RESOLUÇÃO E IMPLEMENTAÇÃO DE SISTEMAS LINEARES DETERMINADOS ATRAVÉS DOS MÉTODOS DIRETOS

Salete Maria Chalub Bandeira[1]

Simone Maria Chalub Bandeira Bezerra[2]

RESUMO: Este trabalho apresenta o resultado de uma pesquisa realizada nos últimos 5 anos, desenvolvida em conjunto com discentes dos Cursos de Bacharelado em Sistemas de informação, Licenciatura em Matemática e Bacharelado em Engenharia Civil, da Universidade Federal do Acre, na qual verificou-se a importância de se criar um *Software* Matemático para auxiliar na Resolução de Sistemas de Equações Lineares Determinados, aplicando os Métodos Diretos: Eliminação de Gauss, Eliminação de Gauss-Jordan e Fatoração LU, assunto trabalhado na disciplina de Cálculo Numérico/ Métodos Numéricos.
Palavras-chave: sistemas de equações lineares, métodos diretos, s*oftware* matemático.

ABSTRACT: This work presents the result of a research done in the last five years, developed together with the students of the Bachelorship courses of Information Systems', licentiateship in Mathematics and Bachelorship in Civil Engineering, of the Federal University of Acre, in which verified the importance to criate a mathematic software to auxiliate in the Resolution of Systems of Linear Equations Determined, applying the Direct Methods: Elimination of Gauss, Elimination of Gauss-Jordan and Fatoration LU, subject worked in the Numeric Calculus/Numeric Methods.
Key words: linear equations system, direct methods, mathematic software.

[1] Mestre em Ciência da Computação: Matemática Aplicada. Professora do Departamento de Matemática e Estatística da Universidade Federal do Acre. Endereço eletrônico para correspondência: saletechalub@gmail.com

[2] Especialista em Matemática. Professora do Departamento de Matemática e Estatística da Universidade Federal do Acre. Endereço eletrônico para correspondência: simonechalub@hotmail.com

1 INTRODUÇÃO

Os problemas de matemática surgem ao se tentar explicar fenômenos da natureza e prever o seu comportamento futuro, procurando uma relação entre causa e efeito. Busca-se inicialmente, uma explicação para um fenômeno por meio de um modelo que estabeleça uma relação entre causas e efeitos, modelo esse que dá origem aos problemas de matemática como, por exemplo, sistemas de equações lineares.

Para encontrar a solução de um dado problema de matemática, existem vários métodos:

1. *Método Dedutivo*: quando se quer provar alguma verdade matemática.
2. *Método Analítico*: procura-se obter uma relação entre as variáveis independentes e variáveis dependentes satisfazendo alguma(s) condições.
3. *Método numérico*: procura-se obter valores solução (variável dependente) de um problema para valores discretos das variáveis independentes, quando a solução procurada for uma função ou os valores numéricos de variáveis que satisfazem alguma condição.
4. *Método Gráfico*: para exibir graficamente o comportamento de um problema ou mesmo de sua solução, cuja inspeção pode fornecer informações importantes sobre o problema e sua solução. (HATTORI, 1992a).

De início, um problema de matemática pode ser resolvido analiticamente, ou seja, existe um método matemático para se obter a solução, mas esse método pode se tornar impraticável quando o tamanho do problema aumenta. Um exemplo é a solução de sistemas de equações lineares, tendo a matemática conseguido estabelecer as condições para a existência da solução e métodos que permitem calcular essa solução. Um desses métodos é conhecido como Regra de Cramer, cujo procedimento é descrito a seguir.

Seja $A\mathbf{x} = \mathbf{b}$ em que A é uma matriz $n \times n$, $\mathbf{x} = [x_1 \cdots x_n]^T$ e $\mathbf{b} = [b_1 \cdots b_n]^T$ são vetores colunas, um sistema de equações lineares em que a matriz A dos coeficientes é não singular, ou seja, $\det A \neq 0$. Seja A_k a matriz obtida substituindo-se em A a coluna k pelo vetor independente **b**. Então, pela Regra de Cramer $x_k = \dfrac{\det A_k}{\det A}, \quad k = 1,\ 2, \cdots,\ n.$

Portanto, a solução de um sistema de n equações requer o cálculo de $n+1$ determinantes de uma matriz de tamanho $n \times n$. Manualmente, um $n > 10$ torna esta regra impraticável, porque o número de operações é proporcional a $n!$. E o pior, mesmo com o uso do computador, a aplicação é inviável pelo tempo astronômico de processamento necessário. Por exemplo, com $n = 20$, requer 21! operações de multiplicação. Supondo que um computador hipotético gaste 100 nanosegundos numa operação de multiplicação, o tempo

necessário para resolver um sistema de 20×20 será $t = 0{,}5 \times 10^{13}$ *segundos*. Como um dia tem 86.400 segundos, o tempo gasto em anos será de 158.548 anos. Para um sistema de 15 equações lineares $(n = 15)$, requer 16! operações de multiplicação. Supondo que cada operação leve 100 nanosegundos, o tempo gasto será de 24 dias ou 0,066 ano.

A Regra de Cramer é um exemplo de que a simples utilização de uma implementação direta de um método matemático no computador pode ser inviável. Com isso, verifica-se que nem sempre os métodos analíticos têm suas implementações viáveis. Buscam-se então, novas maneiras de resolver sistemas lineares, surgindo, assim, a necessidade de desenvolver métodos numéricos.

Segundo Hattori (1992b), dado um problema e dispondo de um programa para resolvê-lo, basta aprender a usar o programa e fornecer dados do problema. Na verdade, não é tão simples assim. Em computação numérica[3] não basta apenas saber utilizar um programa, é necessário verificar se os resultados obtidos são válidos e se o programa em si não apresenta dificuldades.

Exemplo 1: O sistema linear $\begin{cases} 37639840x - 46099201y = 0 \\ 29180479x - 35738642y = -1 \end{cases}$ foi resolvido por um programa que sabidamente é de boa qualidade. Usando um computador compatível com *IBM PC*, a resposta encontrada foi $x = -42587641{,}592475$ e $y = -34772663{,}750032$. Embora a solução correta seja $x = 46099201$ e $y = 37639840$.

Será que o programa contém erros? Na verdade, o que há de excepcional é o problema e não erros no programa. No caso, o sistema linear é quase indeterminado, levando a uma "solução" que tem pouca confiabilidade.

O termo *software matemático* (*mathematical software*) foi cunhado por John Rice ao convidar pesquisadores para participarem de um simpósio que haveria na *Pardue University* em abril de 1970 (RICE, 1969). Nesse convite, Rice definiu um *software* matemático como sendo programas de computador que implementam procedimentos matemáticos amplamente aplicáveis. Nos anais do simpósio publicados em 1971, Rice definiu o *software* matemático como um conjunto de algoritmos na área de matemática. Essas duas definições ilustram a confusão inicial que se fazia entre programas de computador e algoritmos. A plena compreensão de que uma implementação é diferente do algoritmo básico marca o reconhecimento de *software* matemático como uma disciplina (assunto de ensino e pesquisa).

---

[3] Computação Numérica: *É a resolução numérica de problemas de matemática utilizando o computador.*

Portanto, ao contrário da idéia de produto que Rice deu a entender nos primórdios da história do *software* matemático, hoje ele é considerado uma disciplina, uma atividade que envolve a implementação dos métodos computacionais, preocupa-se com padrões de qualidade, com a melhoria da produtividade do processo de desenvolvimento de *software* de métodos numéricos e com o reconhecimento do *hardware* e do *software* do computador em que os programas são executados.

A terminologia utilizada é introduzida a seguir:

> **Definição 1.1:** *Software matemático* é um ramo da engenharia de software que concebe, implementa, testa e mantém software para solução de problemas de matemática.
> **Definição 1.2:** *Software numérico* é um software cuja finalidade é a solução numérica de problemas de matemática.
> **Definição 1.3:** *Algoritmo básico* é aquele definido matematicamente.
> **Definição 1.4:** *Algoritmo computacional* (numérico, algébrico ou gráfico) é aquele obtido de um algoritmo básico ou de uma combinação deles, levando em conta as características do computador com a finalidade de executar uma seqüência de operações para resolver um problema.

Um *Software matemático* pode se apresentar como:

> Um *pacote*, que é uma coleção de programas para resolver os problemas de uma área, por exemplo, sistemas de equações lineares.
> Uma *biblioteca*, uma coleção de programas para resolver diversas classes de problemas de matemática.
> Um *sistema de software*, constituído de um pacote ou uma biblioteca com uma interface de comunicação com o usuário. (HATTORI, 1995).

Os sistemas de equações lineares surgem em praticamente todas as áreas da física, biologia, engenharia, álgebra linear, ciências sociais, nas soluções numéricas de problemas em equações diferenciais ordinárias, equações diferenciais parciais, equações inteiras, problemas de otimização e de vários outros problemas que precisam da solução de sistemas de equações lineares.

Existem duas formas de resolver sistemas de equações lineares: por *métodos diretos* e por *métodos iterativos*.

Definição 1.5: Diz-se que o método é *direto* quando é possível chegar a uma solução exata após um numero finito de operações aritméticas.

Definição 1.6: Diz-se que o método é *iterativo* quando, partindo de uma aproximação inicial, é possível se chegar a aproximações mais precisas que dependem, sempre, de valores anteriormente calculados.

Roque (2000) define os métodos diretos como aqueles que determinam a solução de um sistema linear por meio de um número finito de passos (operações), e métodos iterativos

como aqueles que requerem um número infinito de passos: calcula-se uma seqüência de soluções $X^{(1)}$, $X^{(2)}$,..., $X^{(k)}$ obtidas com base em uma solução inicial $X^{(0)}$.

2 MATERIAL E MÉTODOS

Neste artigo, pretende-se analisar problemas de matemática, em particular solução de sistemas de equações lineares, em que a atenção especial será dada a uma classe de problema, *Sistemas de equações lineares determinados*, em que se quer encontrar um vetor **x** que satisfaça $Ax = b$, em que A é uma matriz $n \times n$, não singular, **x** e **b** são vetores colunas com n componentes. Para tal, seguir-se-á os seguintes passos:

- Amostra da resolução e da implementação dos métodos diretos: Eliminação de Gauss, Eliminação de Gauss-Jordan e a Fatoração LU.
- Construção dos algoritmos e escolha da linguagem de programação a ser utilizada para implementar os métodos diretos.
- Escolha da estratégia de pivoteamento para evitar a ampliação dos erros de arredondamento.

Do ponto de vista da sua natureza a pesquisa é do tipo aplicada, visando gerar conhecimentos que tenham aplicação prática dirigidos à solução do problema. A forma de abordagem do problema será qualitativa, procurando uma relação dinâmica entre o sistema a ser desenvolvido e sua utilização por parte de seus usuários, de forma que seja possível repassar para o sistema os cálculos realizados de forma manual, para um melhor entendimento e utilização por parte dos interessados em utilizar o *software* desenvolvido.

Seja o sistema linear $\begin{cases} x + y = 2 \\ 2x - y = 1 \end{cases}$, classificado como possível e determinado, cuja solução única, $S = \{(1,\ 1)\}$.

Observe-se, conforme **Figura 1**, a classe de problema a ser tratada:

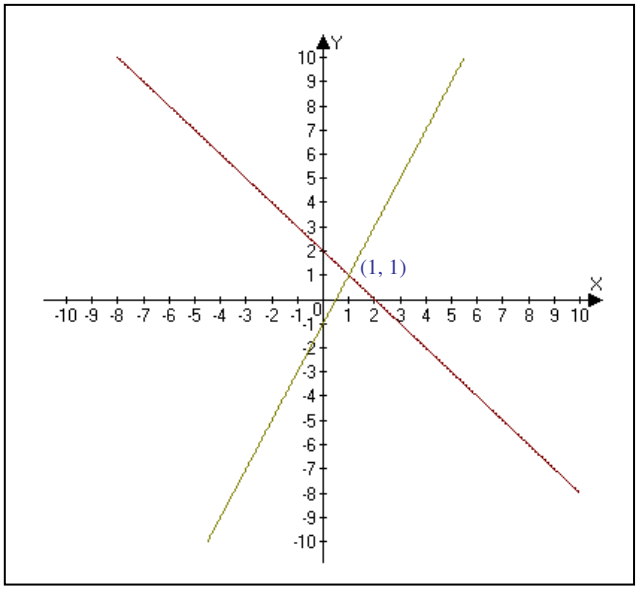


Figura 1: Sistema possível e determinado (solução única)
Software utilizado: *Wingraph* (*shareware*)

3 RESULTADOS E DISCUSSÕES

Conforme acima mencionado, serão apresentados os Métodos Diretos: Eliminação de Gauss, Eliminação de Gauss-Jordan e a Fatoração LU, com seus respectivos algoritmos e telas do protótipo (*software* implementado em ambiente *Delphi* 7.0), descritos nas aulas de cálculo numérico dos cursos de Matemática, Sistemas de Informação e Engenharia Civil da Universidade Federal do Acre - UFAC.

3.1 Eliminação de Gauss

Já conhecido da Análise Numérica, o método de Eliminação de Gauss, triangulariza a matriz A dos coeficientes, ou seja, reduz a matriz A a uma matriz triangular superior ($a_{ij} = 0, \quad \forall i > j$), conforme processo a seguir:

Seja o sistema linear $A\mathbf{x} = \mathbf{b}$, em que A é uma matriz n x n, triangular superior, com elementos da diagonal principal diferentes de zero. Escrevendo as equações deste sistema, tem-se:

$$S = \begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ a_{33}x_3 + \cdots + a_{3n}x_n = b_3 \\ \ddots \qquad\qquad \vdots \\ a_{nn}x_n = b_n \end{cases}$$

Da última equação:

$$x_n = \frac{b_n}{a_{nn}}$$

$\mathbf{x}_{n\text{-}1}$ pode então ser obtido da penúltima equação:

$$x_{n-1} = \frac{b_{n-1} - a_{n-1,n}\,x_n}{a_{n-1,n-1}}$$

e, assim sucessivamente obtém-se $x_{n\text{-}2}$, ..., x_2, e x_1:

$$x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n}{a_{11}} \quad .$$

Uma matriz que pode ser associada ao sistema *S*, é a matriz ampliada, denotada por

$$[A|\mathbf{b}] = \begin{bmatrix} a_{11} & a_{12}\cdots a_{1n} & b_1 \\ a_{21} & a_{22}\cdots a_{2n} & b_2 \\ \vdots & \vdots \quad \vdots & \vdots \\ a_{n1} & a_{n2}\cdots a_{nn} & b_n \end{bmatrix}, \qquad \textbf{(3.0)}$$

em que $b_1 = a_{1(n+1)}, b_2 = a_{2(n+1)}, \cdots, b_n = a_{n(n+1)}$.

O *primeiro passo* é a eliminação de todos os elementos da primeira coluna de [*A*|**b**], exceto a_{11}(pivô). Suponha-se o pivô $a_{11} \neq 0$. Definam-se os multiplicadores por $m_{i1} = \frac{a_{i1}}{a_{11}}$ para $i = 2,3,\ldots,n$. Multiplica-se a primeira linha por m_{i1}, e subtrai-se da *i*-ésima linha, para $i = 2,3,\ldots,n$, obtendo

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}-m_{21}a_{12} & \cdots & a_{2n}-m_{21}a_{1n} & b_2-m_{21}b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}-m_{n1}a_{12} & \cdots & a_{nn}-m_{n1}a_{1n} & b_n-m_{n1}b_1 \end{bmatrix}$$

Reescreve-se a nova matriz como:

$$[A|b]^{(1)} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix},$$

em que (1) indica que uma eliminação foi realizada com: $a_{ij}^{(1)} = a_{ij} - m_{i1}a_{1j}$ para $i, j = 2,3,\cdots,n$ e $b_i^{(1)} = b_i - m_{i1}b_1$ para $i = 2,3,\cdots,n$.

Repetindo este procedimento $(n-1)$ vezes, obtém-se:

$$[A|b]^{(n-1)} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{bmatrix}, \textbf{(3.1)}$$

no qual na *i*-ésima etapa os pivôs $a_{ii}^{(i-1)} \neq 0$, para $i = 1,2,\ldots,n-1$, em que;

$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_{ik}a_{kj}^{(k-1)}$, $b_i^{(k)} = b_i^{(k-1)} - m_{ik}b_k^{(k-1)}$ e $m_{ik} = \dfrac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$, para $k = 1,2,\ldots,n-1$ e $i, j = k+1, k+2,\ldots,n$ e $a_{ij}^{(0)} = a_{ij}$, para $i, j = 1,2,\ldots,n$ e $b_i^{(0)} = b_i$ para $i = 1,2,\ldots,n$.

De **(3.1)**, $x_n = \dfrac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$, se $a_{nn}^{(n-1)} \neq 0$ e $x_i = \dfrac{1}{a_{ii}^{(i-1)}}\left[b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)}x_j\right]$, para $i = n-1, n-2,\ldots,1$.

O sistema linear $A^{(n-1)}\mathbf{x}=\mathbf{b}^{(n-1)}$, **(3.1)** é equivalente ao sistema linear original, $A\mathbf{x} = \mathbf{b}$ **(3.0)**.

O método de Eliminação de Gauss descrito anteriormente, foi implementado conforme o *Algoritmo 1*, sem a estratégia de pivoteamento parcial e aplicando a estratégia de pivoteamento.

Algoritmo 1: Eliminação de Gauss

```
1.  for cont:= 0 until NrLC−2 do
2.      if  Mat[cont, cont]<>0 then
3.          for i:= cont +1 until NrLC−1 do
4.              mult:= Mat[i, cont] / Mat[cont, cont];
5.              for j:=0 until NrLC do
6.                  if  Mat[i, j]<>0 then
7.                      Mat[i,j]:= Mat[i, j] – Mat[cont,j]*mult);
8.          end_for
9.      end_if
10. end_for
11. if  Mat[NrLC–1, NrLC–1]<>0 then
12.      X1[NrLC–1]:= Mat[NrLC–1, NrLC]/ Mat[NrLC–1, NrLC–1];
13.  else
14.     X1[NrLC–1]:=0;
15.  for i:= NrLC–2 downto until 0 do
16.    aux:=0;
17.    for j:= i+1 until NrLC–1 do
18.        aux:= aux + (Mat[i,j]*X1[j]);
19.  end_for
```

Variáveis:

NrLC: Número de linhas e colunas;

cont: controla o número de etapas;

i e j: índices utilizados para acessar posições tanto nos vetores, como na matriz;

Mat[i, j]: Matriz onde está armazenado os valores do sistema;

X1[j]: vetor que armazena a solução do sistema.

aux:= variável auxiliar. Guarda resultados que poderão ser utilizados posteriormente.

O método de Eliminação de Gauss-Jordan, diagonaliza A ($a_{ij} = 0,\ \forall i \neq j$), isto é, apenas os elementos da diagonal principal são não nulos. Caso não queiramos fazer nenhuma

substituição, reduzimos os elementos da diagonal à unidade (transformamos *A* em uma matriz identidade). O processo requer *n* etapas de eliminação, semelhante à Eliminação de Gauss, em que a idéia é reduzir todos os elementos das colunas para zero, exceto os elementos da diagonal. A matriz ampliada, [A, **b**]$^{(n)}$, com *n* etapas de eliminação, terá o formato como segue:

$$[\boldsymbol{A}|\mathrm{b}]^{(n)} = \begin{bmatrix} a_{11} & 0 & \cdots & 0_{1n} & b_1^{(n)} \\ 0 & a_{22}^{(1)} & \cdots & 0 & b_2^{(n)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{bmatrix}, \textbf{(3.2)}$$

O método de Eliminação de Gauss-Jordan descrito anteriormente, foi implementado conforme o *Algoritmo* 2, com a opção de aplicar ou não a estratégia de pivoteamento parcial.

Algoritmo 2: Eliminação de Gauss-Jordan

```
1.  for cont:= 0 until NrLC−1 do
2.     if  Mat[cont, cont]<>0 then
3.        for i:= 0 until NrLC−1 faça
4.           if  i<>cont then
5.              mult:=0;
6.              mult:= Mat[i, cont] / Mat[cont, cont];
7.              for j:=cont until NrLC do
8.                 if (Mat[cont, j]<>0) e (i<>cont) then
9.                    Mat[i,j]:= Mat[i, j] – Mat[cont,j]*mult);
10.          end_if
11.       end_for
12.    end_if
13. end_for
14. for i:=0 until NrLC−1 do
15.       if   Mat[i, i]<>0 then
16.          X1[i]:= Mat[i, NrLC]/ Mat[i, i];
17.       else
18.          X1[i]:=0;
19. end_for
```

Variáveis:

NrLC: Número de linhas e colunas;

cont: controla o número de etapas;

i e j: índices utilizados para acessar posições tanto nos vetores, como na matriz;

Mat[i, j]: Matriz onde está armazenado os valores do sistema;

X1[j]: vetor que armazena a solução do sistema.

aux:= variável auxiliar. Guarda resultados que poderão ser utilizados posteriormente.

O método da Fatoração LU consiste em decompor a matriz *A* dos coeficientes, em um produto de duas matrizes triangulares, isto é, *A*= *LU,* conforme teorema descrito abaixo.

Segundo Ruggiero (1996): Dada uma matriz quadrada *A* de ordem *n*, seja A_k a matriz constituída das primeiras *k* linhas e colunas de *A*. Suponha que $\det A_k \neq 0$ para $k = 1,2,\ldots,(n-1)$. Então, existe uma única matriz triangular inferior $L = (m_{ij})$, com $m_{ii} = 1,\ 1 \leq i \leq n$ e uma única matriz triangular superior $U = (u_{ij})$ tais que $LU = A$. Ainda mais, $\det A = u_{11}u_{22}\ldots u_{nn}$.

Para resolver o sistema linear, *A***x** =**b**, como $LU = A$, tem-se,

- $A\mathbf{x} = \mathbf{b} \Leftrightarrow (LU)\mathbf{x} = \mathbf{b}$, seja $y = Ux$

Assim, a solução do sistema linear pode ser obtida da resolução dos sistemas lineares triangulares, isto é, $Ly = b$ (encontra-se o vetor **y**) e em $Ux = y$ (obtém-se o vetor **x**, solução do sistema linear *A***x** =**b**).

A fatoração L.U consiste em escalonar a matriz A, transformando *A* em uma matriz triangular superior, isto é, decompomos *A*=*LU*, no qual *L* é uma matriz triangular inferior, cujos termos abaixo da diagonal principal são os multiplicadores encontrados nas etapas de Eliminação de Gauss e *U* é uma matriz triangular superior, conforme descrito em Gauss. Vide *Algoritmo 3*.

Algoritmo 3: Fatoração LU

```
1.  for cont:= 0 until NrLC−2 do
2.      if Mat[cont, cont]<>0 then
3.          for i:= cont +1 until NrLC−1 do
4.              L[i, cont]:= Mat[i, cont] / Mat[cont, cont];
5.              for j:=0 until NrLC do
6.                  if Mat[i, j]<>0 then
```

```
7.                    Mat[i,j]:= Mat[i, j] – Mat[cont,j]*L[i, cont]);
8.                  else
9.                     Mat[i, j]
10.       end_for
11.    end_if
12. end_for
13. for  i:=0 until NrLC−1 do
14.     for j:=0 until NrLC−1 do
15.        if j>i then
16.           L[i,j]:= 0;
17.              if i=j then
18.                 L[i,j]:=1;
19.     enf_for
20. end_for
21. Y[0]:= Mat[0, NrLC] / L[0,0];
22. for i:=1 until NrLC−1 do
23.    aux:=0;
24.    for j:=0 until i −1 do
25.       aux:= aux + (L[i,j]*Y[j]);
26. end_for
27. if  Mat[NrLC−1, NrLC−1]<>0 then
28.   X1[NrLC−1]:= Y[NrLC−1]/Mat[NrLC−1, NrLC];
29. else
30.     X1[NrLC-1]:=0;
31. for i:= NrLC−2 downto 0 do
32.     aux:=0;
33.     for j:= i+1 until NrLC−1do
34  .       aux:= aux + (Mat[i,j]*X1[j]);
35.     if Mat[i,i]<>0 then
36.        X1[i]:= (Y[i]−aux)/Mat[I,i];
37.     else
38.        X1:=[0];
39. end_for
```

Variáveis:

L: Matriz dos Multiplicadores;

Y: Vetor para armazenar o vetor "y" da fórmula de decomposição da Matriz A.

NrLC: Número de linhas e colunas;

cont: controla o número de etapas;

i e j: índices utilizados para acessar posições tanto nos vetores, como na matriz;

Mat[i, j]: Matriz onde está armazenado os valores do sistema;

X1[j]: vetor que armazena a solução do sistema.

aux:= variável auxiliar. Guarda resultados que poderão ser utilizados posteriormente.

Para amenizar os erros de arredondamento na resolução e implementação dos sistemas de equações lineares adotou-se a Estratégia de Pivoteamento Parcial, conforme *Algoritmo* 4.

Algoritmo 4: Estratégia de Pivoteamento Parcial

```
1.  if Pivoteamento = 1 then
2.     pivot:= false;
4.     for i:= cont + 1 until NrLC - 1 do
5.        if (Mat[cont,cont]< Mat[i,cont]) or (Mat[cont, cont] < (-Mat[i,cont])) then
6.            for j:= 0 until NrLC do
7.                aux:= Mat[cont,j];
8.                Mat[cont,j]:= Mat[i,j];
9.                Mat[i,j]:= aux;
10.               pivot:= true
11.           end_for;
12.    end_if
13.   end_for;
14. end_if.
```

Para os métodos diretos: entre Eliminação de Gauss e Gauss-Jordan o primeiro é mais bem aplicado; basta verificar a sua complexidade para chegar-se a esta conclusão. Na tabela abaixo, temos o número de operações realizadas com ambos os métodos:

	Eliminação de Gauss			Eliminação de Gauss-Jordan		
N	Divisões	Multiplicações	Adições/ Subtrações	Divisões	Multiplicações	Adições/ Subtrações
3	6	11	11	9	12	12
5	15	50	50	25	60	60
10	55	375	375	100	495	495
100	5050	338250	338250	10000	499950	499950

Tabela 1: Número de operações realizadas com os Métodos de Eliminação de Gauss e Gauss-Jordan.

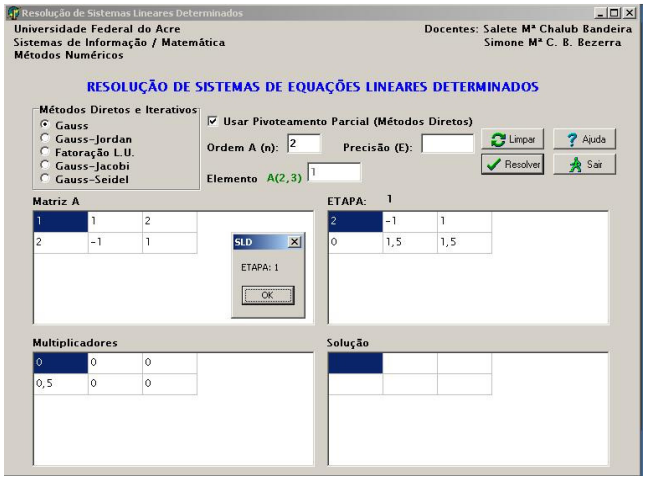
Segundo Bandeira (1998), a Eliminação de Gauss é a mais recomendada, uma vez que para o n grande, o número de operações é proporcional a $2n^3/3$ e para Gauss-Jordan, n^3. Maiores detalhes são descritos em Patel (1994).

Em relação à Fatoração LU, inicialmente não se trabalha com os termos independentes da matriz ampliada relacionada ao sistema, e decompõe-se a matriz A em duas matrizes triangulares, superior e inferior; pode-se dizer que, dentre os três métodos diretos, é o mais recomendado.

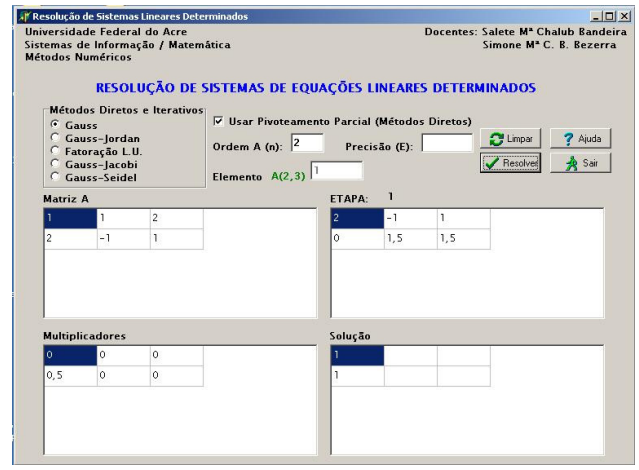
5 CONCLUSÕES

O estudo dos Métodos Diretos na Resolução de Sistemas de Equações Lineares é de suma importância para as áreas de conhecimento nas quais podem surgir problemas modelados na forma de sistemas de equações. Conforme já mencionado, objetiva-se, neste estudo, construir um *software* matemático que resolva Sistema de Equações Lineares Determinados, aplicando os métodos diretos descritos anteriormente, conforme algoritmos construídos para este fim.

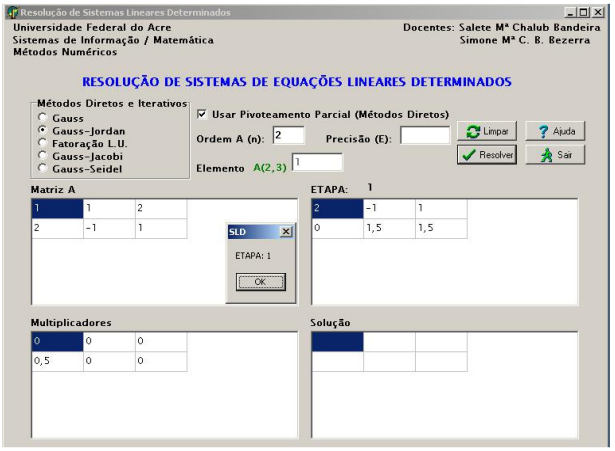
Em relação à convergência, os métodos diretos são processos diretos finitos, portanto, teoricamente obtêm a solução de qualquer sistema não singular de equações. Não são bem aplicados para matrizes esparsas. Em relação aos erros de arredondamento, a utilização da estratégia de pivoteamento parcial constitui-se numa forma de amenizar o problema. Observe-se a seguir as telas do *software* desenvolvido, resultado desta pesquisa.



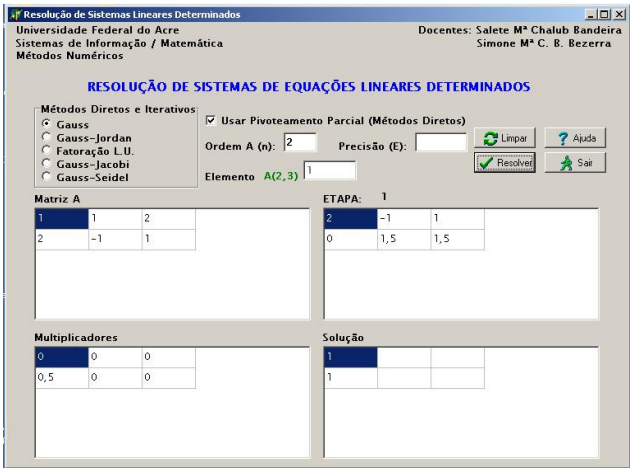
Tela 1: Método de Gauss - I



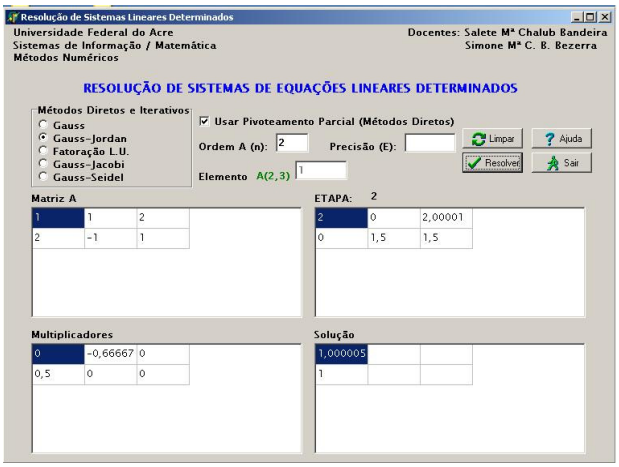
Tela 2: Método de Gauss - II



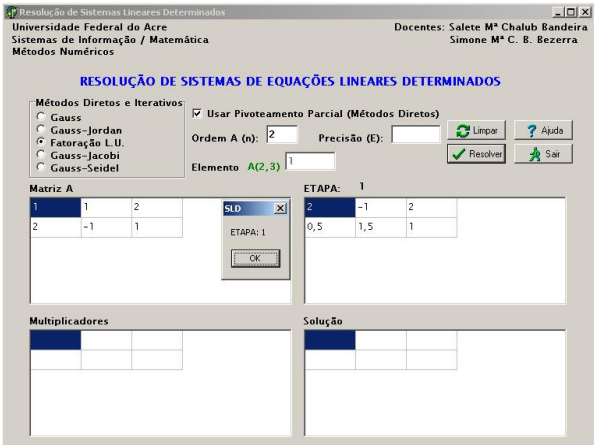
Tela 3: Método de Gauss-Jordan - I



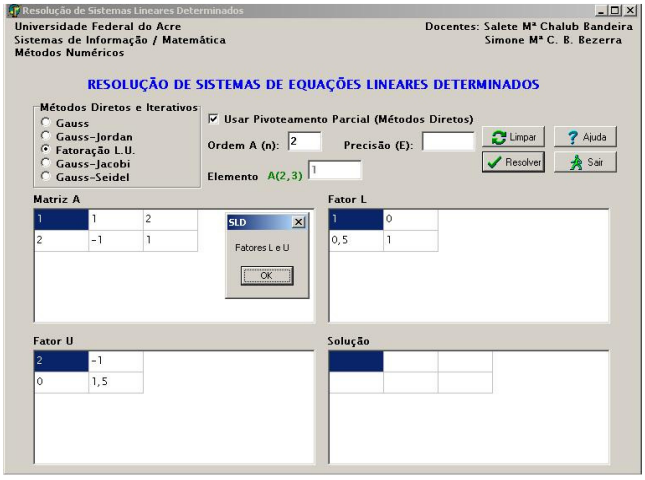
Tela 4: Método de Gauss-Jordan - II



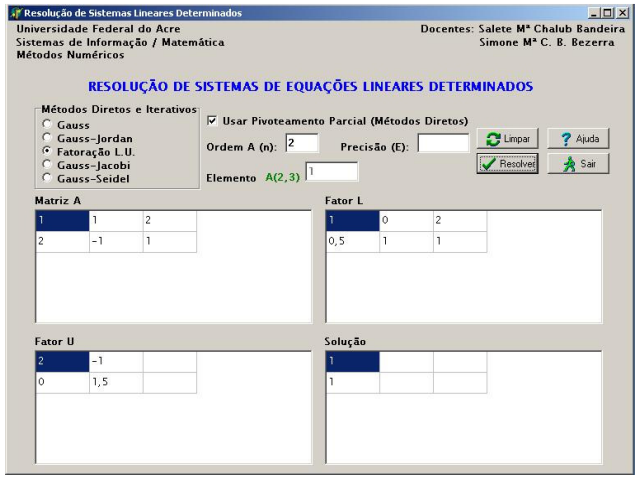
Tela 5: Método de Gauss-Jordan - III



Tela 6: Fatoração LU - I



Tela 7: Fatoração LU - II



Tela 8: Fatoração LU - III

REFERÊNCIAS

BANDEIRA, S. M. C. *Solução Exata de Sistemas de Equações Lineares Utilizando a Aritmética Residual*. Campina Grande: UFPB, 1998.

HATTORI, Mário T. & QUEIROZ, Bruno C. N. *Métodos e Softwares Numéricos*. Paraíba: Universidade Federal da Paraíba, 1995.

HATTORI, Mário Toyotaro. *Software para Computação Numérica*. Relatório Técnico, Departamento de Sistemas e Computação. Campina Grande: Universidade Federal da Paraíba, CampusII,1992a.

_____. Mário Toyotaro. *Solução de Problemas de Matemática usando o Computador*. Relatório Técnico, Departamento de Sistemas e Computação. Universidade Federal da Paraíba, CampusII, Campina Grande, 1992b.

RICE, J. R. Annoucement and Call for Papers. Mathematical Sofware. *SIGNUM Newsletter*. v. 4, p. 7, 1969.
_____. J. R. (ed.). *Mathematical Sofware*. Academic Press, New York, 1971.

ROQUE, Valdir L. *Introdução ao Cálculo Numérico* – Um texto integrado com DERIVE. São Paulo: Editora Atlas, 2000.
RUGGIERO, Márcia A. Gomes & LOPES, Vera Lúcia da Rocha. *Cálculo Numérico* – Aspectos Teóricos e Computacionais. 2. edição. São Paulo: Pearson Education do Brasil, 1996.
SOFTWARE Wingraph Version 1.1 (*Shareware*). Dean Dinnebeil and XWare, Inc. Copyright(c) 1992. Disponível em: **http://www.somatematica.com.br**. Acesso em: 20 ago. 2006.