

Manual da ferramenta MSN - Resolução de Sistemas de Equações Lineares

MSN 2008.2

3 de fevereiro de 2009

Sumário

1	Introdução	2
1.1	Licença	2
1.2	Código Fonte	2
1.3	Equipe de Desenvolvedores	2
1.3.1	Métodos	2
1.3.2	Interface	3
2	Usando a ferramenta	4
2.1	Dependências	4
2.2	Execução	4
2.3	Definindo o Sistema de Equações Lineares	5
2.3.1	Text Mode	5
2.3.2	Equation Mode	6
2.3.3	Matrix Mode	6
2.4	Métodos	6
3	Resolução de Sistemas de Equações Lineares	11
3.1	Métodos de Resolução de Sistemas de Equações Lineares	13
3.1.1	Método da Eliminação de Gauss com e sem pivoteamento	14
3.1.2	Método da Eliminação de Gauss-Jordan com e sem pivoteamento	15
3.1.3	Método da Decomposição LU e SVD	15
3.1.4	Método da Decomposição de Cholesky e QR	16
3.1.5	Método de Gauss-Jacobi	17
3.1.6	Método de Gauss-Seidel	17
4	Manual do Desenvolvedor	18
4.1	Organização do Código	19
4.2	Testes de Unidade	19
4.3	Arquitetura do Sistema	19
4.3.1	Comunicação entre os módulos	20
4.4	Guia de Usabilidade da Interface	22

Capítulo 1

Introdução

A ferramenta MSN - Resolução de Sistemas de Equações Lineares foi desenvolvida para a disciplina de *Métodos de Software Numérico* ministrada pelo professor Eustáquio Rangel na *Universidade Federal de Campina Grande* pelo *Departamento de Sistemas e Computação*. Tendo como desenvolvedores toda a equipe de alunos da disciplina, e usando a linguagem de programação Java, a ferramenta busca resolver sistemas de equações lineares usando diversos métodos pesquisados e discutidos em sala de aula.

Este documento descreve um manual de uso da ferramenta. No capítulo 2, é descrita a instalação e uso geral da ferramenta. O capítulo 3 dá um breve embasamento teórico dos procedimentos realizados para a solução de sistemas de equações lineares. Por fim, o capítulo 4 descreve um pequeno manual para o desenvolvedor, com as decisões arquiteturais do projeto e uma descrição do funcionamento e organização do código desta aplicação.

1.1 Licença

Esta ferramenta está liberada sob a licença GPL 2.0[2].

1.2 Código Fonte

O acesso ao código fonte, documentação e informações sobre o projeto podem ser acessadas no site: <http://code.google.com/p/msn-sistemas-lineares/>. Para acessar o código fonte, basta usar o SVN [5], localizado no endereço: <http://msn-sistemas-lineares.googlecode.com/svn/trunk/>.

1.3 Equipe de Desenvolvedores

1.3.1 Métodos

- Adauto Trigueiro
- Alan Farias
- Anderson Pablo

- Diego Melo Gurjão
- Everton Leandro
- João Felipe Ouriques
- José Wilson
- José Gildo
- Leonardo Ribeiro Mendes
- Rafael Dantas
- Rodrigo Pinheiro

1.3.2 Interface

- Dayane Gaudencio
- Hugo Marques
- Jackson Porciuncula
- Matheus Gaudencio
- Ricardo Araújo
- Roberta Guedes
- Theo Alves

Capítulo 2

Usando a ferramenta

2.1 Dependências

A ferramenta executa sobre a plataforma Java, aproveitando todas as suas vantagens como, por exemplo, a independência de sistema operacional, desde que exista uma máquina virtual compatível com tal sistema.

2.2 Execução

Utilizando o jar disponibilizado, para executar o usuário necessita do comando:

```
java -jar MSN-Sistemas-Lineares-0.1.jar
```

Dessa forma a ferramenta não necessita de instalação, trazendo uma vantagem ao usuário que não tem permissões para instalar aplicativos.

A figura 2.1 mostra um exemplo do programa em execução.

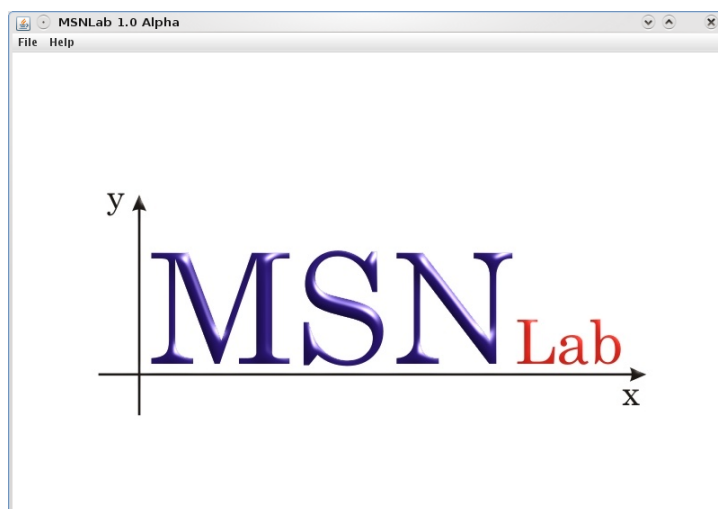


Figura 2.1: Aplicativo em execução

2.3 Definindo o Sistema de Equações Lineares

Ao iniciar o aplicativo, a janela da figura 2.2 é exibida. Nela, é possível ver três formas de entrada para o sistema de equações lineares:

Text Mode Permite que o sistema seja inserido como texto (este é o modo inicial).

Equation Mode Insere equação por equação do sistema.

Matrix Mode Trabalha com o sistema na forma de uma matriz.

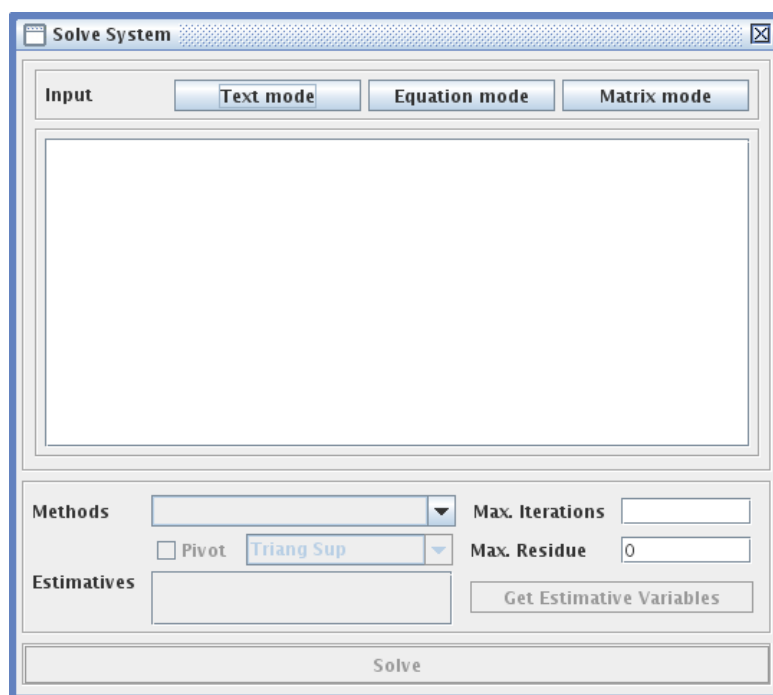


Figura 2.2: Janela inicial (Text Mode)

2.3.1 Text Mode

O modo de inserção textual permite que o sistema seja descrito em um texto onde cada linha representa uma equação. Note que as variáveis tanto podem assumir valores como x , y , z como x_1 , x_2 , x_3 , etc. Um exemplo de entrada permitida:

$$\begin{array}{l} 10x + y = 10 \\ x - y = 5 \end{array}$$

2.3.2 Equation Mode

Neste modo, representado na figura 2.3, cada linha representa uma equação a ser inserida. Assim é possível ao usuário 3 opções de controle quanto ao número de equações de entrada:

Add Equation Adiciona uma nova equação no sistema.

Remove Equation Remove uma equação do sistema (elimina a última equação).

Set Number of Equations Define o número de equações, adicionando ou removendo o número necessário de equações de sistema.

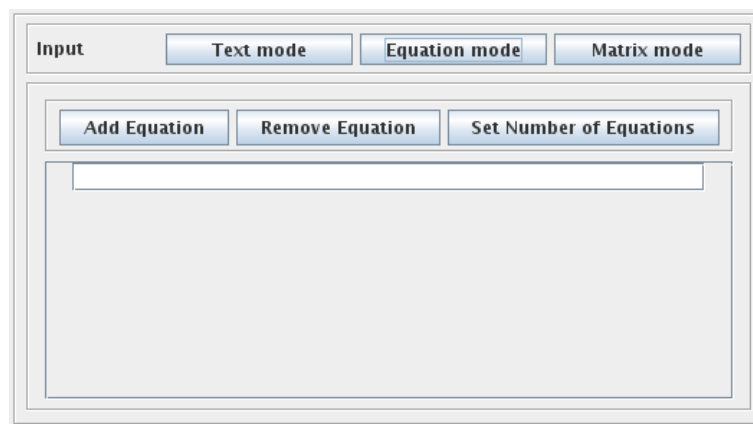


Figura 2.3: Equation Mode

2.3.3 Matrix Mode

O último modo de entrada de sistemas de equações lineares, se dá na forma de matriz. A figura 2.4 mostra a tela que representa este modo. Nela, o usuário entrará com o sistema na forma de matriz podendo, caso necessário: alterar o nome das variáveis (**Set variable's name**), definir o número de variáveis (**Set number of variables**) ou limpar a matriz (**Clear**).

2.4 Métodos

Ao entrar com a matriz, o campo **methods** torna-se disponível. Nele, é possível selecionar o método a ser utilizado, como mostra a figura 2.5. Cada método tem um conjunto de parâmetros associado, de acordo com o método selecionado.

A figura 2.6 mostra os parâmetros a serem configurados no método de resolução de sistemas de equações lineares através da eliminação de Gauss.

Pivot Determina se o método usará ou não um pivot;

Max. Iterations Define o número máximo de iterações para refinar a solução;

Figura 2.4: Matrix Mode

Figura 2.5: Seleção dos Métodos

Max. Residue Valor máximo do resíduo encontrado para determinar que uma solução é válida;

Triang. Sup/Inf Determina se a eliminação usará triangularização superior ou inferior.

Figura 2.6: Método de Eliminação de Gauss

Enquanto **Pivot** é uma seleção comum ao método de eliminação de Gauss e o método de Gauss-Jacobi, a opção de escolher o modo de triangularização é uma opção apenas do método de eliminação de Gauss. As demais opções são comuns a todos os métodos diretos, como o de decomposição QR (como mostra a figura 2.7).

Ao resolver um sistema de equações lineares, os métodos iniciam uma janela de saída mostrando o passo-a-passo da resolução do sistema. A figura 2.8 exibe

Methods: DECOMPOSICAO_QR, Max. Iterations: 10, Max. Residue: 5, Estimatives: (empty), Get Estimative Variables, Solve

Figura 2.7: Método de Decomposição QR

um dos passos da resolução através do método de Eliminação de Gauss. Para exibir o próximo passo, é preciso selecionar o botão \rightarrow . Intuitivamente, para exibir um passo anterior, basta selecionar o botão \leftarrow .

col 1	col 2	Term
1	1	10
1	-1	-5

<- 1/4 ->

Figura 2.8: Eliminação de Gauss: Alterando a Matriz

Nem todos os métodos apresentam uma etapa de alteração de matriz, entretanto, todos os métodos diretos exibem uma etapa de refinamento de solução (Finding Solution) que pode ser visto na figura 2.9.

Variables	Iteration 3	Solution
x1	2.5	2.5
x2	7.5	7.5

<- 3/4 ->

Figura 2.9: Eliminação de Gauss: Refinando a Solução

Terminada esta etapa, o método exibe a solução encontrada para o sistema de equações lineares (figura 2.10).

Os métodos iterativos (Gauss-Seidel e Gauss-Jacobi), por sua vez, apresen-

Solution	
Variables	Values
x1	2.5
x2	7.5

< - 4/4 - >

Figura 2.10: Eliminação de Gauss: Exibindo a Solução

tam três parâmetros configuráveis:

Max. Iterations Número máximo de iterações que o método pode realizar (critério de parada);

Tolerance Tolerância da solução para que ela seja aceita como válida;

Estimatives Valores iniciais (estimativas) de cada variável.

A figura 2.11 mostra um exemplo dos parâmetros no método de Gauss-Seidel. Note que ao selecionar o método, as estimativas ainda não são possíveis de serem definidas. Para definir as estimativas iniciais, pressione o botão **Get Estimative Variables**.

Methods: GAUSS_SEIDEL Max. Iterations: 10

☐ Pivot Triang Sup Tolerance: 5

Estimatives: [Empty field]

[Get Estimative Variables]

[Solve]

Figura 2.11: Gauss-Seidel

É possível ver na figura 2.12 os parâmetros de estimativas a serem definidas pelo usuário. Por padrão, todas as estimativas são definidas como zero.

Methods: GAUSS_SEIDEL Max. Iterations: 10

☐ Pivot Triang Sup Tolerance: 5

Estimatives	X	Y	Z
	0	0	0

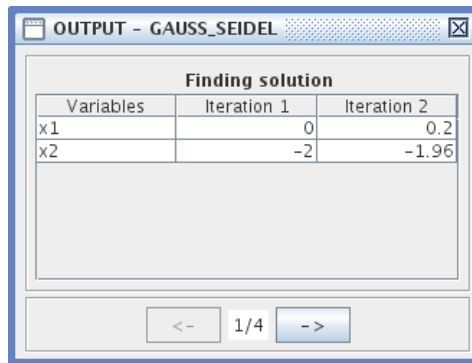
[Get Estimative Variables]

[Solve]

Figura 2.12: Gauss-Seidel: Definindo as Estimativas

Quando o comando de resolver o sistema é chamado, e se faz uso de um método iterativo, uma janela exhibe os valores de cada iteração que o método

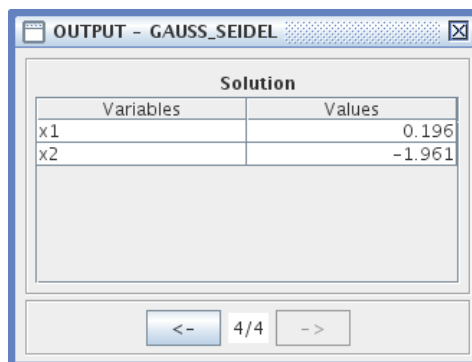
define para as variáveis (figura 2.9). Como último passo, a janela exibe a solução encontrada (figura 2.10).



Variables	Iteration 1	Iteration 2
x1	0	0.2
x2	-2	-1.96

< - 1/4 - >

Figura 2.13: Gauss-Seidel: Iterações



Variables	Values
x1	0.196
x2	-1.961

< - 4/4 - >

Figura 2.14: Gauss-Seidel: Solução

Capítulo 3

Resolução de Sistemas de Equações Lineares

Esta seção descreve como a ferramenta trabalha para a resolução de sistemas de equações lineares. Para contextualizar o leitor, é importante entender como são definidos tais sistemas. Esta base é usada posteriormente na explicação dos métodos de resolução de sistemas de equações lineares.

Todo sistema de equações lineares representa uma composição de equações lineares, como mostra o exemplo a seguir:

$$\begin{array}{ccccccccc} a_{1,1}x_1 & + & a_{1,2}x_2 & \cdots & + & a_{1,n}x_n & = & b_1 \\ a_{2,1}x_1 & + & a_{2,2}x_2 & \cdots & + & a_{2,n}x_n & = & b_2 \\ \vdots & & \vdots & \ddots & & \vdots & & \vdots \\ a_{m,1}x_1 & + & a_{m,2}x_2 & \cdots & + & a_{m,n}x_n & = & b_m \end{array}$$

É possível obter uma matriz representando os coeficientes das equações lineares que compõe o sistema. Esta matriz é da forma:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

Dois outros vetores também são obtidos a partir das equações lineares do sistema inicial. O primeiro vetor representa as incógnitas do sistema:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

E o segundo vetor composto pelos termos independentes do sistema:

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Observe que ao multiplicar cada linha de A , ou seja, da *matriz dos coeficientes*, pelo vetor x (incógnitas do sistema), obtêm-se como resultado cada valor do vetor dos termos independentes, b . Assim, o sistema original pode ser representado por A , x e b como descrito na equação 3.1.

$$Ax = b \quad (3.1)$$

Uma representação útil do sistema, a ser usado em alguns métodos a serem apresentados é a da matriz expandida do sistema. Esta matriz A_{expand} é definida como a matriz dos coeficientes acrescida ao vetor dos termos b . Ou seja:

$$A_{expand} = \left[\begin{array}{cccc|c} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} & b_m \end{array} \right]$$

Visualizando o sistema como uma operação matricial, é possível identificar propriedades do sistema a partir das propriedades da matriz de coeficientes da equação. Em especial, a matriz A de tamanho $m \times n$, apresenta m incógnitas e n equações lineares.

Durante a resolução de tal sistema, três possibilidades de resultados estão descritos na tabela 3.1.

Tabela 3.1: Possíveis Resultados de um Sistema

Tipo	Descrição
Possível e Determinado	O sistema é possível e apresenta uma única solução
Possível e Indeterminado	O sistema é possível, mas não apresenta uma única solução
Impossível	O sistema não tem solução

Num sistema onde $m > n$, ou seja, há mais incógnitas do que equações, não é possível determinar uma solução única para o sistema: cada equação representa um vetor no espaço de m dimensões, e a solução única só é possível quando a interseção de todas as equações forma um ponto no espaço. Note que tal sistema também poderia não ter solução: basta considerar o exemplo de quando dois vetores são paralelos no espaço, assim, não apresentando pontos comuns de interseção.

Quando $m < n$, o sistema possivelmente não apresenta soluções. Fazendo um análogo com a interpretação espacial de um sistema de equações lineares, m equações possivelmente apresentará como interseção um único ponto no espaço. A adição de uma nova equação pode não interceptar este mesmo ponto, fazendo com o que o sistema se torne impossível de ser solucionado. Observe que é possível que a nova equação ainda permita que o sistema tenha uma única

solução: basta que este vetor intercepte o mesmo único ponto que os demais vetores no espaço.

Devido as possíveis interpretações de um sistema em que $m \neq n$, a ferramenta limita-se apenas em explorar os sistemas homogêneos. Nestes sistemas $m = n$. Esta restrição permite explorar a matriz de entrada do sistema, verificado ao usuário qual a classificação do tipo do sistema (possível e determinado, possível e indeterminado ou impossível).

Para realizar esta verificação, utiliza-se de parte da regra de Cramer [6]. Em especial, considerando que a matriz A tem tamanho $n \times n$, então:

$$D \equiv \begin{vmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{vmatrix}$$

Ainda, pra cada incógnita existe um D_k tal que:

$$D_k \equiv \begin{vmatrix} a_{1,1} & \cdots & b_1 & a_{1,k+1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & b_n & a_{n,k+1} & \cdots & a_{n,n} \end{vmatrix}$$

Assim, considerando D e D_k , é possível determinar o tipo de resolução do sistema, como mostra as condições da tabela 3.2.

Tabela 3.2: Condições para Identificação das Soluções de um Sistema

Tipo	Condição
Possível e Determinado	$D \neq 0$
Possível e Indeterminado	$D = 0, \forall k, D_k = 0$
Impossível	$D = 0, \exists k D_k \neq 0$

Assim, considerando a limitação da ferramenta em que apenas sistemas homogêneos são verificados, a ferramenta também verifica se o sistema tem solução possível e determinada, possível e indeterminada, ou se é impossível de ser resolvido. Este cálculo é feito a partir das determinantes do sistema, como apresentado anteriormente.

3.1 Métodos de Resolução de Sistemas de Equações Lineares

Os métodos usados pela ferramenta podem ser classificados como diretos ou iterativos. Nos métodos diretos, são conhecidos os números de passos e o procedimento para se obter um vetor de solução do sistema. Enquanto nos métodos iterativos, a solução é refinada até que seja obtido um vetor que satisfaça à precisão definida.

São métodos diretos:

- Método da Eliminação de Gauss
- Método da Eliminação de Gauss-Jordan

- Método da Decomposição LU e SVD
- Método da Decomposição de Cholesky e QR

São métodos iterativos:

- Método de Gauss-Jacobi
- Método de Gauss-Siedel

As soluções obtidas pelos métodos diretos são refinadas até atingirem um critério definido pelo usuário. Considerando que x representa um vetor de soluções, idealmente:

$$b - Ax = 0$$

Entretanto, pelos erros associados as operações numéricas do software, é possível que se obtenha um resíduo r no cálculo desta solução:

$$b - Ax = r \quad (3.2)$$

Para amenizar o resíduo, procura-se um vetor de correção c tal que:

$$\begin{aligned} A(x + c) - b &= 0 \\ Ax + Ac - b &= 0 \\ Ax - b + Ac &= 0 \\ Ac &= b - Ax \end{aligned}$$

Aplicando a equação 3.2:

$$Ac = r \quad (3.3)$$

Basta resolver a equação 3.3 para obter um novo vetor de soluções x^1 tal que $x^1 = x + c$. Esta nova solução pode ser novamente refinada até que o vetor de resíduo satisfaça os critérios definidos pelo usuário.

3.1.1 Método da Eliminação de Gauss com e sem pivoteamento

O método da eliminação de Gauss faz uso de três operações básicas sobre a matriz expandida do sistema que não altera a solução do mesmo:

- Multiplicação de uma equação (linha) por uma constante não nula
- Soma do múltiplo de uma equação a outra
- Troca de posição de duas ou mais equações

O método utiliza de tais operações na busca de uma matriz triangular, isto é, na forma:

$$A_{expand} = \left[\begin{array}{cccc|c} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,n} & b_1 \\ 0 & \delta_{2,2} & \cdots & \delta_{2,n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \delta_{m,n} & b_m \end{array} \right]$$

Para se obter esta matriz, considere-se este exemplo trivial:

$$Exemplo_{expand} = \left[\begin{array}{cc|c} a_{1,1} & a_{1,2} & b_1 \\ a_{2,1} & a_{2,2} & b_2 \end{array} \right]$$

Para anular o elemento $a_{2,1}$, a segunda linha (L_2) é substituída por $a_{2,1}/a_{1,1} \times L_1 - L_2$, onde L_1 representa a primeira linha. Ao realizar tal operação o termo $a_{2,1}$ na matriz $Exemplo_{expand}$ é anulado. O algoritmo de triangularização da matriz irá operar em diagonal, trocando as linhas que apresentem um valor nulo na diagonal por alguma linha abaixo que possua um valor não-nulo na mesma coluna, e, a partir disto, anulando todos os coeficientes abaixo do elemento da diagonal atual.

A partir da matriz resultante de tal operação, o cálculo da solução é direto: basta resolver recursivamente da última linha para primeira o valor de cada incógnita.

O uso de um pivô determina que o sistema fará a escolha da linha de referência para triangularizar a matriz de acordo com o maior elemento existente da coluna. A linha com o maior elemento, passa a ser a linha cuja diagonal da matriz irá possuir tal elemento nesta coluna. Ao escolher tal pivô, as operações usam como referência um valor maior, o que permite diminuir os erros gerados pelas operações de ponto flutuante do software.

3.1.2 Método da Eliminação de Gauss-Jordan com e sem pivoteamento

No método de Gauss-Jordan, deseja-se obter uma matriz expandida equivalente ao sistema original, onde os coeficientes apresentam em toda diagonal elementos não-nulos. Este método complementa o método de eliminação de Gauss apresentado anteriormente, eliminando a etapa da resolução recursiva da matriz triangular, pois o sistema estará na forma:

$$A_{expand} = \left[\begin{array}{cccc|c} \delta_{1,1} & 0 & \cdots & 0 & \beta_1 \\ 0 & \delta_{2,2} & \cdots & 0 & \beta_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \delta_{m,n} & \beta_m \end{array} \right]$$

Onde a solução é imediata: $x_1 = \beta_1/\delta_{1,1}$.

Para obter esta matriz, o método inicia triangularizando a matriz e, em seguida, tomando a última linha, como referência elimina todos os elementos da última coluna dos coeficientes da matriz. Em seguida, repete o processo para a penúltima linha e assim sucessivamente.

3.1.3 Método da Decomposição LU e SVD

Decomposição LU

A resolução do método da decomposição LU se baseia na decomposição da matriz A de $A.x = b$ em duas outras matrizes L e U :

$$A = L.U$$

Onde L é uma matriz triangular inferior com diagonal unitária, ou seja, todos elementos acima da diagonal são nulos e os da diagonal são iguais a 1; e U é uma matriz triangular superior, onde todos os elementos abaixo da diagonal principal são nulos, de forma que:

$$LU.x = b$$

Dessa forma, a resolução do sistema $A.x = b$ se reduz à resolução de dois sistemas triangulares: $U.x = y$ e $L.y = b$. As matrizes L e U podem ser obtidas usando o processo de Gauss, onde a matriz U é a matriz resultante do processo e a matriz L é composta dos índices usados para multiplicar as linhas no processo e uma diagonal principal unitária.

Decomposição SVD

Quando a matriz A é singular, ou seja, não possui inversa (determinante igual a 0 no teste de Cramer), a decomposição SVD pode ser usada. Matrizes $A(M \times N)$ tal que M seja maior ou igual a N , podem ser decompostas no produto de três matrizes: $A = U.W.V^T$, tal que U é uma matriz coluna ortogonal; W é uma matriz diagonal com elementos positivos ou nulos (valores singulares); e V é uma matriz ortogonal.

Se A for uma matriz quadrada então U , W e V também o serão. Logo, de acordo com [3] podemos definir x como:

$$x = V.[diag(1/w_j)].(U^T.b) \quad (3.4)$$

Caso w_j seja nulo podemos substituir $1/w_j$ por zero.

3.1.4 Método da Decomposição de Cholesky e QR

Decomposição de Cholesky

Quando a matriz A é simétrica e positiva definida, ou seja $v.A.v > 0 \forall \text{vetor } v$, há uma decomposição triangular mais eficiente. A decomposição de Cholesky constrói uma matriz L diagonal inferior tal que L^T serve como matriz U , de modo que $L.L^T = A$.

Dessa forma a equação da decomposição LU torna-se:

$$L.L^T.x = b$$

Decomposição QR

A decomposição QR consiste em decompor a matriz A no produto QR onde R é uma matriz triangular superior e Q é uma matriz ortogonal, ou seja, $Q.Q^T = 1$. Substituindo na equação $Ax = b$ temos que:

$$\begin{aligned} Q.R.x &= b \\ R.x &= Q^T.b \end{aligned}$$

3.1.5 Método de Gauss-Jacobi

O método de Gauss-Jacobi fundamenta-se em resolver a equação:

$$Ax = b$$

Sabendo que a matriz A pode ser escrita na forma $A = D + (L + U)$ onde D representa a diagonal, L a matriz estritamente inferior e U a matriz estritamente superior. Substituindo o resultado na equação e isolando a variável x no primeiro membro, temos que:

$$x^{(k+1)} = D^{-1}[b - (L + U)x^{(k)}]$$

com k sendo o contador de iterações. Assim sendo, uma abordagem iterativa para a equação acima seria:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b - \sum_{j \neq i} a_{ij}x_j^{(k)})$$

Com o uso dessa equação os termos da matrix na iteração $k+1$ são calculados única e exclusivamente com os valores na iteração k . O processo converge quando a matriz é estritamente diagonal dominante, ou seja, o módulo do elemento na diagonal é maior que a soma do módulo de todos os outros elementos (não diagonais) da linha em questão. Mais precisamente:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \forall i$$

Há alguns casos onde o método converge sem satisfazer a condição anterior, sendo necessário que os elementos na diagonal principal sejam maiores em magnitude que os outros elementos.

3.1.6 Método de Gauss-Seidel

O método de Gauss-Seidel é uma melhora no método de Gauss-Jacobi, logo se baseia nos mesmos princípios. A principal diferença é que enquanto o método de Gauss-Jacobi usa única e exclusivamente os elementos da matriz da iteração anterior, o método de Gauss-Seidel utiliza os elementos já calculados na iteração corrente. Dessa forma a iteração de Gauss-Seidel é definida como sendo:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b - \sum_{j < i} a_{ij}x_j^{(k+1)} - \sum_{j > i} a_{ij}x_j^{(k)})$$

Assim como o método de Gauss-Jacobi, o método de Gauss-Seidel também converge para uma matriz A tal que A é estritamente diagonais dominantes. O método também converge para matrizes definidas positivas.

Capítulo 4

Manual do Desenvolvedor

A ferramenta **MSN - Resolução de Sistemas de Equações Lineares** foi feita em Java seguindo princípios básicos de desenvolvimento. Foram requisitos de desenvolvimento:

- Codificação de testes para os métodos
- Separação da lógica de negócio da lógica de interface
- Compatibilidade com o código da ferramenta desenvolvida no semestre anterior[1]
- Codificação em inglês
- Facilidade na instalação e uso

O código disponibiliza um arquivo `build.xml` a ser usado pelo Ant[4] para realizar tarefas comuns de desenvolvimento como:

clean Limpa o código gerado automaticamente pelo projeto (classes e documentação);

prepare Prepara o código para uso, criando os diretórios necessários;

javadoc Cria a documentação associada as classes do sistema;

make-jar Prepara um arquivo JAR executável;

test Testa o sistema;

release Prepara uma versão para uso contendo documentação e JAR.

Para executar qualquer uma destas tarefas, basta executar `ant tarefa` no diretório do projeto. Como pré-requisito o `ant` precisa estar devidamente configurado.

Tabela 4.1: Estrutura de Diretórios

Diretório	Descrição
src	Código fonte da aplicação
tests	Código de teste da aplicação
lib	Bibliotecas utilizadas no desenvolvimento da ferramenta
docs	Diretório com a documentação da ferramenta

4.1 Organização do Código

Todo o código segue a codificação UTF-8, permitindo maior compatibilidade entre os sistemas existentes. O sistema disponibiliza uma estrutura de diretórios para os recursos adequados referentes ao projeto:

O diretório `src` tem `br.edu.ufcg.msnlab` como pacote base da ferramenta, apresentando os seguintes sub-pacotes:

Tabela 4.2: Estrutura de Pacotes

Pacote	Descrição
br.edu.ufcg.msnlab.exceptions	Exceções dos métodos
br.edu.ufcg.msnlab.facade	Facade para comunicação da interface e métodos de resolução
br.edu.ufcg.msnlab.methods	Pacote para armazenar os métodos de resolução de sistemas
br.edu.ufcg.msnlab.util	Pacote com classes utilitárias comuns ao projeto

O diretório de testes segue uma mesma estrutura de pacotes, colocando as classes de teste no mesmo pacote a qual o teste se refere.

4.2 Testes de Unidade

Todos os métodos do sistema são testados através da classe `br.edu.ufcg.msnlab.MethodsAutomatedTest` do diretório `tests`. Esta classe gera matrizes que representam sistemas de maneira aleatória para em seguida realizar algumas manipulações algébricas em cima desse sistema e tentar resolvê-lo, se o resultado do sistema alterado for compatível com o resultado anterior desse sistema então o método obteve sucesso em sua resolução. Os testes são executados com diversos números de iterações além de diversos tipos de sistema com o objetivo de tornar a carga de testes mais extensa garantindo assim um funcionamento mais confiável dos métodos.

4.3 Arquitetura do Sistema

Seis componentes básicos descrevem a arquitetura do sistema:

GUI Componente responsável pela interface gráfica com o usuário

SystemLogic Lógica do sistema, responsável por invocar os métodos, fazer a checagem inicial e executar o *parser* das equações recebidas

SystemParser Componente para *parser* das equações recebidas pela interface

Checker Componente para a avaliação das soluções das matrizes de acordo com a regra de Cramer

Solver Componente de interface para invocação dos métodos do sistema

Methods Componente com os métodos do sistema

A interação estática destes elementos é descrita na figura 4.1.

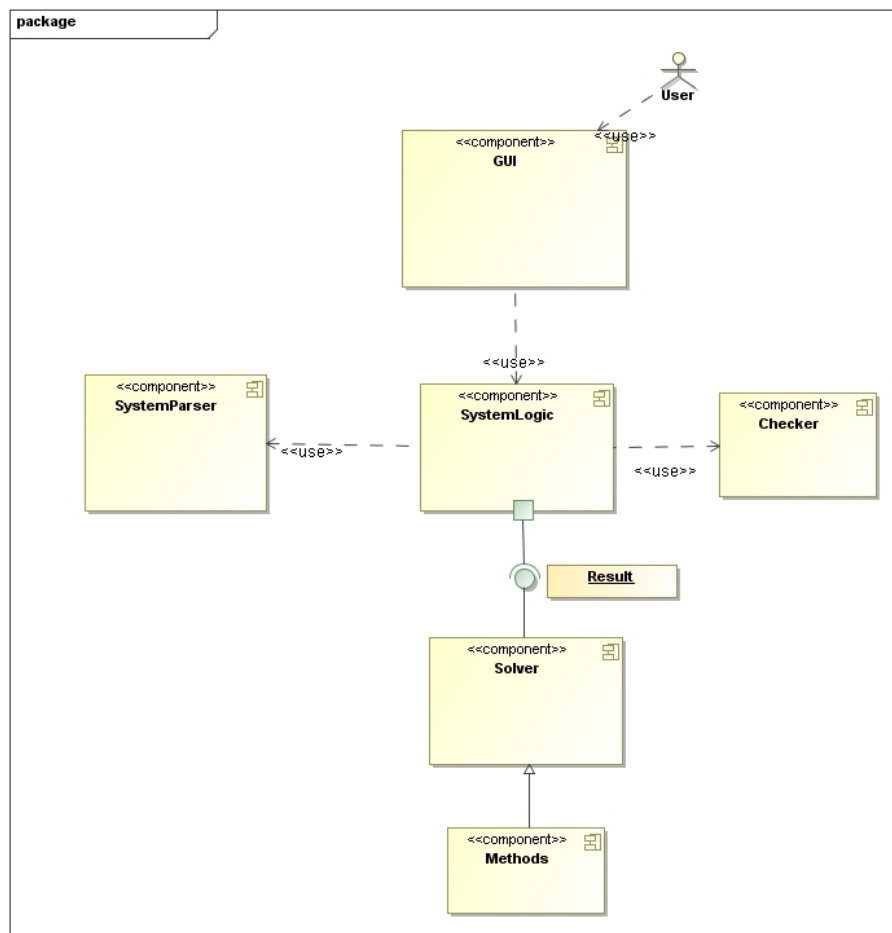


Figura 4.1: Arquitetura do Sistema

4.3.1 Comunicação entre os módulos

Interação dinâmica dos elementos é representada no diagrama de sequência da figura 4.2. Cada elemento atua como as seguintes responsabilidades:

Parser Recebe como entrada um sistema em formato String, retorna um objeto `ParsedSystem` que representa o sistema após o parser;

Checker Recebe o sistema representado através de duas matrizes, uma de coeficientes e outra de termos independentes. Retorna uma String que representa a classificação daquele sistema para definição de tipos de resolução;

Solver Recebe as matrizes que compõem o sistema, o número de iterações, o valor estimado pelo usuário, a matriz de estimativas (para os métodos iterativos) e um objeto de Configuração. Este objeto `Config` contém possíveis configurações que são usados pelos métodos, por exemplo, se o método pode ser feito com ou sem pivoteamento;

Facade A facade retorna um objeto `Result` que contém o passo a passo da resolução feita pelo método.

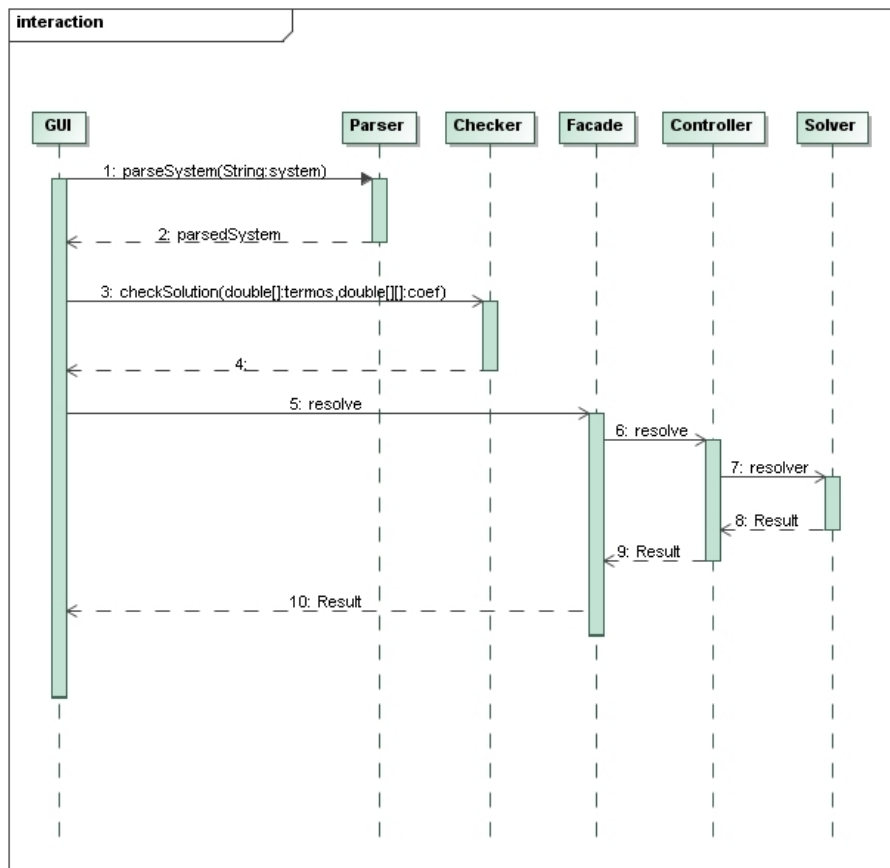


Figura 4.2: Fluxo de Execução do Sistema

4.4 Guia de Usabilidade da Interface

Este guia levanta requisitos a serem usados na interface gráfica na forma de uma lista de fatores de usabilidade a serem adotados na construção da interface.

- Diálogo simples e claro
- Linguagem do usuário
- Minimizar carga de memória
- Consistência
- Feedback
- Atalhos
- Documentação e ajuda
- Saídas claras
- Boas mensagens de erro
- Evitar erros

Referências Bibliográficas

- [1] MSN 2008.1. Msn 2008.1. <http://code.google.com/p/msnlab/>, 2008.
- [2] Inc. Free Software Foundation. Gpl 2.0. <http://www.gnu.org/licenses/gpl-2.0.txt>, 1989, 1991.
- [3] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.
- [4] The Apache Ant Project. The apache ant project. <http://ant.apache.org/>, 2008.
- [5] Tigris.org. Subversion. <http://subversion.tigris.org/>, 2008.
- [6] Eric W. Weisstein. Cramer’s rule. MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/CramersRule.html>, 2008.