# Utilização de Métodos Númericos para Encontrar Raizes

**Jhonattan C. B. Cabral**[1], **Daniel M. P. Carvalho**[1]

[1]Instituto de Informática e Matemática Aplicada (DIMAp)
Universidade Federal do Rio Grande do Norte (UFRN)

jhonattan.yoru@gmail.com, danielmarx08@gmail.com

***Abstract.*** *Task applied in the first unit of the discipline of Numerical Calculus (DIM0404), aims to improve and apply the knowledge acquired in linear system resolutions.*

***Resumo.*** *Tarefa aplicada na primeira unidade da disciplina de Cálculo Numérico (DIM0404), tem como objetivo aprimorar e aplicar os conhecimentos adquiridos em resoluções de sistemas lineares.*

## 1. Dado 3 matrizes quadradas de ordem N igual a 3, 5 e 88, respectivamente. Encontre a resolução do sistema linear abstraído dessas matrizes.

Para solucionar o problema, aplicamos o método de Gauss, onde tem como objetivo triangularizar a matriz, facilitando o encontro da solução do sistema. Como sabemos que o sistema de cada matriz é determinado, ou seja, possui uma única solução, não nos preocupamos em encontrar outras soluções.

### 1.1. Código

Vendo que modularizamos o projeto, mostraremos o código dos arquivos principais, ocultado implementações básicas do arquivo **main.cpp**.

#### 1.1.1. SistemaLinear.h

```
1  /*
2   * @file SistemaLinear.h
3   * @brief Universidade Federal do Rio Grande Do Norte (UFRN)
4   * @brief DIM0404 - Calculo Numerico
5   * @brief Tarefa: Encontrar raizes de equacoes
6   * @date 27/03/2018
7   * @author Jhonattan Cabral e Daniel Marx.
8   */
9
10 #ifndef _SISTEMALINEAR_H_
11 #define _SISTEMALINEAR_H_
12
13 #include <iostream>
14 #include <vector>
15 #include <string>
```

```
16 #include <fstream>
17
18 class SistemaLinear
19 {
20
21         using matriz = std::vector<std::vector<float>>;
22
23         private:
24
25                 matriz sistema_m;
26
27         public:
28
29                 SistemaLinear(std::string nome_arquivo);
30
31                 ~SistemaLinear() = default;
32
33                 void solucoesSistema();
34
35                 void imprimeMatrizSistema();
36
37         private:
38
39                 matriz gerarMatrizTriangular();
40                 matriz lerMatriz(std::string nome_arquivo);
41 };
42 #endif
```

Aqui como podemos ver no código acima, temos a declaração da nossa classe. Possuindo como atributo privado a matriz (passada através de aquivo pelo usuário) na qual será feito a eliminação de Gauss, tem também métodos públicos como **solucoesSistema()**, **imprimeMatrizSistema()**, além do construtor e destrutor, por fim temos os métodos privados que auxiliaram na resolução do projeto **gerarMatrizTriangular()** e o **lerMatriz(*fileName*)**.

### 1.1.2. SistemaLinear.cpp

```
1  /*
2   * @file SistemaLinear.cpp
3   * @brief Universidade Federal do Rio Grande Do Norte (UFRN)
4   * @brief DIM0404 - Calculo Numerico
5   * @brief Tarefa: Encontrar raizes de equacoes
6   * @date 27/03/2018
7   * @author Jhonattan Cabral e Daniel Marx.
8   */
9
10 #include "SistemaLinear.h"
11
12 using matriz = std::vector<std::vector<float>>;
13
```

```
14 /*
15  * Construtor
16  */
17 SistemaLinear::SistemaLinear(std::string nome_arquivo)
18 {
19         sistema_m = lerMatriz(nome_arquivo);
20 }
21
22 /*
23  *@brief Metodo responsavel por ler a matriz de um determinado
       arquivo
24  *@param nome_arquivo - nome do arquivo no qual sera lido
25  *@return A - matriz lida do arquivo
26  */
27 matriz SistemaLinear::lerMatriz(std::string nome_arquivo)
28 {
29         std::ifstream file(nome_arquivo);
30     unsigned size;
31
32     file >> size; //Le a primeira linha e pega o tamanho da
          matriz
33
34         if(!file){ std::cout << "Erro"; }
35
36     matriz A;
37     double value;
38     for(auto i = 0u; i < size; i++)
39     {
40             std::vector<float> temp;
41         for(auto j = 0u; j < size+1; j++)
42         {
43                 file >> value;
44             temp.push_back(value);
45         }
46         A.push_back(temp);
47         }
48         return A;
49 }
50
51 /*
52  *@brief Metodo responsavel por triangular a matriz atraves da
       eliminacao de Gauss
53  *@return A - matriz triangularizada
54  */
55 matriz SistemaLinear::gerarMatrizTriangular()
56 {
57         matriz A = sistema_m;
58
59         for(size_t k = 0u; k <= A.size(); k++) //Numero de pivos
```

```
60      {
61                      for(size_t i = 1 + k; i < A.size(); i++) //
                           Numero de linhas a baixo da primeira
62              {
63                      double fatorM = A[i][k]/A[k][k];
64                              for(size_t j = 0 + k ; j < A.size() + 1;
                                    j++) //Numero de colunas
65                      {
66                              A[i][j] = A[i][j] - A[k][j]*(fatorM);
67                      }
68                   }
69              }
70
71              return A;
72      }
73
74      /*
75       *@brief Metodo responsavel por resolver o sistema linear a
            partir de uma matriz triangular
76       */
77      void SistemaLinear::solucoesSistema()
78      {
79              matriz A = gerarMatrizTriangular();
80              std::vector<float> resultados;
81
82              for(int i = A.size() - 1; i >= 0; i = i - 1) // Calcula-
                   se de baixo para cima.
83      {
84                      //Valor da diagonal principal, que multiplica a
                           incognita.
85                      float X = A[i][i];
86
87                      //Armazenara o valor da incognita que esta sendo
                            buscada
88                      float Y = 0;
89
90              //Calculo do Y considera os valores posteriores ao da
                   diagonal principal
91              for(size_t j = i+1, k = 0; j < A.size() + 1; j++, k++)
92              {
93                      //Enquanto j nao chegar na posicao da ultima
                           coluna
94                      if(j < A.size())
95                              Y = Y + resultados[k] * A[i][j]; //Soma
                                   valores depois dos pivores, e os
                                   multiplica pelas incognitas ja
                                   achadas
96                          else
97                                   Y = (A[i][j] - Y)/X; //Calcula o
```

```
                                valor da incognita da linha i + 1
                                 da matriz
98                  }
99
100                 //Insere o valor procurado no inicio do vetor de
                        resultados.
101                 resultados.insert(resultados.begin(),Y);
102         }
103
104         std::cout << "Solucoes do sistema: " << std::endl;
105         for(size_t i = 0; i < resultados.size(); i++)
106         {
107                 std::cout <<"X" << i << " = "<< resultados[i] <<
                        std::endl;
108         }
109         std::cout << std::endl;
110 }
111
112 /*
113  *@brief Metodo que imprime a matriz
114  */
115 void SistemaLinear::imprimeMatrizSistema()
116 {
117          for(matriz::iterator it = sistema_m.begin(); it !=
                sistema_m.end(); ++it)
118          {
119                 for(std::vector<float>::iterator jt = it->begin
                        (); jt != it->end(); ++jt)
120         {
121                         std::cout << *jt << " ";
122                 }
123                 std::cout << std::endl;
124          }
125          std::cout << std::endl;
126 }
```

Neste arquivo temos a implementação da nossa classe, na seção a seguir mostraremos uma breve explicação para cada método implementado.

## 1.2. Principais métodos

Como falado antes, nesta seção abordaremos uma simples explicação de como foi elaborado cada método.

### 1.2.1. SistemaLinear(fileName)

Trata-se do construtor da classe, note que para criar um objeto é necessário passar o nome de um arquivo, neste arquivo precisa está armazenada a matriz. O nome do arquivo é passado para outro método dentro do construtor, sendo este outro método o responsável carregamento da matriz.

### 1.2.2. lerMatriz(fileName)

Método que é chamado dentro do construtor, ele é responsável por carregar a matriz a partir de um arquivo e armazena-la dentro de um vector, fizemos este processo utilizando os recursos da biblioteca *fstream*.

### 1.2.3. gerarMatrizTriangular()

Aqui utilizamos o processo de eliminação de Gauss para gerar uma matriz triangular superior. A função em si, consiste basicamente de três laços de iteração para o percorrimento adequado da matriz a ser escalonada. O mais externo determina em que coluna está o pivô que está sendo utilizado para zerar os elementos abaixo dele, o segundo laço itera pelas linhas abaixo do linha do pivô e calcula o fator multiplicativo que será usado atualizar os valores dos elementos de cada uma dessas linhas, e o terceiro laço percorre cada elemento de certa linha, atualizando seu valor com base naquele fator e nos elementos da linha do pivô.

### 1.2.4. solucoesSistema()

Método responsável em encontrar a resolução do sistema linear, utilizando como base, a matriz triangular gerada pelo método descrito acima. Neste algoritmo, a matriz será percorrida de baixo para cima, linha por linha, considerando apenas o elemento da diagonal, os valores posteriores a ele (coeficientes das incógnitas), e o último valor da linha, uma constante. O elementos da diagonal está a multiplicar a incógnita que se quer achar em cada iteração completa em uma linha, esta será encontrada com base nos valores anteriormente calculados, por meio de somas e multiplicações. O código dessa função foi bem comentado para facilitar o entendimento. Por fim, esse método imprime todos os valores do vetor solução do sistema.

### 1.2.5. imprimeMatrizSistema()

Simples método utilizado para imprimir a matriz, basicamente itera a matriz salva e a imprime na tela.