibilidades.

O conceito de vetores normalmente é aprendido indiretamente no Ensino Médio quando os professores de Matemática apresentam os conceitos de Progressão Aritmética (PA) e Progressão Geométrica (PG). Neste momento da aprendizagem torna-se necessário o manuseio de variáveis indexadas ($a_0, a_1, a_2, a_3, \ldots, a_i, , \ldots, a_n$), sendo também reforçado o conceito de Lei de Formação, conforme a "Equação (1)" e a "Equação (2)" que descrevem as leis de formação de uma PA e uma PG.

$$PA \rightarrow a_i = a_0 + \text{i.r} \tag{1}$$

$$PG \rightarrow a_i = a_0 \times q^i \tag{2}$$

Computacionalmente, um vetor é uma lista de valores do mesmo tipo (p.ex: números inteiros, números decimais, caracteres, etc), que na linguagem de programação C/C++ pode ser declarado conforme a sintaxe da "Equação (3)".

```
TipoDeDado NomeVetor [QtdPosições];
```
(3)

Como exemplo apresenta-se na "Equação (4)" a declaração de um vetor de números do tipo inteiro, com capacidade para armazenar até 100 (cem) valores.

```
int Vet [100];
```
(4)

Por sua vez, o MatLab é um software que foi desenvolvido para trabalhar com matrizes e vetores de maneira simples e natural, sem a necessidade de se declarar tipos e variáveis. Um vetor ou uma matriz é declarado no momento da entrada de dados. Assim um vetor A pode ser criado com o comando conforme a "Equação (5)".

```
A = [1 2 3 4 5 6]
```
(5)

Observa-se nesta expressão que não existe a necessidade da declaração da dimensão do vetor ou da matriz.

Matrizes, assim como vetores, são conceitos que o aluno também começa a se familiarizar no Ensino Médio. Na universidade, em cursos direcionados às áreas de Ciências Exatas, este tema poderá ser abordado em disciplinas como Cálculo Numérico (MIRSHAWKA, 1989; ZAMBONI; MONEZI JÚNIOR, 2002) ou Matemática Discreta. Não raro, em outras disciplinas, frequentemente o cálculo matricial é adotado como forma de solucionar problemas complexos de engenharia, que envolvem uma grande quantidade de dados e de operações matemáticas.

De um ponto de vista computacional, uma matriz é uma tabela de valores do mesmo tipo (p.ex: números inteiros, números decimais, caracteres, etc), que na linguagem de programação C/C++ pode ser declarada conforme a "Equação (6)".

```
TipoDeDado NomeMatriz [QtdLinhas] [QtdColunas];
```
(6)

Como exemplo apresenta-se na "Equação (7)" a declaração de uma matriz de números do tipo decimais (*float*), com capacidade para armazenar até 5 (cinco) linhas com até 10 (dez) colunas em cada linha, ou seja, 50 (cinqüenta) valores (5 linhas x 10 colunas).

```
float Mat [5][10];
```
(7)

No MatLab, para se declarar e preencher uma matriz de 2 (duas) linhas por 3 (três) colunas com valores de 1 a 6 emprega-se o comando conforme a "Equação (8)".

```
B = [1 2 3 ; 4 5 6]
```
(8)

É importante registrar que uma das facilidades do uso de um produto como o MatLab é a existência de uma série de funções destinadas à criação de matrizes especiais. Podem ser citadas algumas, entre elas: uma matriz nula preenchida com zeros conforme exemplificado na "Equação (9)", matrizes unitárias com seus valores todos iguais a um conforme indicado na "Equação (10)", a possibilidade de criação com dados aleatórios como apresentada pela "Equação (11)", além da possibilidade da criação de matrizes identidades (em matrizes quadradas, toda diagonal principal preenchida com um e o restante com zeros) conforme a "Equação (12)".

`C = zeros(2, 3)` → matriz de 2 linhas por 3 colunas preenchida por números zeros (9)

`D = ones(2, 3)` → matriz de 2 linhas por 3 colunas preenchida por números uns (10)

`E = rand(2, 3)` → matriz de números aleatórios (11)

`F = eye(4, 4)` → matriz (quadrada) identidade de ordem 4 (12)

Matrizes podem proporcionar o armazenamento ideal para dados representativos de uma superfície em mapas de contorno (curvas de nível), conforme pode ser observado na "Figura 1", uma vez que possibilitam estudos quanto à distribuição das informações geograficamente.

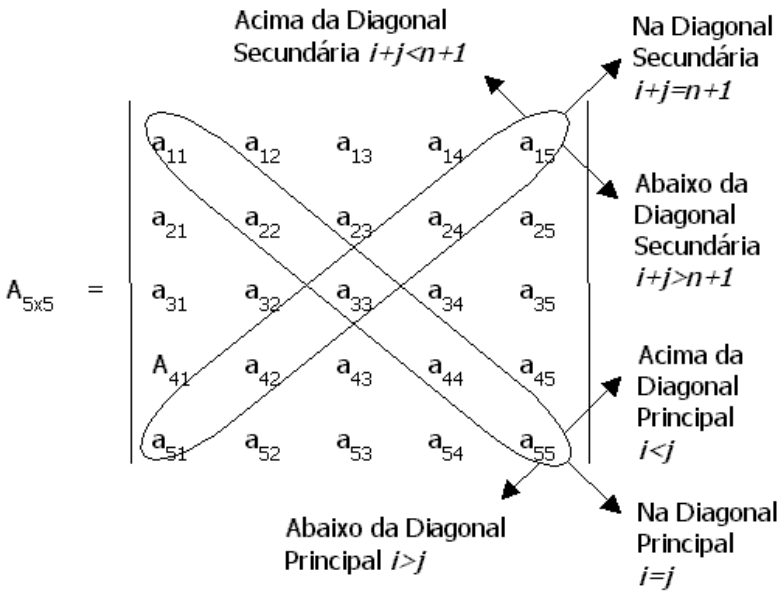

Figura 1 – Exemplo de matriz quadrada (MAC SYSTEM EDUCACIONAL, 2007).

Dados vetoriais ou matriciais podem ser vistos como variáveis em problemas de engenharia. Na disciplina "Software Aplicado a Engenharia" usa-se o *software* MatLab para se efetuar estudos onde as variáveis são vetores ou matrizes. Dessa forma, se torna necessário algum procedimento na linguagem de programação capaz de agrupar uma massa de dados como se fosse um único dado.

A facilidade de usar um tipo de dado capaz de armazenar vários valores, uma lista ou tabela de dados, proporciona ao programador (no caso ao aluno) a capacidade de desenvolver complexos raciocínios abstratos que servirão de suporte ao entendimento de temas avançados na engenharia.

Computacionalmente, na linguagem C++, pode-se criar um tipo de dado vetorial ou um tipo de dado matricial conforme ilustrado pela "Equação (13)" e "Equação (14)".

```
typedef TipoDeDado NomeTipoVetor [QtdPosições];          (13)
```

```
typedef TipoDeDado NomeTipoMatriz [QtdLinhas][QtdColunas];  (14)
```

Observa-se que:

- vetores e matrizes podem ser declarados como estáticos ou dinâmicos. Para efeito da metodologia de ensino, como descrito neste artigo, só se apresentarão vetores e matrizes **estáticas**, ou seja, dados homogêneos declarados em linhas de programação (não em tempo de execução);

- O dimensionamento em texto programa necessita de um valor máximo que pode ser definido pela criação de uma constante, conforme exemplificado pela "Equação (15)".

```
const int MAX = 100;
```
(15)

Dessa forma, ilustrativamente, a "Equação (4)" e a "Equação (7)" poderiam ter sido feitas da forma apresentada pela "Equação (16)".

```
const int MAX = 100;
const int LIN = 5;
const int COL = 10;
typedef int Vetor [MAX];
typedef float Matriz [LIN][COL];
Vetor Vet;
Matriz Mat;
```
(16)

Em essência tanto a "Equação (4)" e a "Equação (7)", quanto às descritas pela "Equação (16)" resultam exatamente na mesma ocupação de memória, porém a programação no último formato é mais legível e pode proporcionar melhor uso do código programa.

5 FUNÇÕES COM PARÂMETROS VETORIAIS E MATRICIAIS

Assim sendo, tornou-se importante nas disciplinas de programação, trabalhos com funções que são dotadas de parâmetros de entrada e/ou saída vetoriais ou matriciais, conforme a "Equação (17)".

$C = f(A,B)$, onde A, B e C são vetores e/ou matrizes (17)

Na linguagem C++, não se pode atribuir resultado direto em vetores ou matrizes. A atribuição tem que ser por passagem de parâmetro, ou seja, a expressão matemática dada pela "Equação (17)" pode ser escrita conforme a "Equação (18)"

```
void Funcao(Vetor &A, Vetor &B, Vetor &C);
```
(18)

Na "Equação (18)", A e B podem ser vetores de entrada de dados, enquanto C representa a saída de dados.

Em funções, a passagem de vetores e matrizes como parâmetros deve ser feita por referência (símbolo &), pois a declaração, tanto de vetor quanto de matriz, se refere a um endereço de memória (ponteiro).

O fluxo natural de um problema computacional é Entrada → Processamento → Saída, conforme ilustrado na "Figura 2".

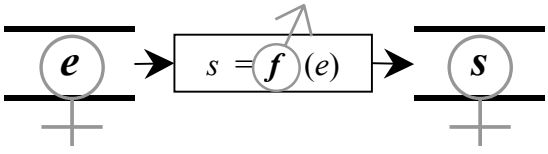

FIGURA 2 – Visão algorítmica de uma função.

No que tange ao MatLab as funções são criadas em arquivos separados, denominados arquivos M de funções. Estas funções podem ser definidas para utilizarem parâmetros ou escalares ou matriciais, e retornam valores ou escalares ou matriciais. Sua declaração é efetuada utilizando-se a palavra *function* na primeira linha do arquivo M, conforme ilustrado pela "Equação (19)", sendo que a e b podem ser escalares ou matriciais:

```
Function[b]= NomeDaFuncao(a)
```
(19)

6 METODOLOGIA

A metodologia descrita neste artigo procura tornar mais fácil, e menos árida, a compreensão de exercícios com vetores e matrizes. Um fato importante é que todo programa que envolve arranjos homogêneos pode ser dividido em quatro etapas:

- Dimensionamento do arranjo (Vetor ou Matriz);
- Preenchimento do arranjo com dados;
- Cálculo propriamente dito com a massa de dados
- Exibição dos resultados.

Sobre o Dimensionamento, pode-se afirmar que, na maioria das linguagens de programação que utiliza dados indexados, há a necessidade de algumas definições, tais como: quantas dimensões de índices serão necessárias (p.ex: uma, duas, três ou mais?) ou quantos elementos farão parte da massa de dados (p.ex: quantas linhas por quantas colunas?). Esta etapa corresponde a "Equação (4)", "Equação (7)" e/ou "Equação (16)".

Quanto ao preenchimento dos dados, este pode ocorrer por: cadastramento direto em programa, conforme "Equação (20)"; por atribuição de fórmula (seja função matemática ou randômica), como ilustrado pela "Equação (21)" e "Equação (22)"; ou ainda por cadastramento direto, como apresentado pela "Equação (23)" e "Equação (24)".

```
typedef int Matriz[3][3];
Matriz MatInt = {{12, 4, 66},
                {16,  5, 67},
                {22, 21, 19}};
```
(20)

```
typedef double MatrizDec[MAX][MAX];
void Calcular(MatrizDec &m, int l, int c)
{
  for (int i = 0; i < l; i++)
    for (int j = 0; j < c; j++)
      m[i][j] = sin(i) + cos(j);
}
```
(21)

```
typedef int Matriz [MAX][MAX];
void sortear(Matriz &m, int l, int c, int Menor,
  int Maior)
{srand(time(0));
  for (int i = 0; i < l; i++)
    for (int j = 0; j < c; j++)
      m[i][j] = rand() % (Maior - Menor + 1)+
        Menor;
}
```
(22)

```
typedef double VetorDec[MAX];
int LerVetor(VetorDec &a)
{
  int n;
  do{
    cout << "Quantidade de itens?";
    cin >> n;
  }while (n < MIN || n > MAX);
  for(int i = 0; i < n; i++){
    cout << "Valor[" << i << "]? ";
    cin >> a[i];
  }
  return n;
}
```
(23)

Na "Equação (23)" pergunta-se o valor da quantidade de posições necessárias ao vetor e os dados que se pretende cadastrar. Ao término da execução da função o tamanho do vetor é retornado por valor e o vetor por referência.

```
typedef double MatrizDec[MAX][MAX];
void LerMatriz(MatrizDec &a, int &l, int &c)
{
  do{cout << "Quantidade de linhas?";
     cin  >> l;
  }while (l < MIN || l > MAX);
  do{cout << "Quantidade de colunas?";
     cin >> c;
  }while (c < MIN || c > MAX);
  for(int i = 0; i < l; i++)
    for(int j = 0; j < c; j++){
      cout << "Valor[" << i<< "," << j << "]? ";
      cin >> a[i][j];
}}
```
(24)

Na "Equação (24)" pergunta-se a quantidade de linhas e a quantidade de colunas necessárias para a matriz, seguido dos dados que se pretende cadastrar. Ao término da execução da função todos os dados da matriz são devolvidos por referência.

Para os procedimentos vistos na "Equação (20)", "Equação (21)" e "Equação (22)" as funções equivalentes para o MatLab correspondem respectivamente aos fragmentos de programa descritos na "Equação (25)", "Equação (26)" e "Equação (27)".

```
MatInt = [12 4 66; 16 5 67; 22 21 19]
```
(25)

```
function [m] = Calcular (l, c)
for i=1:l
  for j=1:c
    m(i,j) = sin(i-1) + cos(j-1)
  end
end
```
(26)

```
function [m] = Sortear (l, c, menor, maior)
for i=1:l
  for j=1:c (27)                                          (27)
    m(i,j) = Rem(rand,(maior-menor+1) + menor)
  end
end
```

Os procedimentos descritos na “Equação (23)” e “Equação (24)” são entradas de dados, que no MatLab podem ser obtidos diretamente ou na construção da matriz ou por uma função como apresentada na “Equação (28)”.

```
function [m] = Entrar (l, c)
for i=1:l
  for j=1:c                                               (28)
    m(i,j) = input('digite um numero:');
  end
end
```

Por Cálculo, propriamente dito, pode-se entender qualquer manuseio de dados que proporcione algum resultado relevante. Como exemplo pode-se citar: na “Equação (29)” uma função para calcular a soma dos itens de um vetor; na “Equação (30)” o produto de um escalar por um vetor; na “Equação (31)” uma rotina para atribuir zeros em todas as posições da matriz; na “Equação (32)” a soma de duas matrizes.

```
int soma(Vetor &v, int n)
{ int s = 0;
  for (int i = 0; i < n; i++)                             (29)
    s += v[i];
  return s;
}
```

```
void EscVet(double x, Vetor &ent, Vetor &sai,
  int n)
{for(int i = 0; i < n; i++)                               (30)
  sai[i] = x * ent[i];
}
```

```
void ZeraMatriz(Matriz &ent, int l ,int c)
{ for (int i = 0; i < l; i++)
    for (int j = 0; j < c; j++)                           (31)
      ent[i][j]=0.0;
}
```

```
void SomaMatriz(MatrizDec &ent1,
                MatrizDec &ent2,
                MatrizDec &sai,
                int l ,int c)                             (32)
{ for (int i = 0; i < l; i++)
    for (int j = 0; j < c; j++)
      sai[i][j] = ent1[i][j] + ent2[i][j];
}
```

No MatLab a manipulação e operação de matrizes ocorre de forma simples e direta. O procedimento, da "Equação (32)", soma de matrizes, supondo-se A e B duas matrizes de mesma dimensão, resume-se a expressão expressa na "Equação (33)".

```
C = A + B                                                    (33)
```

Analogamente, assegurando-se as compatibilidades matemáticas, no MatLab são simples os cálculos como: subtração de matrizes, exemplificado pela "Equação (34)"; produto matricial, ilustrado na "Equação (35)"; divisão matricial (produto de uma matriz pela inversa de outra → $A.B^{-1}$), "Equação (36)"; cálculo do determinante, "Equação (37)"; cálculo da matriz inversa, "Equação (38)"; entre varias outras possibilidades incluindo até o uso de matrizes de dados complexos.

```
D = A - B                                                    (34)

E = A * B                                                    (35)

F = A / B                                                    (36)

G = det(A)                                                   (37)

H = inv(A)                                                   (38)
```

Para a Exibição dos resultados, quarta e última etapa da metodologia, faz-se necessário algum procedimento ou função. Se a resposta for um vetor, seu conteúdo será exibido de forma análoga ao apresentado em "Equação (39)".

Caso a resposta seja uma matriz, a forma de exibir o seu conteúdo está indicada na "Equação (40)".

```
void ExibeVetor(VetorDec &a, int n)
{for(int i=0;i<n;i++)                                        (39)
   cout<<"Valor["<<i<<"]= "<<a[i]<<endl;
}

void ExibeMatriz(MatrizDec &a, int l, int c)
{for (int i=0;i<l;i++)
   for (int j=0;j<c;j++)                                     (40)
      cout << "Valor[" << i << "," << j <<
              "]=" << a[i][j] << endl;
}
```

No Matlab não é necessário construir função especifica para a exibição em tela do vetor ou da matriz. Digitando-se o nome da variável na Tela de Comando (*Command Window*) o conteúdo da variável, independente de ser vetor ou matriz, será exibido no formato de linhas e colunas, conforme "Equação (41)".

```
>> B = [1 2 3 ; 4 5 6]; [enter]
>> B [enter]                                                 (41)
B =       1   2   3
          4   5   6
```

7 VANTAGENS E DESVANTAGENS DE CADA PROGRAMA

As vantagens e desvantagens de cada programa podem ser resumidas na tabela 1.

TABELA 1 – Quadro de vantagens e desvantagens de cada programa

	MatLab	**C++**
Custo do programa	↓ - Alto Custo	↑ - Gratuito ou baixo custo
Velocidade de aprendizado	↑ - Rápido	↓ - Lento
Facilidade de programação	↑ - Requer menor formalismo	↓ - Requer maior formalismo
Velocidade de programação	↑ - Rápido	↓ - Lento
Funções pré-programadas	↑ - Incluídas no software	↓ - Existem bibliotecas na internet
Velocidade de execução	↓ - Lento	↑ - Rápido
Consumo de memória	↓ - Maior consumo	↑ - Menor consumo
Precisão de calculo	↑ - Alta precisão	↑ - Alta precisão
Requisitos de processamento	↓ - Equipamentos modernos Muita memória	↑ - Qualquer equipamento

Em termos gerais os programas se equivalem para o fim desejado. O critério de escolha irá recair em função do orçamento da escola para investimentos em infra-estrutura e da disponibilidade de horas/aula.

8 CONCLUSÕES

Vetores e Matrizes representam apenas parte dos problemas computacionais existentes nas diversas áreas científicas da Engenharia, como os sistemas para manipulação de elementos finitos. Problemas dessa natureza podem ser propostos, estudados e aplicados em pesquisas construtivas e direcionadas para os mais variados segmentos tecnológicos.

Com esta visão, alguns dos professores da EE/UPM, das disciplinas de Computação e Programação, têm desenvolvido diversas atividades de ensino buscando proporcionar ao corpo discente uma melhor compreensão da arte de programar, como a proposta apresentada neste artigo.

Esses profissionais fundaram, neste ano de 2007, o Grupo de Computação para Engenharias, certificado pela instituição junto ao CNPQ, para explorar temas relacionados à computação cientifica, com as quais o engenheiro necessita para o seu pleno e completo exercício profissional.

O Grupo espera que o exposto no presente trabalho venha colaborar com outros profissionais da área em outras instituições de ensino e pesquisa.

9 REFERÊNCIAS BIBLIOGRÁFICAS

BARROS, E. A. R.; PAMBOUKIAN, S. V. D.; ZAMBONI, L. C. **Ensino de Computação para estudantes de Engenharia**. COBENGE - Congresso Brasileiro de Ensino de Engenharia. Anais. 2004. Disponível em: <http://meusite.mackenzie.com.br/edsonbarros/ publicacoes/ BARROS_Edson_Ensino_de_Computacao.pdf> Acesso em: 23 mai. 2007.

BARROS, E. A. R.; PAMBOUKIAN, S. V. D.; ZAMBONI, L. C. **Ensinando Programação Através de Funções**. WCCSETE - World Congress on Computer Science, Engineering and Technology Education. Anais. 2006. Disponível em: <http://meusite.mackenzie.com.br/edsonbarros/publicações/Artigo_658_WCCSETE2006.doc > Acesso em: 23 mai. 2007.

BARROS, E. A. R.; GRINKRAUT, M. L.; PAMBOUKIAN, S. V. D.; ZAMBONI, L. C.. **Delphi para Universitários**. Edição Compacta. São Paulo: Páginas & Letras, 3 ed., 2006. 200 p.

BARROS, E. A. R.; ZAMBONI, L. C.; PAMBOUKIAN, S. V. D.. **C++ Builder para Universitários**. São Paulo: Páginas & Letras, 2 ed., 2003. 560 p.

CANTÚ, M. **Dominando o Delphi 7 – a Biblía**. Makron, 2005.

CHAPMAN, Stephen J. **Programação em MATLAB para Engenheiros**. Thomson. 2006.

DEITEL, P. J.; DEITEL, H. M. **C How to program**. Upper Saddle River : Pearson Prentice Hall, 2007. 1130 p.

FARRER, H; BECKER, C. G.; FARIA, E. C.; CAMPOS FILHO, F. F.; DE MATOS, H. F.; DOS SANTOS, M. A.; MAIA, M. L. **Pascal Estruturado**. Rio de Janeiro: LTC - Livros Técnicos e Científicos, 1999. 279 p.

FARRER, H; BECKER, C. G.; FARIA, E. C.; CAMPOS FILHO, F. F.; DE MATOS, H. F.; DOS SANTOS, M. A.; MAIA, M. L. **Fortran Estruturado**. Rio de Janeiro: LTC - Livros Técnicos e Científicos, 1992. 194 p.

FARRER, H; BECKER, C. G.; FARIA, E. C.; CAMPOS FILHO, F. F.; DE MATOS, H. F.; DOS SANTOS, M. A.; MAIA, M. L. **Algoritmos Estruturados**. Rio de Janeiro: LTC - Livros Técnicos e Científicos, 1999. 284 p.

LEÃO, M. **Introdução ao C++ Builder**. Rio de Janeiro: Axcel Books do Brasil, 1998. 165 p.

MAC SYSTEM EDUCACIONAL. **Matriz Quadrada**. Disponível em: <http://www. macsystemeduc.com.br/livros/ex12/ exemplo12.html> Acesso em: 23 mai. 2007.

MATHWORKS. **MATLAB – The language of technical computing**. **Language Reference Manual. Ver.5**. The MathWorks, Inc. Natick, MA, USA. 1997.

MATHWORKS. **MATLAB – The language of technical computing**. **Learning MATLAB. Ver.5.3 (release 11)**. The MathWorks, Inc. Natick, MA, USA. 1999.

MATHWORKS. **MATLAB – The language of technical computing**. **Getting Started with MATLAB. Ver.6**. The MathWorks, Inc. Natick, MA, USA. 2001a.

MATHWORKS. **MATLAB – The language of technical computing**. **Using MATLAB. Ver.6**. The MathWorks, Inc. Natick, MA, USA. 2001b.

MIRSHAWKA, V. **Exercícios de Cálculo Numérico**. São Paulo: Nobel, 1989. 380 p.

STEINBRUCH, M. **Introdução à linguagem BASIC**. Porto Alegre: Editora da UFRGS, 1981.

SULLIVAN, M.; SULLIVAN, M., III. **My MATHLAB Algebra & Trigonometry**. LSU Edition. Prentice Hall. 2007.

ZAMBONI, L. C.; BARROS, E. A. R.; PAMBOUKIAN, S. V. D. **A Programação Orientada a Objetos como Ferramenta para o Aprendizado e Auxílio em Projetos de Engenharia**. COBENGE - Congresso Brasileiro de Ensino de Engenharia. Anais. 2005. Disponível em: <http://meusite.mackenzie.com.br/edsonbarros/publicacoes/ COBENGE_ 2005_Galerias.pdf> Acesso em: 23 mai. 2007.

ZAMBONI, L. C.; MONEZI JÚNIOR, O.. **Cálculo Numérico Para Universitários**. São Paulo: Páginas & Letras, 1.ed., 2002. 360 p.

ZAMBONI, L. C.; PAMBOUKIAN, S. V. D.; BARROS, E. A. R.. **C++ para Universitários**. São Paulo: Páginas & Letras, 2006. 402 p.

TEACHING OF THE HANDLING OF VECTORS AND MATRIXES THROUGH FUNCTIONS

Abstract: *This paper details the methodology used by the professors of Escola de Engenharia of Universidade Presbiteriana Mackenzie (EEUPM) in the teaching of programming techniques, with C/C++ language and MatLab software, for handling homogeneous data structures, also known as vectors and/or matrixes.*

Key-words: *computation, programming, functions, matrixes, vectors.*