CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

ESTUDO COMPARATIVO SOBRE MÉTODOS ITERATIVOS DE RESOLUÇÃO DE SISTEMAS LINEARES DE GRANDE PORTE

LUÍS RICARDO FERNANDES

Orientador: Dr. Rodrigo Tomás Nogueira Cardoso

Coorientador: Dr. Carlos Magno Martins Cosme

BELO HORIZONTE
JULHO DE 2017

LUÍS RICARDO FERNANDES

ESTUDO COMPARATIVO SOBRE MÉTODOS ITERATIVOS DE RESOLUÇÃO DE SISTEMAS LINEARES DE GRANDE PORTE

Dissertação de Mestrado submetido ao Programa de Pós-graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Área de concentração: Modelagem Matemática e Computacional

Linha de pesquisa: Métodos Matemáticos Aplicados

Orientador: Dr. Rodrigo Tomás Nogueira Cardoso

Coorientador: Dr. Carlos Magno Martins Cosme

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL
BELO HORIZONTE
JULHO DE 2017

Fernandes, Luís Ricardo
F363e Estudo comparativo sobre métodos iterativos de resolução de sistemas lineares de grande porte. / Luís Ricardo Fernandes. – – Belo Horizonte, 2017.
xii, 78 f. : il.

Dissertação (mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais, Programa de Pós-Graduação em Modelagem Matemática e Computacional, 2017.
Orientador: Prof. Dr. Rodrigo Tomás Nogueira Cardoso
Coorientador: Prof. Dr. Carlos Magno Martins Cosme

Bibliografia

1. Sistemas Lineares. 2. Métodos Iterativos (Matemática). 3.Modelos Matemáticos. I. Cardoso, Rodrigo Tomás Nogueira. II. Centro Federal de Educação Tecnológica de Minas Gerais. III. Título

CDD 511.8

Elaboração da ficha catalográfica pela Biblioteca-Campus II / CEFET-MG

Agradecimentos

Agradeço primeiramente a Deus, pelo dom da vida e por estar presente em todos os momentos, bons ou ruins.

Agradeço especialmente à minha mãe Glória, que sempre me incentivou, me apoiou e esteve comigo. Aos meu irmãos Ênio e Heitor, minha avó Judith e toda minha família que, de alguma forma, participaram dessa jornada. A vocês Cinara, Ramana, Querubina, Adriana, Patrícia e Sandra por serem presentes em minha vida.

Agradeço aos meus colegas de graduação Taís, Francielle, Amanda, Dayane, Gustavo, Adeliene, Simone, Taciany, Tarcyelle, Fabiane. Às minhas colegas de Iniciação Científica Thaís e Hoyama. Vocês são muito importante para mim.

Agradeço aos professores da graduação, em especial Beth, Laudo e Niltom. Grandes mestres.

Aos meus amigos do mestrado Samara, Alexandre, Dângelo, Gislane, Priscila, Renan, Flávia, Silvana e Gustavo. Em especial, agradeço à Rafaela e Aline que, por tantas vezes, me acolheram em suas residências. Muito obrigado mesmo.

Agradeço ao meu orientador Rodrigo por ter aceito me conduzir nessa jornada e, em especial, ao meu coorientador Carlos Magno pela paciência e conhecimento transmitido. Sou muito grato a vocês.

Aos professores e funcionários do CEFET-MG.

Agradeço à CAPES pelo apoio financeiro durante parte do curso.

A todos vocês, muito obrigado.

Resumo

Esta dissertação apresenta um estudo sobre métodos iterativos para resolução de sistemas lineares de grande porte não simétricos. Foram estudados três métodos: GMRES, CMRH e LCD. Os métodos GMRES e CMRH são baseados em espaços de $Krylov$. A principal diferença entre eles é a forma como a base desses espaços é construída. Enquanto o GMRES utiliza o método de Arnoldi, através do processo de ortogonalização de Gram-Schmidt, o método CMRH utiliza o processo de Hessenberg, via fatoração LU. O método LCD é baseado em vetores de direções conjugadas à esquerda. Uma versão do método CMRH, denominado CMRH-OVER, que inclui um processo de sobre-armazenamento, também é abordado. Em todos os métodos, exceto o CMRH-OVER, aplica-se um processo de reinicialização cujo intuito é diminuir o custo computacional em armazenar os vetores que formam a base do espaço. Com o objetivo de comparar o desempenho computacional, seja o tempo de execução ou a quantidade de iterações necessárias para calcular a solução do sistema, os métodos foram testados em matrizes genéricas e em um problema aplicado, através da equação de convecção-difusão transiente, resolvida numericamente via elementos finitos. A solução numérica dessa equação também é apresentada neste trabalho.

Palavras-chave: Sistemas Lineares. Métodos Iterativos. Equação de convecção-difusão-reação transiente.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Símbolos

$\mathcal{K}_k$	Espaço de $Krylov$ de ordem k
H_k	Matriz de Hessenberg de dimensão $k \times k$
$\bar{H}_k$	Matriz com dimensão $(k+1) \times k$ cujos elemntos são os mesmos de H_k acrescido de uma linha.
V_k	Matriz formada pelos vetores que formam a base para o espaço de $Krylov$
$\boldsymbol{e_1}$	Vetor $[1\ 0\ \cdots\ 0]^T$
$\boldsymbol{r_0}$	Resíduo inicial: $b - Ax_0$
$\boldsymbol{x_0}$	Solução inicial
β	Norma do resíduo inicial: $\|\|r_0\|\|$
F_j	Matriz de rotação de Givens
Q_k	Matriz da fatoração QR
k_{max}	Parâmetro que determina a quantidade de vetores que irão compor a base para o subespaço de $Krylov$
l_{max}	Parâmetro que determina a quantidade de vezes que cada método irá reinicializar
I_j	Matriz identidade com dimensão $j \times j$
L_j	Matriz L proveniente da fatoração LU
U_j	Matriz U proveniente da fatoração LU
$\boldsymbol{p_k}$	Vetor de permutações
Γ_D	Fronteira onde estão prescritas condições de contorno de Dirichlet
Γ_N	Fronteira onde estão prescritas condições de contorno de Neumann
β	Campo de velocidades
f	Função de fonte
Ω	Domínio aberto com fronteira poligonal Γ.

neq	Número de equações
nel	Número de elementos da malha
$nnos$	Número de nós da malha

Sumário

Capítulo 1

Introdução

A resolução de sistemas lineares, um dos problemas fundamentais da álgebra, possui várias aplicações em diferentes áreas do conhecimento. Esses sistemas estão presentes no cálculo de tensões e correntes em um circuito elétrico, na solução de equações diferenciais via métodos numéricos assim como na determinação do esforço mecânico em cada viga de uma estrutura metálica, entre outros. Para exemplificar um problema em que aparece a resolução de sistemas lineares, será mostrado o balanceamento de uma equação química, forma de se descrever uma reação química envolvendo reagentes e produtos. No caso, tem-se a reação do carbonato férrico $(Fe_2(CO_3)_3)$ com ácido sulfúrico (H_2SO_4) transformados em sulfato férrico $(Fe_2(SO_4)_3)$, água (H_2O) e gás carbônico (CO_2s). Essa reação é descrita na equação 1:

$$\mathbf{x_1}Fe_2(CO_3)_3 + \ \mathbf{x_2}H_2SO_4 \longrightarrow \mathbf{x_3}Fe_2(SO_4)_3 + \ \mathbf{x_4}H_2O + \ \mathbf{x_5}CO_2. \tag{1}$$

Com variáveis $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_5}$, tem-se o sistema linear oriundo da equação 1:

$$\begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 3 & 0 & 0 & 0 & -1 \\ 9 & 4 & -12 & -1 & -2 \\ 0 & 2 & 0 & -2 & 0 \\ 0 & 1 & -3 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \mathbf{x_3} \\ \mathbf{x_4} \\ \mathbf{x_5} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A solução deste sistema informa a quantidade de cada substância presente no reagente e no produto dessa reação química. Sistemas como esse são observados na solução de diferentes tipos de problemas e buscar a forma mais eficiente de resovê-los é uma das razões que motivam este trabalho.

A definição de um sistema linear pode ser dada da seguinte maneira: seja $A \in \mathbb{R}^{n \times n}$ uma matriz não singular, $\boldsymbol{x} \in \mathbb{R}^n$ um vetor de incógnitas e $\boldsymbol{b} \in \mathbb{R}^n$, um sistema linear pode ser

escrito como

$$A\boldsymbol{x} - \boldsymbol{b} = \mathbf{0}. \tag{2}$$

A não singularidade de A garante que esse sistema tenha solução única. Para resolver esses sistemas são utilizados métodos diretos ou métodos iterativos. Os métodos diretos encontram a solução exata do sistema enquanto os métodos iterativos encontram um solução aproximada. Entre os métodos diretos mais conhecidos estão eliminação Gaussiana, fatoração LU ou fatoração de Cholesky (BURDEN; FAIRES, 2011). Contudo, dependendo da estrutura e da dimensão da matriz, os métodos diretos tornam-se inviáveis devido ao alto custo computacional que apresentam. Já os métodos iterativos partem de uma solução inicial $\boldsymbol{x_0}$ e criam uma sequência de soluções que convergem para a solução exata do sistema. Para matrizes simétricas, o método Gradiente Conjugado é eficiente. Para matrizes não simétricas existem diferentes métodos na literatura dos quais, neste trabalho, três serão estudados, todos baseados em espaços de $Krylov$.

O GMRES (Generalized Minimal Residual Method)(SAAD; SCHULTZ, 1986) usa o método de Arnoldi, através do processo de ortogonalização de Gram-Schmidt, para construção de uma base para o espaço de $Krylov$, enquanto o método CMRH (Changing Minimal Residual Method Based on the Hessenberg Process) (SADOK, 1999) usa o processo de Hessenberg via fatoração LU. Outra versão do método CMRH (CMRH-OVER), com sobrearmazenamento, proposta por (HEYOUNI; SADOK, 2008) também é estudado neste trabalho. O terceiro método estudado, LCD (Left Conjugate Directions) (DAI; YUAN, 2004), constrói vetores de direções conjugadas à esquerda para calcular a solução do sistema. Todos os métodos apresentados encontram uma solução aproximada para sistemas lineares não simétricos de grande porte.

Equações diferenciais parciais possuem, em geral, solução analítica complexa. Por isso, é comum usar-se ferramentas numéricas para encontrar uma solução aproximada para essas equações, como Método de Elementos Finitos (MEF) (ZIENKIEWICZ et al., 1977) (SOLIN, 2006), Método de Diferenças Finitas (THOMAS, 2013) e Método de Volumes Finitos (EYMARD; GALLOUËT; HERBIN, 2000). Na aplicação dessas ferramentas aparecem sistemas lineares que, dependendo da discretização do espaço das funções, resultam em matrizes esparsas e de grande porte. Além disso, essas funções podem ser dependentes do tempo. Por isso, em cada iteração, haverá um sistema linear para ser resolvido. Surge então a necessidade de haver métodos que sejam eficientes para a resolução desses sistemas.

Neste trabalho será estudada a equação de convecção-difusão-reação-transiente via MEF. Este é um tipo de equação diferencial presente em diversos problemas de modelagem: escoamento de fluidos (CROCHET; DAVIES; WALTERS, 2012), (WERNER, 2011), turbulência (ILINCA; PELLETIER, 1998), entre outros. A discretização temporal da equação será feita pelo método de diferenças finitas de segunda ordem de Crank-Nicolson, enquanto a

espacial será feita via MEF. Com isso, a solução da equação recai na resolução de sistemas lineares.

Inicialmente, os métodos LCD, GMRES, CMRH e CMRH-OVER serão aplicados em matrizes genéricas, como realizado por (SADOK; SZYLD, 2012), com o objetivo de avaliar seu desempenho computacional. Em seguida, a comparação dos métodos será feita através de uma aplicação em um problema de Elementos Finitos.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo deste trabalho é realizar um estudo comparativo de três métodos iterativos de resolução de sistemas lineares de grande porte, através do tempo de execução e da quantidade de iterações que os métodos levam para convergir.

1.1.2 Objetivos Específicos

1. Apresentar a fundamentação teórica sobre os métodos iterativos para resolução de sistemas lineares de grande porte: LCD, GMRES e CMRH.

2. Apresentar o MEF para a resolução do problema de convecção-difusão-reação-transiente usando discretização temporal através de diferenças finitas de Crank-Nicolson.

3. Fazer um estudo comparativo entre os métodos estudados.

4. Apresentar os resultados obtidos avaliando a acurácia e a eficiência dos métodos.

1.2 Organização do Trabalho

Esta Dissertação de Mestrado encontra-se divida em capítulos. No primeiro capítulo é apresentada a introdução do trabalho, assim como a definição do problema e quais são os objetivos a serem alcançados.

O capítulo 2 apresenta os três métodos estudados neste trabalho: GMRES, CMRH e LCD, assim como os algoritmos correspondentes.

O capítulo 3 apresenta a equação de convecção-difusão-reação-transiente, sua formulação variacional e o MEF, bem como os aspectos computacionais necessários para sua resolução.

O capítulo 4 apresenta os resultados dos experimentos numéricos assim como a aplicação no problema de Elementos Finitos.

Por fim, o capítulo 5 expõe as conclusões e trabalhos futuros.

Capítulo 2

Métodos Iterativos para Sistemas Lineares não Simétricos

De modo geral, os métodos iterativos para resolução de sistemas lineares do tipo $A\boldsymbol{x}=\boldsymbol{b}$, sendo A não singular, consistem na geração de uma sequência finita de aproximações $\boldsymbol{x_1}$, $\boldsymbol{x_2}$, $\boldsymbol{x_3}$, $\cdots$, $\boldsymbol{x_n}$ a partir de uma solução inicial $\boldsymbol{x_0}$. A não singularidade da matriz A garante a unicidade da solução, portanto ao longo de todo o texto, A será sempre uma matriz não singular. Dois dos métodos estudados neste trabalho, GMRES e CMRH, são baseados em projeções em espaços de $Krylov$, sendo descritos a seguir.

Definição 2.0.1. *Seja $A \in \mathbb{R}^{n\times n}$ e $\boldsymbol{v} \in \mathbb{R}^n, \boldsymbol{v} \neq \boldsymbol{0}$. O espaço*

$$\mathcal{K}_k(A, \boldsymbol{v}) = span\{\boldsymbol{v}, A\boldsymbol{v}, A^2\boldsymbol{v}, \cdots, A^{k-1}\boldsymbol{v}\} \tag{3}$$

é chamado de espaço de $Krylov$ de ordem k, associado a A e $\boldsymbol{v}$.

Sendo $\boldsymbol{x_0}$ uma aproximação inicial para a solução do sistema $A\boldsymbol{x} = \boldsymbol{b}$, define-se o resíduo inicial desta aproximação como $\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$. Os métodos de $Krylov$ constroem a sequência de aproximações por

$$\boldsymbol{x_k} = \boldsymbol{x_0} + \boldsymbol{z_k}, \tag{4}$$

sendo que $\boldsymbol{z_k} \in \mathcal{K}_k(A, \boldsymbol{r_0})$ e minimiza a norma do resíduo $\boldsymbol{r_k} = \boldsymbol{b} - A\boldsymbol{x_k}$ sobre este espaço. O produto interno usual em $\mathbb{R}^n$ é representado por $\langle\cdot,\cdot\rangle$, enquanto a norma euclidiana é representada por $||\cdot||$. Uma dificuldade que surge nesse processo é exatamente a minimização da norma do resíduo sobre o espaço de $Krylov$. Uma estratégia para contornar este problema é, a partir da construção de bases especiais para $\mathcal{K}_k$, transformar o problema de minimização sobre $\mathcal{K}_k$ em um problema de minimização sobre $\mathbb{R}^n$. Essa técnica é empregada tanto no GMRES quanto no CMRH.

2.1 GMRES

Como mencionado anteriormente, deve-se construir uma base para o espaço $\mathcal{K}_k(A, \boldsymbol{r_0})$. O método GMRES faz este processo empregando o método de Arnoldi (SAAD, 2003), uma variação do método de Gram-Schmidt (HOFFMAN; KUNZE, 1971), construindo uma base ortonormal $\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_k}$ para $\mathcal{K}_k(A, \boldsymbol{r_0})$. O vetor $\boldsymbol{v_1}$ é obtido normalizando-se $\boldsymbol{r_0}$, ou seja,

$$\boldsymbol{v_1} = \frac{\boldsymbol{r_0}}{||\boldsymbol{r_0}||}.$$

Desta maneira, obtém-se uma base para o subespaço $\mathcal{K}_1(A, \boldsymbol{r_0})$. A fim de se obter uma base ortonormal para $\mathcal{K}_{j+1}(A, \boldsymbol{r_0})$ a partir da base ortonormal $\mathcal{K}_j(A, \boldsymbol{r_0})$, o vetor $A\boldsymbol{v_j}$ de $\mathcal{K}_{j+1}(A, \boldsymbol{r_0})$ será ortonormalizado com relação ao espaço $\mathcal{K}_j(A, \boldsymbol{r_0})$ da forma como segue. Seja $h_{i,j} = \langle \boldsymbol{v_i}, A\boldsymbol{v_j} \rangle$, então

$$\hat{\boldsymbol{v}}_{j+1} = A\boldsymbol{v_j} - \sum_{i=1}^{j} h_{ij}\boldsymbol{v_i}. \tag{5}$$

O vetor $\boldsymbol{v_{j+1}}$ é dado por

$$\boldsymbol{v_{j+1}} = \frac{\hat{\boldsymbol{v}}_{j+1}}{||\hat{\boldsymbol{v}}_{j+1}||}. \tag{6}$$

Tal procedimento encontra-se descrito no algoritmo a seguir.

Algoritmo 1: Método de Arnoldi

Entrada: Matriz A, vetores $\boldsymbol{x_0}$ e $\boldsymbol{b}$ e dimensão k.
$\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$
$\boldsymbol{v_1} = \boldsymbol{r_0}/||\boldsymbol{r_0}||$
para $j = 1, 2, \cdots, k$ **faça**
  $\boldsymbol{v_{j+1}} = A\boldsymbol{v_j}$
  para $i = 1, 2, \cdots, j$ **faça**
    $h_{ij} = \langle \boldsymbol{v_i}, A\boldsymbol{v_j} \rangle$
    $\boldsymbol{v_{j+1}} = \boldsymbol{v_{j+1}} - h_{ij}\boldsymbol{v_i}$
  fim
  $h_{j+1,j} = ||\boldsymbol{v_{j+1}}||$
  se $h_{j+1,j} = 0$ **então**
    Pare
  fim
  $\boldsymbol{v_{j+1}} = \boldsymbol{v_{j+1}}/h_{j+1,j}$
fim

O algoritmo 1, além de construir uma base para o espaço $\mathcal{K}_k(A, \boldsymbol{r_0})$ através do processo de Arnoldi, constrói também uma matriz de Hessenberg H_k (GOLUB; LOAN, 1989) com

dimensão $k \times k$ definida por

$$(H_k)_{ij} = h_{ij} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1k} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2k} \\ 0 & h_{32} & h_{33} & \cdots & h_{3k} \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & h_{k,k-1} & h_{kk} \end{bmatrix}$$

além do elemento $h_{k+1,k}$ que não está presente em H_k. A matriz H_k vem da definição de $h_{ij} = \langle A\boldsymbol{v_j}, \boldsymbol{v_i} \rangle$. Os vetores $\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_k}$ são tais que:

$i)$ $\mathcal{K}_k(A, \boldsymbol{r_0}) = span\{\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_k}\}$;

$ii)$ $\langle \boldsymbol{v_i}, \boldsymbol{v_j} \rangle = \delta_{ij}$ e $||\boldsymbol{v_i}|| = 1$.

Isto é uma consequência direta do processo de ortogonalização de Gram-Schmidt, uma vez que os vetores $\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_k}$ são ortonormais. Em seguida, será mostrado como a matriz H_k e a base $\{\boldsymbol{v_1}, \cdots, \boldsymbol{v_k}\}$ resolverão o problema de minimização do resíduo $\boldsymbol{r_k}$.

Seja $V_k = [\boldsymbol{v_1}\ \boldsymbol{v_2}\ \cdots\ \boldsymbol{v_k}]_{n \times k}$ a matriz cujas colunas são os vetores $\boldsymbol{v_i}$. É possível verificar que

$$H_k = V_k^T A V_k. \tag{7}$$

De fato, observe que

$$AV_k = [A\boldsymbol{v_1}\ A\boldsymbol{v_2}\ \cdots\ A\boldsymbol{v_k}]$$

e

$$V_k^T = \begin{bmatrix} \boldsymbol{v_1}^T \\ \boldsymbol{v_2}^T \\ \vdots \\ \boldsymbol{v_k}^T \end{bmatrix}.$$

Se $i \leq j + 1$, então

$$(H_k)_{ij} = \boldsymbol{v_i}^T \cdot A\boldsymbol{v_j} = \langle \boldsymbol{v_i}, A\boldsymbol{v_j} \rangle = \langle A\boldsymbol{v_j}, \boldsymbol{v_i} \rangle.$$

Portanto, pode-se concluir que

$$(H_k)_{i,j} = h_{ij}.$$

Das equações (5) e (6), tem-se

$$A\boldsymbol{v_j} - \sum_{i=1}^{j} h_{ij} v_i = \boldsymbol{v_{j+1}} \cdot ||\hat{\boldsymbol{v}}_{j+1}||.$$

Como $h_{j+1,j} = ||\hat{\boldsymbol{v}}_{j+1}||$, então

$$A\boldsymbol{v_j} = \sum_{i=1}^{j} h_{i,j}\boldsymbol{v_i} + \boldsymbol{v_{j+1}} h_{j+1,j} \tag{8}$$

$$= \sum_{i=1}^{j+1} h_{i,j}\boldsymbol{v_i}, \quad \forall j = 1, \cdots, k. \tag{9}$$

Dessa forma

$$\langle A\boldsymbol{v_j}, \boldsymbol{v_i}\rangle = \langle \sum_{k=1}^{j+1} h_{kj}\boldsymbol{v_k}, \boldsymbol{v_i}\rangle = \sum_{k=1}^{j+1} h_{kj}\langle \boldsymbol{v_k}, \boldsymbol{v_i}\rangle.$$

Se $i \geq j+2$, então $i \neq k$. Neste caso, $\boldsymbol{v_k}$ e $\boldsymbol{v_i}$ são ortonormais e assim $\langle \boldsymbol{v_k}, \boldsymbol{v_i}\rangle = 0$. Portanto, para $i \geq j+2$,

$$\langle V^T A V\rangle_{ij} = (H_k)_{ij} = 0,$$

e isso prova a identidade (7). Seja $\bar{H}_k$ a matriz com dimensão $(k+1) \times k$ cujos elementos são os mesmos da matriz H_k, exceto por uma linha adicional cujo único elemento não nulo está na posição $(k+1) \times k$. A matriz $\bar{H}_k$ e os vetores $\boldsymbol{v_i}$ satisfazem a relação:

$$AV_k = V_{k+1}\bar{H}_k. \tag{10}$$

Com o objetivo de resolver o sistema $A\boldsymbol{x} = \boldsymbol{b}$, pode-se aplicar uma técnica de projeção ortogonal sobre o espaço de $Krylov$ $\mathcal{K}_k(A, \boldsymbol{r_0})$. Para isso, seja $\boldsymbol{x_0}$ uma aproximação inicial para a solução do sistema e a aproximação na k-ésima iteração definida como $\boldsymbol{x_k} = \boldsymbol{x_0} + \boldsymbol{z_k}$, sendo $\boldsymbol{z_k} \in \mathcal{K}_k(A, \boldsymbol{r_0})$ obtido de modo a se garantir que o resíduo $\boldsymbol{r_k} = \boldsymbol{b} - A\boldsymbol{x_k}$ seja ortogonal a $\mathcal{K}_k(A, \boldsymbol{r_0})$. Como $\boldsymbol{z_k} \in \mathcal{K}_k(A, \boldsymbol{r_0})$ e os vetores $\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_k}$ são linearmente independentes, pode-se então escrever $\boldsymbol{z_k}$ como uma combinação linear de V_k. Desse modo, tem-se

$$\boldsymbol{z_k} = \sum_{j=1}^{k} y_j \boldsymbol{v_j} \tag{11}$$

$$\boldsymbol{z_k} = V_k \boldsymbol{y_k}, \tag{12}$$

sendo $\boldsymbol{y_k} = (y_1, y_2, \cdots, y_k)^T \in \mathbb{R}^k$. Nesse caso,

$$\boldsymbol{r_k} = \boldsymbol{b} - A\boldsymbol{x_k} = \boldsymbol{b} - A(\boldsymbol{x_0} + \boldsymbol{z_k}) = \boldsymbol{r_0} - AV_k\boldsymbol{y_k}.$$

Como o resíduo é ortogonal a $\mathcal{K}_k$, então

$$V_k^T \boldsymbol{r_k} = 0.$$

Sendo assim,

$$0 = V_k^T(\boldsymbol{r_0} - AV_k\boldsymbol{y_k}) \quad \Rightarrow \quad V_k^T\boldsymbol{r_0} = V_k^T A V_k \boldsymbol{y_k} = H_k\boldsymbol{y_k}. \tag{13}$$

Sabendo que $\boldsymbol{v_1} = \dfrac{\boldsymbol{r_0}}{||\boldsymbol{r_0}||}$,

$$\begin{aligned} H_k \boldsymbol{y_k} &= V_k^T \boldsymbol{r_0} && (14)\\ &= V_k^T ||\boldsymbol{r_0}|| v_1 && (15)\\ &= ||\boldsymbol{r_0}|| V_k^T v_1 && (16)\\ &= ||\boldsymbol{r_0}|| e_1, && (17) \end{aligned}$$

e sendo $\boldsymbol{e_1} = (1\ 0 \cdots 0)^T$ tem-se que

$$\boldsymbol{y_k} = H_k^{-1} ||\boldsymbol{r_0}|| \boldsymbol{e_1}. \quad (18)$$

Pode-se então definir o seguinte algoritmo, uma estratégia para resolução do sistema $A\boldsymbol{x} = \boldsymbol{b}$.

Algoritmo 2: Método de Ortogonalização Completa.

Entrada: Matriz A, vetores $\boldsymbol{x_0}$ e $\boldsymbol{b}$.
$\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$
$\boldsymbol{v_1} = \boldsymbol{r_0}/||\boldsymbol{r_0}||$
para $j = 1, 2, \cdots, k$ **faça**
  para $i = 1, 2, \cdots, j$ **faça**
    $h_{i,j} = \langle A\boldsymbol{v_j}, \boldsymbol{v_i} \rangle$
  fim
  $\hat{\boldsymbol{v}}_j + 1 = A\boldsymbol{v_j} - \sum_{i=1}^{j} h_{i,j} \boldsymbol{v_i}$
  $h_{j+1,j} = ||\hat{\boldsymbol{v}}_{j+1}||$
  se $\boldsymbol{v_{j+1}} = 0$ **então**
    Pare
  fim
  $\boldsymbol{v_{j+1}} = \hat{\boldsymbol{v}}_{j+1}/h_{j+1,j}$
fim
$\boldsymbol{x_k} = \boldsymbol{x_0} + V_k \boldsymbol{y_k}$, sendo $\boldsymbol{y_k} = H_k^{-1} ||\boldsymbol{r_0}|| \boldsymbol{e_1}$

Uma dificuldade encontrada no algoritmo (2) é a inversão da matriz H_k que, mesmo sendo uma matriz de Hessenberg, pode ter um alto custo computacional. Para contornar esse problema de inversão da matriz, o método GMRES obtém a melhor solução aproximada $\boldsymbol{x_k}$ minimizando o resíduo $\boldsymbol{r_k}$ através da solução do problema de mínimos quadrados.

$$\begin{aligned} \min\{||\boldsymbol{b} - A\boldsymbol{x}||;\ \boldsymbol{x} = \boldsymbol{x_0} + \boldsymbol{z}, \boldsymbol{z} \in \mathcal{K}_k\} &= \min_{\boldsymbol{z} \in \mathcal{K}_k} ||\boldsymbol{b} - A\boldsymbol{x_0} - A\boldsymbol{z}|| && (19)\\ &= \min_{\boldsymbol{z} \in \mathcal{K}_k} ||\boldsymbol{r_0} - A\boldsymbol{z}||. && (20) \end{aligned}$$

Se $\boldsymbol{z} \in \mathcal{K}_k$, então existe $\boldsymbol{y} \in \mathbb{R}^n$ tal que $\boldsymbol{z} = V_k \boldsymbol{y}$ e pode-se ver a norma a ser minimizada como o seguinte funcional de $\boldsymbol{y}$:

$$J(\boldsymbol{y}) = \|\beta \boldsymbol{v_1} - A V_k \boldsymbol{y}\|, \quad (21)$$

sendo $\beta = ||\boldsymbol{r_0}||$. Sabendo que $\beta \boldsymbol{v_1} = V_{k+1}\beta \boldsymbol{e_1}$, isto porque o produto matriz vetor $V_{k+1}\boldsymbol{e_1}$ resulta na primeira linha da matriz V_{k+1}, que é justamente o próprio $\boldsymbol{v_1}$, tem-se das equações (10) e (21) que

$$\begin{aligned} J(\boldsymbol{y}) &= ||\beta \boldsymbol{v_1} - V_{k+1}\bar{H}_k \boldsymbol{y}|| && (22) \\ &= \|V_{k+1}[\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{y}]\| && (23) \\ &= \|V_{k+1}\| \ \|\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{y}\|. && (24) \end{aligned}$$

Como V_{k+1} é ortogonal, o funcional a ser minimizado é

$$J(\boldsymbol{y}) = \|\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{y}\|. \quad (25)$$

A solução para o problema de mínimos quadrados é dada por

$$\boldsymbol{x_k} = \boldsymbol{x_0} + V_k \boldsymbol{y_k}, \quad (26)$$

sendo que $\boldsymbol{y_k}$ minimiza o funcional $J(\boldsymbol{y})$ sobre $\boldsymbol{y} \in \mathbb{R}^k$. A minimização desse funcional é computacionalmente menos cara que a inversão da matriz H_k.

Algoritmo 3: GMRES

Entrada: Matriz A, vetores $\boldsymbol{x_0}$ e $\boldsymbol{b}$, dimensão do subespaço k.
$\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$
$\boldsymbol{v_1} = \boldsymbol{r_0}/||\boldsymbol{r_0}||$
para $j = 1, 2, \cdots, k$ **faça**
  para $i = 1, 2, \cdots, j$ **faça**
    $h_{i,j} = \langle A\boldsymbol{v_j}, \boldsymbol{v_i} \rangle$
  fim
  $\hat{\boldsymbol{v}}_j + 1 = A\boldsymbol{v_j} - \sum_{i=1}^{j} h_{i,j}\boldsymbol{v_i}$
  $h_{j+1,j} = ||\hat{\boldsymbol{v}}_{j+1}||$
  se $h_{j+1,j} = 0$ **então**
    Pare
  fim
  $\boldsymbol{v_{j+1}} = \hat{\boldsymbol{v}}_{j+1}/h_{j+1,j}$
fim
$\boldsymbol{x_k} = \boldsymbol{x_0} + V_k \boldsymbol{y_k}$, sendo que $\boldsymbol{y_k}$ minimiza $||\beta \boldsymbol{v_1} - V_{k+1}\bar{H}_k \boldsymbol{y}||$.

O custo computacional cresce significativamente à medida que a dimensão do subespaço de $Krylov$ aumenta. Isso se deve ao espaço de memória utilizado para armazenar os vetores $\boldsymbol{v_i}$ e à quantidade de operações $flop$ necessárias para execução do método. Para resolver esse problema, utiliza-se a versão GMRES reinicializado -GMRES(m)- que, após m passos, reinicializa o algoritmo tomando como aproximação inicial o último valor $\boldsymbol{x_k}$ calculado. Nesse caso, a quantidade de vetores que precisa ser armazenada é condicionada à quantidade de iterações que o método irá executar antes de ser reinicializado. Consequentemente, a quantidade de $flops$ também será menor uma vez que se tem uma menor quantidade de vetores armazenados. O algoritmo com reinicialização é apresentado a seguir.

Algoritmo 4: GMRES(m)

Entrada: Matriz A, vetores $\boldsymbol{x_0}$ e $\boldsymbol{b}$, iterações m.

Passo 1:

$\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$

$\boldsymbol{v_1} = \boldsymbol{r_0}/||\boldsymbol{r_0}||$

Passo 2:

para $j = 1, 2, \cdots, m$ **faça**

  para $i = 1, 2, \cdots, j$ **faça**

    $h_{i,j} = \langle A\boldsymbol{v_j}, \boldsymbol{v_i} \rangle$

  fim

  $\hat{\boldsymbol{v}}_j + 1 = A\boldsymbol{v_j} - \sum_{i=1}^{j} h_{i,j}\boldsymbol{v_i}$

  $h_{j+1,j} = ||\hat{\boldsymbol{v}}_{j+1}||$

  se $h_{j+1,j} = 0$ **então**

    Pare

  fim

  $\boldsymbol{v_{j+1}} = \hat{\boldsymbol{v}}_{j+1}/h_{j+1,j}$

fim

$\boldsymbol{x_k} = \boldsymbol{x_0} + V_k\boldsymbol{y_k}$, sendo que $\boldsymbol{y_k}$ minimiza $||\beta\boldsymbol{v_1} - V_{k+1}\bar{H}_k\boldsymbol{y}||$.

Reinício: calcule $\boldsymbol{r_m} = \boldsymbol{b} - A\boldsymbol{x_m}$. Se for satisfatório, PARE. Senão, então calcule

$\boldsymbol{x_0} = \boldsymbol{x_m}$

$\boldsymbol{v_1} = \boldsymbol{r_m}/||\boldsymbol{r_m}||$

Volte para o passo 2.

2.1.1 Problema de Mínimos Quadrados

Um dos passos cruciais do algoritmo consiste na resolução do seguinte problema de mínimos quadrados:

$$\min_{\boldsymbol{y} \in \mathbb{R}^k} ||\beta\boldsymbol{e_1} - \bar{H}_k\boldsymbol{y}||.$$

Um modo clássico e eficiente de resolver esse tipo de problema é através do processo de fatoração QR usando rotações de Givens no plano (GOLUB; LOAN, 1989), isso devido ao fato de $\bar{H}_k$ ser uma matriz de Hessenberg. Conhecida a estrutura desta matriz $\bar{H}_k$, o que é necessário ser feito é a eliminação dos elementos abaixo da diagonal principal, tornando-a uma matriz triangular.

Seja a matriz $(F_j)_{(k+1)\times(k+1)}$ dada por

$$F_j = \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_j & -s_j & & & \\ & & & s_j & c_j & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix} \leftarrow j\text{-ésima linha}$$

com $c_j^2 + s_j^2 = 1$. Matrizes com o padrão acima são ortogonais e chamadas matrizes de rotação de Givens. Os coeficientes c_j e s_j são selecionados a fim de eliminar os elementos $h_{j+1,j}$ da matriz $\bar{H}_k$ ao multiplicá-la por F_j. Para isso, os coeficientes, na j-ésima rotação F_j, são definidos como:

$$s_j = \frac{h_{j+1,j}}{\sqrt{h_{j,j}^2 + h_{j+1,j}^2}} \quad \text{e} \quad c_j = \frac{h_{j,j}}{\sqrt{h_{j,j}^2 + h_{j+1,j}^2}}. \tag{27}$$

Em cada iteração, isto é, a cada vez que multiplica-se a matriz $\bar{H}_k$ por F_j, o elemento $h_{j+1,j}$ é eliminado. Assim, após k passos do processo, encontra-se uma decomposição para $\bar{H}_k$. Para isso, seja

$$R_k^{(0)} = \bar{H}_k \tag{28}$$

e

$$R_k^{(j)} = F_j R_k^{(j-1)}. \tag{29}$$

Após k iterações, tem-se:

$$\bar{R}_k = R_k^{(k)} = F_k R_k^{(k-1)} = F_k F_{k-1} \cdots F_2 F_1 R_k^{(0)}. \tag{30}$$

Definindo $Q_k = F_k F_{k-1} \cdots F_2 F_1$, pode-se escrever

$$\bar{R}_k = Q_k \bar{H}_k. \tag{31}$$

Como F_j é ortogonal para todo j, Q_k também é e, por construção, $\bar{R}_k$ é triangular superior. Sendo assim, tem-se a fatoração QR para $\bar{H}_k$:

$$\bar{H}_k = Q_k^{-1} \bar{R}_k. \tag{32}$$

Se simultaneamente forem aplicadas as rotações F_j a H_k e modificar o vetor $\beta \boldsymbol{e_1}$, ao fim do processo será obtido o vetor

$$\bar{\boldsymbol{g}}_k = Q_k \beta \boldsymbol{e_1} = (\gamma_1, \cdots, \gamma_{k+1}), \tag{33}$$

com

$$\gamma_j = c_j \gamma_j; \tag{34}$$

$$\gamma_{j+1} = -s_j \gamma_j. \tag{35}$$

Uma vez que Q_k é ortogonal, então

$$\begin{aligned} J(\boldsymbol{y}) &= ||\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{y}|| && (36)\\ &= ||Q_k||\ ||\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{y}\| && (37)\\ &= ||Q_k(\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{y})|| && (38)\\ &= ||\bar{\boldsymbol{g}}_k - \bar{R}_k \boldsymbol{y}||. && (39) \end{aligned}$$

Como a última linha de $\bar{H}_k$ possui um único elemento não nulo ($h_{k+1,k}$), a última linha de $\bar{R}_k$ será completamente nula. Assim, o problema de minimização de $J(\boldsymbol{y})$ é obtido ao resolver um sistema triangular superior que resulta da remoção da última linha de $\bar{R}_k$ e da última linha que compõe $\bar{g}_k$. A Proposição (2.1.1) traz três importantes resultados, cuja demonstração parcial encontra-se discutida anteriormente, e resume toda a discussão precedente.

Proposição 2.1.1. *Dada a decomposição $\bar{R}_k = Q_k \bar{H}_k$ e o vetor $\bar{g}_k$, obtidos anteriormente, seja R_k a matriz triangular superior de dimensão $k \times k$ obtida de $\bar{R}_k$ eliminando sua última linha e $\boldsymbol{g_k}$ o vetor com dimensão k obtido pelos k primeiros elementos de $\bar{\boldsymbol{g}}_k$. Então:*

1. *O posto de AV_k é igual ao posto de R_k. Em particular, se $\boldsymbol{r_{kk}} = 0$ então A é singular.*

2. *O vetor $\boldsymbol{y} \in \mathbb{R}^k$ que minimiza*

$$J(\boldsymbol{y}) = ||\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{y}||$$

é dado por

$$\boldsymbol{y} = R_k^{-1} \boldsymbol{g_k}.$$

3. *O resíduo na k-ésima iteração satisfaz*

$$\boldsymbol{b} - A\boldsymbol{x_k} = V_{k+1}(\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{y_k}) = V_{k+1} Q_k^T (\gamma_{k+1} \boldsymbol{e_{k+1}})$$

e, como resultado

$$||\boldsymbol{b} - A\boldsymbol{x_k}|| = |\gamma_{k+1}|.$$

A demonstração formal da Proposição (2.1.1) pode ser encontrada em (SAAD, 2003).

Encontrada a solução do sistema, o vetor obtido da diferença $\boldsymbol{b} - A\boldsymbol{x_k}$ possui um único componente não nulo, na $(k+1)$-ésima posição, cujo valor é γ_{k+1}. Analisando o algoritmo (4), observa-se que a única possibilidade de parada é quando $h_{j+1,j} = 0$, na j-ésima iteração. O próximo vetor não pode ser calculado, portanto o algoritmo para, encontrando a solução exata. Pode-se também afirmar que, caso o algoritmo pare na j-ésima iteração, com $b - Ax_j = 0$, então $h_{j+1,j} = 0$.

Teorema 2.1.1. *Seja A uma matriz não singular. Então o algoritmo GMRES para no passo j, ou seja, $h_{j+1,j} = 0$, se e somente se a solução aproximada x_j é exata.*

A demonstração desse teorema também é encontrada em (SAAD, 2003)

Para um sistema com dimensão $n \times n$, o método GMRES converge para a solução com, no máximo, n iterações. Definir a quantidade ideal de vetores na base do subespaço de $Krylov$ é uma dificuldade pertinente no método. Caso essa quantidade seja pequena, tem-se uma quantidade menor de operações, porém são necessárias mais iterações. Caso seja uma grande quantidade de vetores, a convergência do método se dá com menos iterações, contudo com custo computacional maior, uma vez que será necessário armazenar mais vetores para a base do espaço. O algoritmo completo do GMRES(m) é apresentado a seguir cujos parâmetros de entrada são: matriz A, solução inicial $\boldsymbol{x_0}$, vetor $\boldsymbol{b}$, quantidade de vezes que o algoritmo pode reinicializar l_{max}, número de vetores da base do subespaço de $Krylov$ k_{max} e tolerância para a solução tol.

Algoritmo 5: GMRES(m) com fatoração QR

Entrada: $A,\ \boldsymbol{x_0},\ \boldsymbol{b},\ k,\ l_{max},\ tol.$
Calcule $\epsilon = tol/||\boldsymbol{b}||$
Defina $l = 1$
$i = 1; \boldsymbol{u_i} = \boldsymbol{b} - A\boldsymbol{x_i}; \boldsymbol{e_i} = ||\boldsymbol{u_i}||; \boldsymbol{u_i} = \frac{\boldsymbol{u_i}}{e_i}; \rho = e_i;$
para $\rho > \epsilon$ *e* **faça**
  $\boldsymbol{u_{i+1}} = A\boldsymbol{u_i}$
  Ortogonalização de Gram-Schmidt:
  para $j = 1, 2, \cdots, i$ **faça**
    $h_{j,i} = \boldsymbol{u_{i+1}^T}\boldsymbol{u_j};\ \boldsymbol{u_{i+1}} = \boldsymbol{u_{i+1}} - h_{j,i}\boldsymbol{u_j};$
  fim
  $h_{i+1,i} = ||\boldsymbol{u_{i+1}}||;\ \boldsymbol{u_{i+1}} = \frac{\boldsymbol{u_{i+1}}}{h_{i+1,i}}$
  Algoritmo QR:
  para $j = 1, 2, \cdots, i-1$ **faça**
    $h_{i,j} = c_j h_{j,i} + s_j h_{j+1,j};\ h_{j+1,i} = -s_j h_{j,i} + c_j h_{j+1,i};$
  fim
  $r = (h_{i,i}^2 + h_{i+1,i}^2)^{\frac{1}{2}};\ c_i = \frac{h_{i,i}}{r};$
  $s_i = \frac{h_{i+1,i}}{r};\ h_{i,i} = r;$
  $h_{i+1,i} = 0;\ e_{i+1} = -s_i e_i;$
  $e_i = c_i e_i;\ \rho = |e_{i+1}|;$
  $i = i + 1;$
fim
$i = i - 1;$
para $j = i, i-1, \cdots, 1$ **faça**
  $y_j = \frac{e_j - \sum_{l=j+1}^{i} h_{j,l} y_l}{h_{j,j}};$
fim
$\boldsymbol{x} = \sum_{j=1}^{i} y_j u_j;$

2.1.2 Análise Computacional

Seja A a matriz do sistema $A\boldsymbol{x} = \boldsymbol{b}$ cuja dimensão é $(n \times n)$ e n_z o número de elementos não nulos pertencentes à matriz A. Para calcular os vetores $\boldsymbol{v_1}, \cdots, \boldsymbol{v_k}$, base do espaço de $Krylov$ de ordem k, através do processo de Arnoldi, na j-ésima iteração, o algoritmo

necessita realizar $(2j+1)n + n_z$ multiplicações, isso para o vetor não normalizado. Na última iteração, o algoritmo executa $(k+1)n$ multiplicações, ou seja, $k \cdot n$ iterações a menos que nos demais casos. Portanto, para a construção dos vetores, o total de multiplicações é aproximadamente

$$k(k+2)n + kn_z - kn = k(k+1)n + kn_z.$$

Para calcular a solução aproximada $\boldsymbol{x_k} = \boldsymbol{x_0} + V_k \boldsymbol{y_k}$ são executadas $k \cdot n$ multiplicações. As k iterações do método GMRES requerem $k(k+2)n + kn_z$ multiplicações, resultando $(k+2)n + n_z$ iterações em média (SAAD; SCHULTZ, 1986). A complexidade para calcular os vetores $\boldsymbol{y_k}$, solução do problema de mínimos quadrados, é $O(nm^3)$.

2.2 CMRH

Assim como o GMRES, o CMRH (SADOK, 1999), (HEYOUNI; SADOK, 2008) é um método de resolução de sistemas lineares que usa a ideia de projeção da solução sobre o espaço de $Krylov$. O que os difere é a forma como são construídas as bases de $\mathcal{K}_k(A,\ \boldsymbol{r_0})$. No GMRES, a base é construída usando-se o processo de Arnoldi enquanto no CMRH usa-se o processo de Hessenberg.

2.2.1 Método de Hessenberg

Dados $\boldsymbol{r_0} \in \mathbb{R}^n$ e $A \in \mathbb{R}^{n \times n}$, considere o espaço de $Krylov$

$$\mathcal{K}_k = \mathcal{K}_k(A, \boldsymbol{r_0}) = span\{\boldsymbol{r_0}, A\boldsymbol{r_0}, \cdots, A^{k-1}\boldsymbol{r_0}\}$$

para algum inteiro $1 \leq k \leq n$, exatamente como no método GMRES.

Seja $\boldsymbol{v_j} = A^{j-1}\boldsymbol{r_0}, \forall j = 1, \cdots k$ e $(V_j)_{n \times j}$ a matriz cujas colunas são os vetores $\boldsymbol{v_1}, \cdots, \boldsymbol{v_j}$. A matriz V_j é chamada matriz de $Krylov$ associada a $\mathcal{K}_k$. O objetivo é construir uma base $\{\boldsymbol{l_1}, \cdots, \boldsymbol{l_k}\}$ para $\mathcal{K}_k$, distinta da apresentada no GMRES. Para isso, será usada a fatoração LU de V_j.

Dada uma matriz com dimensão $n \times k$, pode-se particioná-la como

$$V_j = \begin{pmatrix} V1_j \\ V2_j \end{pmatrix}, \forall j = 1, \cdots, k,$$

sendo $V1_j$ uma matriz quadrada com dimensão $j \times j, \forall j = 1, \cdots k$ e $V2_j$ uma matriz com dimensão $(n-j) \times j, \forall j = 1, \cdots k$. Se $V1_j$ é não singular, define-se $V_j^{\dagger}$ como a pseudo-inversa de V_j por

$$V_j^{\dagger} = (I_j^{(n)^T} V_j)^{-1} I_j^{(n)^T} = (V1_j^{-1}, 0),$$

sendo $(V1_j^{-1}, 0)$ a matriz formada pela inversa de $V1_j$ acrescida de $n-k$ colunas de zeros e I_j^n é a matriz formada pela matriz de identidade de ordem j acrescida de $n-j$ linhas nulas.

Sendo V_j uma matriz retangular com dimensão $n \times k$, caso a fatoração LU exista, pode-se escrever $V_j = L_j U_j$ sendo L_j uma matriz trapezoidal com dimensão $n \times k$ e U_j uma matriz triangular superior quadrada de ordem k.

Uma condição suficiente para que V_j tenha fatoração LU ((GOLUB; LOAN, 1989)) é:

$$\det(V1_j) \neq, 0 \ \forall j = 1, \cdots, k. \tag{40}$$

Obviamente, se V_j satisfaz a condição acima, para qualquer $j = 1, \cdots, k$, a matriz V_j possui fatoração LU. Além disso, se

$$V_j = L_j U_j \text{ e } V_{j+1} = L_{j+1} U_{j+1},$$

sendo $(L_j)_{n \times j}$ e $(U_j)_{j \times j}$ fatorações LU de V_j e V_{j+1}, respectivamente, tem-se que

$$L_{j+1} = [L_j \ l_{j+1}]$$

e pode-se assumir que U_j tem diagonal unitária para todo j.

Teorema 2.2.1. *Seja $\boldsymbol{r_0} \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}$ e $\boldsymbol{v_j} = A^{j-1}\boldsymbol{r_0}$, para $j = 1, \cdots, k$. Suponha que para cada j, $\det(V1_j) \neq 0$. Então:*

1. *$\mathcal{K}_k = span\{\boldsymbol{l_1}, \cdots, \boldsymbol{l_j}\}$, sendo os vetores $\boldsymbol{l_j}$ as colunas da matriz L_k da fatoração LU de V_k.*

2. *Para cada $j = 1, \cdots, k-1$, tem-se*

$$AL_j = L_{j+1}\bar{H}_j$$

sendo L_j, L_{j+1} as matrizes da fatoração LU de V_j e V_{j+1} e

$$\bar{H}_j = \begin{bmatrix} (H_j)_{j \times j} \\ \boldsymbol{e_j}^T \end{bmatrix}$$

no qual $(H_j)_{j \times j}$ é uma matriz de Hessenberg e $\boldsymbol{e_j} = (0, \cdots, 0, 1)^T \in \mathbb{R}^j$.

3. *$\boldsymbol{l_{j+1}} = A\boldsymbol{l_j} - L_j L_j^{\dagger} A\boldsymbol{l_j}, \ \forall j = 1, \cdots, k-1$ com $\boldsymbol{l_1} = \boldsymbol{v_1} = \boldsymbol{r_0}$.*

Demonstração

1. *Considere a fatoração LU da matriz $V_j = L_j U_j$. Dessa maneira, os vetores $\boldsymbol{l_1}, \cdots, \boldsymbol{l_j}$ são linearmente independentes devido ao fato de que, pela hipótese sobre $V1_j$, a matriz U_j é não singular. Assim, os vetores $\boldsymbol{l_1}, \cdots, \boldsymbol{l_j}$ são uma base para o espaço $\mathcal{K}_k$ e consequentemente $span\{\boldsymbol{l_1}, \cdots, \boldsymbol{l_j}\} = \mathcal{K}_k(A, \boldsymbol{r_0})$.*

2.

$$V_{j+1} = [\boldsymbol{v_1}\ \boldsymbol{v_2}\ \cdots\ \boldsymbol{v_{j+1}}] = [\boldsymbol{r_0}\ A\boldsymbol{r_0}\ \cdots\ A^{j-1}\boldsymbol{r_0}\ A^j\boldsymbol{r_0}].$$

Assim, as j últimas colunas de V_{j+1} podem ser escritas como $A\boldsymbol{r_0},\ A(A\boldsymbol{r_0}), \cdots,\ A(A^{j-1}\boldsymbol{r_0})$. Dessa forma,

$$\begin{aligned}[A\boldsymbol{r_0}\ A^2\boldsymbol{r_0}\ \cdots A^j\boldsymbol{r_0}] &= A[\boldsymbol{r_0}\ A\boldsymbol{r_0}\ \cdots\ A^{j-1}A\boldsymbol{r_0}] \\ &= AV_j.\end{aligned}$$

Logo $V_{j+1} = [\boldsymbol{r_0}\ AV_j]$. Seja I_j a matriz identidade com dimensão $j \times j$. Considere a matriz

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{(j+1)\times j} = \begin{bmatrix} \mathbf{0_{1\times j}} \\ I_j \end{bmatrix}.$$

Observe que

$$V_{j+1}\begin{bmatrix} \mathbf{0_{1\times j}} \\ I_j \end{bmatrix} = [\boldsymbol{r_0}\ AV_j]\begin{bmatrix} \mathbf{0_{1\times j}} \\ I_j \end{bmatrix} = AV_j. \tag{41}$$

Pela hipótese $\det(V1_j) \neq 0$, pode-se tomar a decomposição LU de V_j e V_{j+1}:

$$V_j = L_j U_j \text{ e } V_{j+1} = L_{j+1}U_{j+1}, \text{ sendo } L_{j+1} = [L_j\ \boldsymbol{l_{j+1}}].$$

Sendo assim:

$$L_{j+1}U_{j+1}\begin{bmatrix} \mathbf{0_{1\times j}} \\ I_j \end{bmatrix} = AL_jU_j \quad \Longrightarrow \quad AL_j = L_{j+1}U_{j+1}\begin{bmatrix} \mathbf{0_{1\times j}} \\ I_j \end{bmatrix}U_j^{-1}. \tag{42}$$

Como U_j é triangular superior unitária, devido à fatoração LU, U_j^{-1} também o é. Então, tem-se o seguinte produto:

$$\begin{bmatrix} \mathbf{0_{1\times j}} \\ I_j \end{bmatrix}U_j^{-1} = \begin{bmatrix} \mathbf{0_{1\times j}} \\ U_j^{-1} \end{bmatrix}_{(j+1)\times j} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & u_{12} & \cdots & u_{1j} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{(j-1)\times j} \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{(j+1)\times j}. \tag{43}$$

Seja a matriz

$$\bar{H}_j = U_{j+1} \begin{bmatrix} \mathbf{0_{1\times j}} \\ I_j \end{bmatrix} U_j^{-1}$$

com dimensão $(j+1)\times j$*. Observe que se* $i \geq m+2$*, então* $h_{im} = 0$*. De fato, a linha* i *de* U_{j+1} *possui as* $i-1$ *primeiras entradas nulas, enquanto a coluna* m *de* $\begin{bmatrix} \mathbf{0_{1\times j}} \\ U_j^{-1} \end{bmatrix}$ *possui as* $m+1$ *primeiras entradas não todas nulas e as demais nulas. Como* $i \geq m+2 \Rightarrow i-1 \geq m+1$*. Sendo assim, o produto da linha pela coluna se anula. Além disso, como as diagonais de* U_{j+1} *e* U_{j+1}^{-1} *são unitárias, os elementos* $h_{i+1,i} = 1$*. Portanto, a matriz* $\bar{H}_j$ *tem a forma*

$$\bar{H}_j = \begin{bmatrix} h_{1,1} & & \cdots & & h_{1,j} \\ 1 & h_{2,2} & \cdots & & h_{2,j} \\ 0 & 1 & h_{3,3} & \cdots & h_{3,j} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & h_{jj} \end{bmatrix}.$$

Definindo $(H_j)_{j\times j}$ *como a matriz obtida de* $\bar{H}_j$ *pela exclusão de sua última linha, tem-se:*

$$\bar{H}_j = \begin{bmatrix} H_j \\ \boldsymbol{e_j^T} \end{bmatrix} \Longrightarrow AL_j = L_{j+1}\bar{H}_j,\ \forall j = 1, \cdots, k \tag{44}$$

o que prova o item 2.

3. Observe que

$$AL_j = L_{j+1}\bar{H}_j = [L_j\ \boldsymbol{l_{j+1}}] \begin{bmatrix} H_j \\ \boldsymbol{e_j^T} \end{bmatrix} = L_jH_j + \boldsymbol{l_{j+1}e_j^T}. \tag{45}$$

Se multiplicar a matriz identidade pela pseudo-inversa à esquerda de L_j*, tem-se*

$$L_j^\dagger AL_j = L_j^\dagger[L_j\ \boldsymbol{l_{j+1}}] \begin{bmatrix} H_j \\ \boldsymbol{e_j^T} \end{bmatrix} = [L_j^\dagger L_j\ L_j^\dagger \boldsymbol{l_{j+1}}] \begin{bmatrix} H_j \\ \boldsymbol{e_j^T} \end{bmatrix}. \tag{46}$$

Como L_j *é triangular inferior, então* $L_j^\dagger$ *tem o formato*

$$L_j^\dagger = \begin{bmatrix} z_{1,1} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots & & \vdots \\ z_{j,1} & \cdots & z_{j,j} & 0 & 0 & \cdots & 0 \end{bmatrix}_{j\times n}.$$

Além disto, $\boldsymbol{l_{j+1}^T} = (0\ \cdots\ 0\ \boldsymbol{l_k}\ \cdots\ \boldsymbol{l_j l_{j+1}})$*. Logo* $L_j^\dagger \boldsymbol{l_{j+1}} = \mathbf{0_{j\times 1}}$*. Devido à fatoração LU, tem-se*

$$L_j^\dagger AL_j = [I_j\ \mathbf{0_{j\times 1}}] \begin{bmatrix} H_j \\ \boldsymbol{e_j^T} \end{bmatrix} = H_j \tag{47}$$

Novamente, da equação (45) tem-se

$$AL_j = L_j L_j^{\dagger} AL_j + \boldsymbol{l_{j+1}} \boldsymbol{e_j^T}. \tag{48}$$

Portanto

$$\begin{aligned} \boldsymbol{l_{j+1}} &= AL_j \boldsymbol{e_j} - L_j L_j^{\dagger} AL_j \boldsymbol{e_j} && (49) \\ &= A\boldsymbol{l_j} - L_j L_j^{\dagger} A\boldsymbol{l_j}. && (50) \end{aligned}$$

Observe ainda que a equação (47) mostra que a linha i da matriz H_j pode ser calculada por

$$L_j^{\dagger} A\boldsymbol{l_i}.$$

O processo de Hessenberg, que constroi os vetores $\boldsymbol{l_k}$, é apresentado a seguir, com base no Teorema (2.2.1).

Algoritmo 6: Processo de Hessenberg

Entrada: Matriz A, vetor $\boldsymbol{r_0}$, escalar n.

$\boldsymbol{l_1} = \boldsymbol{v}$
para $k = 1, 2, \cdots, n$ **faça**
  $\boldsymbol{u} = A\boldsymbol{l_k}$
  para $j = 1, 2, \cdots, k$ **faça**
    $\beta_j = u_j / \boldsymbol{l}_{\boldsymbol{j}\,j}$
    $h_{j,k} = \beta_j$
    $u_j = 0$
    para $l = j + 1, \cdots, n$ **faça**
      $u_l = u_l - \beta_j \boldsymbol{l}_{\boldsymbol{j}\,j}$
    fim
  fim
  $h_{k+1,k} = 1$
  $\boldsymbol{l_{k+1}} = \boldsymbol{u}$
fim

No algoritmo (6) pode ocorrer uma falha caso $\boldsymbol{l}_{\boldsymbol{j}\,j} = 0$. Para contornar esse problema, usa-se uma estratégia de pivoteamento, análoga à aplicada na Eliminação Gaussiana, equivalente a substituir a matriz inversa de L_k por

$$L_k^{\dagger} = (Y_k^T L_k)^{-1} Y_k^T \tag{51}$$

sendo $Y_k = [\boldsymbol{y_1} \ \cdots \ \boldsymbol{y_k}]$ a matriz das permutações necessárias para a estratégia de pivoteamento, com colunas $\boldsymbol{y_k} = \boldsymbol{e}_{p_k}^n$ e $p_k \in \{1, \cdots, n\}$. Será mostrado agora como determinar p_k.

Suponha que $p_1, \cdots, p_k$ já tenham sido obtidos e deseja-se determinar p_{k+1}. Calcula-se, inicialmente, $\boldsymbol{u} = A\boldsymbol{l_k}$ e subtrai-se os múltiplos de $\boldsymbol{l_1}, \cdots, \boldsymbol{l_k}$ com o objetivo de eliminar os k

componentes $p_1, \cdots, p_k$ de $\boldsymbol{u}$ para obter o vetor $\boldsymbol{w}$. Escolhe-se i_0 tal que $||\boldsymbol{w}||_\infty = |\boldsymbol{w_{i_0}}|$. Define-se $p_{k+1} = i_0$ e o vetor $\boldsymbol{l_{k+1}}$ é dado por

$$\boldsymbol{l_{k+1}} = \frac{\boldsymbol{w}}{\boldsymbol{w}_{i_0}}. \tag{52}$$

Desta forma, o vetor $\boldsymbol{l_{k+1}}$ é tal que $||\boldsymbol{l_{k+1}}||_\infty = 1$, contendo k componentes nulas. O algoritmo (6) produz uma matriz de Hessenberg superior unitária H_k e uma matriz trapezoidal L_k não unitária. Usando o pivoteamento, o algoritmo a seguir produz uma matriz de Hessenberg superior não unitária e uma matriz trapezoidal inferior PL_k unitária (SADOK, 1999).

Algoritmo 7: Processo de Hessenberg com Pivoteamento

Entrada: Matriz A, vetor $\boldsymbol{r_0}$, escalar n.
Seja $\boldsymbol{p} = (0, 1, 2, \cdots, n)^T$
Seja i_0 tal que $|\boldsymbol{r_{0_{i_0}}}| = ||\boldsymbol{r_0}||_\infty$
$\boldsymbol{l_1} = \boldsymbol{r_0}/\boldsymbol{r_{0_{i_0}}}$
$\boldsymbol{p_1} 1 \leftrightarrow \boldsymbol{p_{i_0}}$
para $k = 1, 2, \cdots, n$ **faça**
  $\boldsymbol{u} = A\boldsymbol{l_k}$
  para $j = 1, 2, \cdots, k$ **faça**
    $h_{j,k} = u_{p_j}$
    $\boldsymbol{u}_{p_j} = 0$
    para $l = j + 1, \cdots, n$ **faça**
      $\boldsymbol{u_{p_l}} = \boldsymbol{u}_{p_l} - c\ \boldsymbol{l}_{j_{p_l}}$
    fim
  fim
  se $k < n$ **então**
    Determine i_0 tal que $|\boldsymbol{u}_{i0}| = ||\boldsymbol{u}||_\infty$
    $p_{k+1} \leftrightarrow p_{i_0}$
    $h_{k+1,k} = \boldsymbol{u_{i_0}}$
    $l_{k+1} = \boldsymbol{u}/u_{i_0}$
  fim
fim

O símbolo $\leftrightarrow$ significa troca de valores das variáveis.

2.2.2 Descrição do Método CMRH

Seja a matriz $A \in \mathbb{R}^{n \times n}$ e o vetor $\boldsymbol{b} \in \mathbb{R}^n$, o método CMRH busca uma sequência de aproximações $\boldsymbol{x_1}, \cdots, \boldsymbol{x_k}$ para a solução do sistema linear $A\boldsymbol{x} = \boldsymbol{b}$, a partir de uma aproximação inicial $\boldsymbol{x_0}$, através da minimização da norma residual $||\boldsymbol{b} - A\boldsymbol{x_k}||$ na k-ésima iteração:

$$\min_{\boldsymbol{x_k}} ||\boldsymbol{b} - A\boldsymbol{x_k}||. \tag{53}$$

O vetor aproximação $\boldsymbol{x_k}$ é dado por

$$\boldsymbol{x_k} = \boldsymbol{x_0} + \boldsymbol{z_k},\ \ \boldsymbol{z_k} \in \mathcal{K}_k = span\{\boldsymbol{r_0},\ A\boldsymbol{r_0}, \cdots, A^{k-1}\boldsymbol{r_0}\} \tag{54}$$

Se $\boldsymbol{z} \in \mathcal{K}_k$, então $A\boldsymbol{z} \in \mathcal{K}_{k+1}$, bem como $\boldsymbol{r_0} \in \mathcal{K}_k \supset \mathcal{K}_{k+1}$. Sendo assim

$$A\boldsymbol{z} - \boldsymbol{r_0} \in \mathcal{K}_{k+1}. \tag{55}$$

Isso implica que existe $\boldsymbol{u} \in \mathbb{R}^{k+1}$ tal que

$$A\boldsymbol{z} - \boldsymbol{r_0} = L_{k+1}\boldsymbol{u} \quad \Longrightarrow \quad \boldsymbol{u} = L_{k+1}^{\dagger}(A\boldsymbol{z_k} - \boldsymbol{r_0}). \tag{56}$$

A matriz $L_{k+1}^{\dagger}$ é uma pseudo-inversa de L_{k+1} e por essa razão o problema de minimização

$$\min_{\boldsymbol{z} \in \mathcal{K}_k} ||A\boldsymbol{z} - \boldsymbol{r_0}||$$

é semelhante ao problema de minimização

$$\boldsymbol{z_k} = \min_{\boldsymbol{z} \in \mathcal{K}_k} ||L_{k+1}^{\dagger}(A\boldsymbol{z} - \boldsymbol{r_0})||. \tag{57}$$

Por outro lado, se $\boldsymbol{z} \in \mathcal{K}_k$, então existe $\boldsymbol{y} \in \mathbb{R}^k$ tal que $\boldsymbol{z} = L_k\boldsymbol{y}$. Desta forma, tem-se

$$\begin{aligned}
\min_{\boldsymbol{z} \in \mathcal{K}_k} L_{k+1}^{\dagger}(A\boldsymbol{z} - \boldsymbol{r_0}) &= \min_{\boldsymbol{y} \in \mathbb{R}^k} ||L_{k+1}^{\dagger}(AL_k\boldsymbol{y} - \boldsymbol{r_0})|| && (58)\\
&= \min_{\boldsymbol{y} \in \mathbb{R}^k} ||L_{k+1}^{\dagger}(L_{k+1}H_k\boldsymbol{y} - \beta L_{k+1}\boldsymbol{e_1}^{(k+1)})|| && (59)\\
&= \min_{\boldsymbol{y} \in \mathbb{R}^k} ||\beta\boldsymbol{e_1}^{(k+1)} - H_k\boldsymbol{y}|| && (60)
\end{aligned}$$

sendo $\beta = ||\boldsymbol{r_0}||$ e $\boldsymbol{r_0} = \beta L_{k+1}\boldsymbol{e_1^{k+1}}$ pois $L_1 = \dfrac{\boldsymbol{r_0}}{||\boldsymbol{r_0}||_\infty}$. O algoritmo CMRH é apresentado a seguir.

Algoritmo 8: CMRH

Entrada: Matriz A, vetor $\boldsymbol{b}$, $\boldsymbol{x_0}$, ε.
$\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$
$\boldsymbol{p} = [0, 1, 2, \cdots, n]$
Determine i_0, tal que $|\boldsymbol{r_{0}}_{i_0}| = ||\boldsymbol{r_0}||_\infty$
$\boldsymbol{l_1} = \boldsymbol{r_0}/\boldsymbol{r_{0}}_{i_0}$
$p_1 \leftrightarrow p_{i_0}$
para $k = 1, 2, \cdots$ *condição* **faça**
  $\boldsymbol{u} = A\boldsymbol{l_k}$
  para $j = 1, \cdots, k$ **faça**
    $h_{j,k} = \boldsymbol{u}_{p_j}$
    $\boldsymbol{u}_{p_j} = 0$
    para $l = j + 1, \cdots, n$ **faça**
      $\boldsymbol{u}_{p_l} = \boldsymbol{u}_{p_l} - cl_{j_{p_l}}$
    fim
  fim
  se $k < n$ **então**
    Determine $i_0 \in \{k + 1 \cdots n\}$ tal que $|\boldsymbol{u}_{i0}| = ||\boldsymbol{u}_{p_{k+1}:p_n}||_\infty$
    $p_{k+1} \leftrightarrow p_{i_0}$
    $h_{k+1,k} = u_{i_0}$
    $\boldsymbol{l_{k+1}} = \boldsymbol{u}/u_{i_0}$
  fim
  se $k = n$ *ou* $||\boldsymbol{b} - A\boldsymbol{x_k}|| < \varepsilon$ **então**
    $\boldsymbol{x_k} = \boldsymbol{x_0} + L_k\boldsymbol{y_k}$, sendo que $\boldsymbol{y_k}$ minimiza $||H_k\boldsymbol{y_k} - \boldsymbol{r_{0}}_{i_0}\boldsymbol{e_1}^{(k+1)}||, \boldsymbol{y_k} \in \mathbb{R}^k$
    Pare a iteração!
  fim
fim

A resolução pelo método CMRH também recai em um problema de minimização. Para isto, será utilizado o mesmo procedimento adotado no método GMRES: Fatoração QR via rotação de Givens. A aplicação desse procedimento irá transformar a matriz H_k em uma matriz triangular e o sistema pode ser resolvido utilizando retrosubstituição. Assim como o GMRES, o método CMRH pode apresentar alto custo computacional caso a quantidade de vetores da base do subespaço de $Krylov$ seja grande, devido ao espaço de memória utilizado. Nesse caso, novamente utiliza-se uma versão reinicializada -CMRH(m)- sendo a aproximação inicial o último vetor calculado. Essa versão é apresentada no algoritmo a seguir.

Algoritmo 9: CMRH (m)

Entrada: Matriz A, vetor $\boldsymbol{b}$, $\boldsymbol{x_0}$, ε, m.

$\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$

$\boldsymbol{p} = [\boldsymbol{0, 1, 2, \cdots, n}]$

Determine i_0, tal que $|\boldsymbol{r_{0i_0}}| = ||\boldsymbol{r_0}||_\infty$

$\boldsymbol{l_1} = \boldsymbol{r_0}/||\boldsymbol{r_{0i_0}}||$

$p_1 \leftrightarrow p_{i_0}$

para $k = 1, 2, \cdots, m$ **faça**

  $\boldsymbol{u} = A\boldsymbol{l_k}$

  para $j = 1, \cdots, k$ **faça**

    $h_{j,k} = \boldsymbol{u_{p_j}}$

    $\boldsymbol{u}_{p_j} = 0$

    para $l = j + 1, \cdots, n$ **faça**

      $\boldsymbol{u}_{p_l} = \boldsymbol{u}_{p_l} - c\boldsymbol{l}_{jp_l}$

    fim

  fim

  se $k < n$ **então**

    Determine $i_0 \in \{k + 1 \cdots n\}$ tal que $|\boldsymbol{u}_{i0}| = ||\boldsymbol{u}_{p_{k+1}:p_n}||_\infty$

    $\boldsymbol{p_{k+1}} \leftrightarrow \boldsymbol{p_{i_0}}$

    $h_{k+1,k} = \boldsymbol{u_{i_0}}$

    $\boldsymbol{l}_{k+1} = \boldsymbol{u}/||\boldsymbol{u_{i_0}}||$

  fim

  se $k = n$ *ou* $||\boldsymbol{b} - A\boldsymbol{x_k} < \varepsilon||$ **então**

    $\boldsymbol{x_k} = \boldsymbol{x_0} + L_k\boldsymbol{y_k}$, sendo que $\boldsymbol{y_k}$ minimiza $||H_k\boldsymbol{y_k} - \boldsymbol{r_{0i_0}}\boldsymbol{e_1}^{(k+1)}||, \boldsymbol{y_k} \in \mathbb{R}^k$

    Pare a iteração!

  fim

fim

$\boldsymbol{x_m} = \boldsymbol{x_0} + L_m\boldsymbol{y_m}$, sendo que $\boldsymbol{y_m}$ minimiza $||H_m\boldsymbol{y} - (\boldsymbol{r_0})_{i_0}\boldsymbol{e_1}^{(m+1)}||, \boldsymbol{y} \in \mathbb{R}^k$

$\boldsymbol{x_0} = \boldsymbol{x_m}$

Volte ao início.

2.2.3 Análise Computacional

Seja A a matriz do sistema $A\boldsymbol{x} = \boldsymbol{b}$ cuja dimensão é $n \times n$ e seja n_z o número de elementos não nulos desta matriz. O processo de Hessenberg, na k-ésima iteração, requer $\sum_{j=1}^{k}(n - j) = nk - k(k + 1)/2$ multiplicações além de um produto matriz-vetor com $n(n - k)$ multiplicações, sendo $n(n - k) < n_z$. Para m iterações, o algoritmo requer cerca de $nm^2/2 - m^3/6 + \sum_{k=1}^{m} n(n - k)$ multiplicações. Portanto, o processo de Hessenberg tem custo computacional menor que o processo de Arnoldi.

Para calcular o vetor $\boldsymbol{u} = AL_k$ são necessárias $n(n - k)$ multiplicações para uma matriz densa e $n(n - k) < n_z$ para uma matriz esparsa. Sem considerar o cálculo de $\boldsymbol{y_k}$, o método CMRH necessita, para a k-ésima iteração, $kn + n(n - k)$ com $n(n - k) < n_z$. Por isso, o método CMRH tem custo computacional menor que o método GMRES (SADOK, 1999). Assim como no GMRES, a complexidade para calcular os vetores $\boldsymbol{y_k}$, solução do problema

de mínimos quadrados, é $O(nm^3)$.

2.2.4 CMRH com Sobre-armazenamento

Nesta subseção, será feito um resumo sobre o método CMRH com sobre-armazenamento (CMRH-OVER), outra versão do método CMRH proposto por (HEYOUNI; SADOK, 2008). Este método apresenta uma implementação sem a estratégia de reinicialização. A principal diferença entre os métodos CMRH e CMRH-OVER é a forma de armazenar as matrizes A, L e H. Diferentemente do método CMRH, o método CMRH-OVER não armazena três matrizes, mas apenas uma. Isso diminui expressivamente o uso de memória. Neste método, as matrizes L_k e H_k são escritas sobre a matriz A do sistema $A\boldsymbol{x} = \boldsymbol{b}$. Resumidamente, o método pode ser divido em duas partes: construção da base para o espaço de $Krylov$ através do processo de Hessenberg e a resolução do sistema por fatoração QR.

Após k iterações do algoritmo que constrói a base para o espaço de $Krylov$, através do processo de Hessenberg, obtém-se uma matriz L_k trapezoidal inferior unitária. Ao calcular o produto $A\boldsymbol{l_k}$, as $k-1$ primeiras colunas da matriz A não são utilizadas. Assim, para minimizar o uso de memória, a parte superior de H_k e a parte inferior de L_k podem ser escritas sobre a matriz A . Um vetor $\boldsymbol{h}$ com dimensão k armazena a subdiagonal de H_k. O algoritmo abaixo descreve o processo de Hessenberg com sobre-armazenamento.

Algoritmo 10: Processo de Hessenberg com sobre-armazenamento.

Entrada: Matriz A, vetor $\boldsymbol{b}$, $\boldsymbol{x_0}$, ϵ, m.
$\boldsymbol{p} = [1, \cdots, n]^T$
Determine i_0 tal que $|\boldsymbol{v}_{i0}| = ||\boldsymbol{v}||_\infty$
$\beta = \boldsymbol{v}_{i0}$; $\boldsymbol{v} = \boldsymbol{v}/\beta$
$\boldsymbol{p_{i0}} \longleftrightarrow \boldsymbol{p_1}$; $\boldsymbol{v_{i0}} \longleftrightarrow \boldsymbol{v_1}$
$A_{i0,:} \longleftrightarrow A_{1,:}$; $A_{:,i0} \longleftrightarrow A_{:,1}$
para $k = 1$ *até* m **faça**
  $\boldsymbol{u} = A_{:,k} + A_{:,k+1:n}\boldsymbol{v_{k+1:n}}$
  $A_{k,k+1:n} = \boldsymbol{v_{k+1:n}}$
  para $j = 1$ *até* k **faça**
    $A_{j,k} = \boldsymbol{u_j}$; $\boldsymbol{u_j} = 0$
    $\boldsymbol{u_{j+1:n}} = \boldsymbol{u_{j+1:n}} - A_{j,k}A_{j+1:n,j}$
  fim
  Determine i_0 tal que $|\boldsymbol{v}_{i0}| = ||\boldsymbol{v}||_\infty$
  $h_{k+1} = \boldsymbol{u_{i0}}$; $\boldsymbol{v} = \boldsymbol{u}/h_{k+1}$
  $\boldsymbol{p_{i0}} \longleftrightarrow \boldsymbol{p_{k+1}}$; $\boldsymbol{v_{i0}} \longleftrightarrow \boldsymbol{v_{k+1}}$
  $A_{i0,:} \longleftrightarrow A_{k+1,:}$; $A_{:,i0} \longleftrightarrow A_{:,k+1}$
fim

Assim como o CMRH, o CMRH-OVER é um método que usa projeção sobre o espaço de $Krylov$ $\mathcal{K}_k(A, \boldsymbol{r_0})$, baseado no processo de Hessenberg e consiste em criar uma sequência

de aproximações $\boldsymbol{x_1}, \cdots, \boldsymbol{x_k}$ da forma

$$\boldsymbol{x_k} = \boldsymbol{x_0} + \boldsymbol{z_k},\ \boldsymbol{z_k} \in \mathcal{K}_k(A\boldsymbol{r_0})$$

para o sistema $A\boldsymbol{x} = \boldsymbol{b}$, sendo $\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$ o resíduo inicial.

A matriz L_k com dimensão $(n \times k)$ é calculada pelo processo de Hessenberg e suas colunas formam uma base para o espaço de $Krylov$. Dessa forma, $\boldsymbol{z_k}$ pode ser escrito como uma combinação linear das colunas de L_k. Sendo assim, tem-se

$$\boldsymbol{z_k} = L_k\boldsymbol{d_k}$$

com $\boldsymbol{d_k} \in \mathbb{R}^k$. Assim como no CMRH, o objetivo do método é minimizar o resíduo e a aproximação $\boldsymbol{x_k}$ é obtida como

$$\boldsymbol{x_k} = \boldsymbol{x_0} + L_k\boldsymbol{d_k}$$

e $\boldsymbol{d_k}$ é calculado através do problema de mínimos quadrados

$$\min_{\boldsymbol{d_k} \in \mathbb{R}^{k+1}} ||\beta\boldsymbol{e_1} - H_k\boldsymbol{d_k}||.$$

Para resolver o problema de mínimos quadrados, o método transforma a matriz de Hessenberg H_k em uma matriz triangular superior por fatoração QR através do processo de Rotações de Givens, essencialmente, os mesmos procedimentos usados no método CMRH. O algoritmo completo do método CMRH-OVER é apresentado a seguir.

Algoritmo 11: CMRH com sobre-armazenamento.

Entrada: Matriz A, vetor $\boldsymbol{b}$, solução inicial $\boldsymbol{x_0}$ e ε.

$\boldsymbol{p} = [1, \cdots, n]^T$

$\boldsymbol{b} = \boldsymbol{b} - A\boldsymbol{x_0}$

Determine i_0 tal que $|\boldsymbol{b}_{i0}| = ||\boldsymbol{b}||_\infty$

$\beta = \boldsymbol{b}_{i0}$; $\boldsymbol{b} = \boldsymbol{b}/\beta$

$\boldsymbol{p_{i0}} \longleftrightarrow \boldsymbol{p_1}$; $\boldsymbol{b_{i0}} \longleftrightarrow \boldsymbol{b_1}$

$A_{i0,:} \longleftrightarrow A_{1,:}$; $A_{:,i0} \longleftrightarrow A_{:,1}$

para $k = 1$ *até a convergência do método* **faça**

- $\boldsymbol{u} = A_{:,k} + A_{:,k+1:n}\boldsymbol{b_{k+1:n}}$
- $A_{k,k+1:n} = \boldsymbol{b_{k+1:n}}$
- **para** $j = 1$ *até* k **faça**
  - $A_{j,k} = \boldsymbol{u_j}$; $\boldsymbol{u_j} = 0$
  - $\boldsymbol{u_{j+1:n}} = \boldsymbol{u_{j+1:n}} - A_{j,k}A_{j+1:n,j}$
- **fim**
- Determine i_0, tal que $|\boldsymbol{u}_{i_0}| = ||\boldsymbol{u}||_\infty$
- $h = \boldsymbol{u_{p_{i0}}}$; $\boldsymbol{v} = \boldsymbol{u}/h$
- $\boldsymbol{p_{i0}} \longleftrightarrow \boldsymbol{p_{k+1}}$; $\boldsymbol{v_{i0}} \longleftrightarrow \boldsymbol{v_{k+1}}$
- $A_{i0,:} \longleftrightarrow A_{k+1,:}$; $A_{:,i0} \longleftrightarrow A_{:,k+1}$
- Atualização da Fatoração QR de H_k
- Aplique as rotações de Givens sobre a k- ésima coluna de H_k, ou seja aplique $F_i, i = 1, \cdots, k-1$ sobre $[A_{1:k,k}, h]$
- Calcule os coeficientes c_k e s_k
- $rs_{k+1} = -s_k rs_k$; $rs_k = c_k rs_k$
- $A_{k,k} = c_k A_{k,k} + s_k h$
- Se $|rs_{k+1} < \varepsilon|$, siga passo 2.
- Passo 2:
- Resolva $H_k\boldsymbol{d_k} = \beta\boldsymbol{e_1}$
- $\boldsymbol{x_k} = \boldsymbol{x_0} + L_k\boldsymbol{d_k}$
- Reordenação dos componentes de $\boldsymbol{x_k}$
- **para** $i = 1, \cdots, n$ **faça**
  - $\boldsymbol{b}_{\boldsymbol{p_i}} = x_{k_i}$
- **fim**

fim

Além de ter um custo de armazenamento menor que o CMRH, uma vez que os vetores que formam a base do espaço de $Krylov$ são armazenados na própria matriz A, este método executa, na k-ésima iteração, um produto matriz-vetor, como no CMRH, porém não da matriz A inteira, mas sim das $n-k$ colunas de A. Portanto, à medida que k aumenta, o método executa menos multiplicações. Isso o torna mais eficiente que o CMRH. Uma consequência da forma como esse método armazena os vetores é que, ao fim do processo iterativo, a matriz A está totalmente modificada.

2.3 LCD

O método LCD foi desenvolvido para resolver sistemas lineares do tipo $A\boldsymbol{x} = \boldsymbol{b}$, sendo A uma matriz não singular e não simétrica (DAI; YUAN, 2004). O método consiste basicamente em encontrar um conjunto de vetores de direções conjugadas à esquerda de A, que formam uma base do espaço vetorial, no qual a solução exata $\boldsymbol{x}^* = A^{-1}\boldsymbol{b}$ pode ser obtida através de uma combinação linear dos vetores dessa base.

Definição 2.3.1. *Os vetores $\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}$ são chamados de direções conjugadas à esquerda de uma matriz real A não singular se*

$$\begin{cases} \boldsymbol{p_i^T} A \boldsymbol{p_j} = 0 \ \forall \ i < j; \\ \boldsymbol{p_i^T} A \boldsymbol{p_i} \neq 0 \ \forall \ i. \end{cases} \tag{61}$$

Seja $P = [\boldsymbol{p_1} \ \boldsymbol{p_2} \ \cdots \ \boldsymbol{p_n}]$ a matriz cujas colunas são os vetores $\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}$, então a matriz L, definida como $L = P^T AP$, é triangular inferior com $L_{ii} \neq 0$.

A seguir, serão vistas quais as condições necessárias para que uma matriz tenha vetores de direções conjugadas à esquerda, assim como um algoritmo para sua construção.

Seja $A \in \mathbb{R}^{n \times n}$ uma matriz não singular e seja o conjunto $\{\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}\} \subset \mathbb{R}^n$ formado por vetores de direções conjugadas a esquerda de A. Então $\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}$ são linearmente independentes.
De fato, assuma que $P = [\boldsymbol{p_1} \ \boldsymbol{p_2} \ \cdots \ \boldsymbol{p_n}]$. Pela definição, tem-se

$$P^T AP = L$$

sendo L uma matriz triangular inferior e portanto $\boldsymbol{l_{ii}} \neq 0$. Então,

$$\det(P^T AP) = \det(L) \neq 0.$$

Sabendo que $\det(P^T AP) = \det(P^T) \cdot det(A) \cdot \det(P)$ e, da não singularidade de A , segue que $\det(P) \neq 0$. Portanto, $\{\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}\}$ são linearmente independentes.

Lema 2.3.1. *Suponha que $\boldsymbol{p_1}, \cdots, \boldsymbol{p_l} \in \mathbb{R}^n$ sejam direções conjugadas à esquerda de uma matriz A não singular. Seja $S_l = \{\boldsymbol{u} \in \mathbb{R}^n : \boldsymbol{p_i^T} A\boldsymbol{u} = 0 \ \forall \ i = 1, 2, \cdots l\}$. Então para cada $\boldsymbol{u_1} \in S_l \backslash \{0\}$, deve existir algum vetor $\boldsymbol{u_2} \in S_l$ tal que $\boldsymbol{u_1^T} A \boldsymbol{u_2^T} \neq 0$.*

Demonstração: *Sejam $\boldsymbol{p_1}, \cdots, \boldsymbol{p_l}$ direções conjugadas de A e $T_l = span\{\boldsymbol{p_1}, \cdots, \boldsymbol{p_l}\}$. Como os vetores $\boldsymbol{p_i}$ são linearmente independentes, então $dim(T_l) = l$.*
Se $\boldsymbol{u} \in S_l$, pela definição de S_l, tem-se que $A\boldsymbol{u} \in T_l^{\perp}$. Por outro lado, se $\boldsymbol{v} \in T_l^{\perp}$, então $\boldsymbol{p_i^T v} = 0 \ \forall \ i = 1, \cdots, l$ e existe um único $\boldsymbol{u_v} \in \mathbb{R}^n$ tal que $\boldsymbol{v} = A\boldsymbol{u_v}$, pois A é não singular. Decorre que:

$$\forall i = 1, \cdots, l : 0 = \boldsymbol{p_i v} = \boldsymbol{p_i} A \boldsymbol{u_v} \Longrightarrow \boldsymbol{u_v} \in S_l. \tag{62}$$

Sendo assim, existe uma correspondência $1-1$ entre os elementos de S_l e de $T_l^{\perp}$. Portanto

$$dim(S_l) = dim(T_l^{\perp}) = n - l \tag{63}$$

Ainda como consequência da discussão acima,

$$T_l = \{\boldsymbol{v} \in \mathbb{R}^n; \boldsymbol{v^T} A\boldsymbol{u} = 0, \ \forall \ \boldsymbol{u} \in S_l\}. \tag{64}$$

Pela definição de direção conjugada à esquerda, tem-se

$$\boldsymbol{p_{i_0}^T} A \boldsymbol{u_1} = \alpha_{i_0} \boldsymbol{p_{i_0}^T} A \boldsymbol{p_{i_0}} \neq 0.$$

Se $\boldsymbol{u_1^T} A\boldsymbol{u} = 0, \ \forall \boldsymbol{u} \in S_l$, então $\boldsymbol{u_1} \in T_l$ e portanto

$$\boldsymbol{u_1} = \sum_{i=1}^{l} \alpha_i \boldsymbol{p_i}.$$

Como $\boldsymbol{u_1} \neq \boldsymbol{0}$, escolha o menor índice i tal que $\alpha_i \neq 0$, que será denotado por i_0. Desta forma

$$\boldsymbol{p_{i_0}^T} A \boldsymbol{u_1} = \boldsymbol{p_{i_0}^T} A \sum_{i=1}^{l} \alpha_i \boldsymbol{p_i} = \sum_{i=i_0}^{l} \alpha_i \boldsymbol{p_{i_0}^T} A \boldsymbol{p_i}$$

e, pela definição de direções conjugadas à esquerda,

$$\boldsymbol{p_{i_0}^T} A \boldsymbol{u_1} = \alpha_{i_0} \boldsymbol{p_{i_0}^T} A \boldsymbol{p_{i_0}} \neq 0.$$

Por outro lado, como $\boldsymbol{u_1} \in S_l$, $\boldsymbol{p_{i_0}^T} A \boldsymbol{u_1} = 0$. Uma contradição. Logo, existe $\boldsymbol{u_2} \in S_l$ tal que

$$\boldsymbol{u_1^T} A \boldsymbol{u_2} \neq 0.$$

■

Lema 2.3.2. *Suponha que $\boldsymbol{p_1}, \cdots, \boldsymbol{p_l} \in \mathbb{R}^n \ (l < n)$ sejam direções conjugadas à esquerda de uma matriz A não singular. Suponha que $\boldsymbol{u_1}$ e $\boldsymbol{u_2}$ são vetores em S_l com $\boldsymbol{u_1^T} A \boldsymbol{u_2} \neq 0$. Então existem números reais não nulos α e β tais que os vetores $\boldsymbol{p_1}, \cdots, \boldsymbol{p_{l-1}}, \ \boldsymbol{\overline{p}_l} = \boldsymbol{p_l} + \alpha \boldsymbol{u_1}, \ \boldsymbol{\overline{p}_{l+1}} = \boldsymbol{p_l} + \beta \boldsymbol{u_2}$ são $l+1$ direções conjugadas à esquerda de A.*

Demonstração: *Como $\boldsymbol{p_1}, \cdots, \boldsymbol{p_l}$ são direções conjugadas à esquerda de A, para que $\overline{\boldsymbol{p}}_{\boldsymbol{l}}^{\boldsymbol{T}}$ e $\overline{\boldsymbol{p}}_{\boldsymbol{l+1}}^{\boldsymbol{T}}$ também sejam, basta encontrar α e β tais que:*

$$\overline{\boldsymbol{p}}_{\boldsymbol{l}}^{\boldsymbol{T}} A \overline{\boldsymbol{p}}_{\boldsymbol{l}} \neq 0, \tag{65}$$

$$\overline{\boldsymbol{p}}_{\boldsymbol{l+1}}^{\boldsymbol{T}} A \overline{\boldsymbol{p}}_{\boldsymbol{l+1}} \neq 0, \tag{66}$$

$$\overline{\boldsymbol{p}}_{\boldsymbol{l}}^{\boldsymbol{T}} A \overline{\boldsymbol{p}}_{\boldsymbol{l+1}} \neq 0 \tag{67}$$

$$\begin{aligned} \overline{\boldsymbol{p}}_{\boldsymbol{l}}^{\boldsymbol{T}} A \overline{\boldsymbol{p}}_{\boldsymbol{l}} = &\quad \boldsymbol{p}_{\boldsymbol{l}}^{\boldsymbol{T}} A \boldsymbol{p}_{\boldsymbol{l}} + \alpha \boldsymbol{u}_{\boldsymbol{1}}^{\boldsymbol{T}} A \boldsymbol{p}_{\boldsymbol{l}} + \alpha^2 \boldsymbol{u}_{\boldsymbol{1}}^{\boldsymbol{T}} A \boldsymbol{u}_{\boldsymbol{1}} \\ \overline{\boldsymbol{p}}_{\boldsymbol{l+1}}^{\boldsymbol{T}} A \overline{\boldsymbol{p}}_{\boldsymbol{l+1}} = &\quad \boldsymbol{p}_{\boldsymbol{l}}^{\boldsymbol{T}} A \boldsymbol{p}_{\boldsymbol{l}} + \beta \boldsymbol{u}_{\boldsymbol{2}}^{\boldsymbol{T}} A \boldsymbol{p}_{\boldsymbol{l}} + \beta^2 \boldsymbol{u}_{\boldsymbol{2}}^{\boldsymbol{T}} A \boldsymbol{u}_{\boldsymbol{2}} \\ \overline{\boldsymbol{p}}_{\boldsymbol{l}}^{\boldsymbol{T}} A \overline{\boldsymbol{p}}_{\boldsymbol{l+1}} = &\quad \boldsymbol{p}_{\boldsymbol{l}}^{\boldsymbol{T}} A \boldsymbol{p}_{\boldsymbol{l}} + \alpha \boldsymbol{u}_{\boldsymbol{1}}^{\boldsymbol{T}} A \boldsymbol{p}_{\boldsymbol{l}} + \alpha\beta \boldsymbol{u}_{\boldsymbol{1}}^{\boldsymbol{T}} A \boldsymbol{u}_{\boldsymbol{2}}. \end{aligned} \tag{68}$$

Seja $\mathscr{A}$ o conjunto de todos os α tal que $\overline{\boldsymbol{p}}_{\boldsymbol{l}}^{\boldsymbol{T}} A \overline{\boldsymbol{p}}_{\boldsymbol{l}} = 0$, e $\mathscr{B}$ o conjunto de todos os β tais que $\overline{\boldsymbol{p}}_{\boldsymbol{l+1}}^{\boldsymbol{T}} A \overline{\boldsymbol{p}}_{\boldsymbol{l+1}} = 0$. Observe que $\mathscr{A}$ e $\mathscr{B}$ possuem no máximo dois elementos uma vez que, do conjunto de equações (68), tem-se α e β como incógnitas de equações de grau 2. Então, existe algum $\alpha \notin \mathscr{A} \cup \{0\}$ tal que

$$\beta = -\frac{\boldsymbol{p}_{\boldsymbol{l}}^{\boldsymbol{T}} A \boldsymbol{p}_{\boldsymbol{l}} + \alpha \boldsymbol{u}_{\boldsymbol{1}}^{\boldsymbol{T}} A \boldsymbol{p}_{\boldsymbol{l}}}{\alpha \boldsymbol{u}_{\boldsymbol{1}}^{\boldsymbol{T}} A \boldsymbol{u}_{\boldsymbol{2}}} \notin \mathscr{B} \cup \{0\}. \tag{69}$$

Assim, desde que $\alpha \notin \mathscr{A}$ e $\beta \notin \mathscr{B}$, então as duas igualdades (65) e (66) continuam válidas assim como α e β satisfazem (68). Isso completa a prova. ■

A demonstração do Lema (2.3.2) pode ser encontrada em (DAI; YUAN, 2004).

Teorema 2.3.1. *Para cada matriz $A \in \mathbb{R}^{n \times n}$ não singular e não antissimétrica, $A^T \neq -A$, existem n direções conjugadas à esquerda.*

Demonstração: *Desde que A seja não antissimétrica, existe um vetor $\boldsymbol{p_1}$ tal que ${\boldsymbol{p_1}}^T A \boldsymbol{p_1} \neq 0$. Este vetor $\boldsymbol{p_1}$ pode ser considerado como uma direção conjugada à esquerda de A. Geralmente, assume-se que l $(< n)$ direções conjugadas à esquerda $\boldsymbol{p_1}, \cdots, \boldsymbol{p_l}$ foram encontradas. Então pelo Lema (2.3.1), existem $\boldsymbol{u_1}$ e $\boldsymbol{u_2} \in S_l$ que satisfaça ${\boldsymbol{u_1}}^T A \boldsymbol{u_2} \neq 0$, e pelo Lema (2.3.2), existem $l+1$ direções conjugadas à esquerda de A e, por indução, A possui n direções conjugadas à esquerda (DAI; YUAN, 2004).* ■

Seja $A \in \mathbb{R}^{n \times n}$ uma matriz não singular e não antissimétrica e sejam $\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}$ vetores de direções conjugadas à esquerda de A. Dado um vetor $\boldsymbol{x_0} \in \mathbb{R}^n$ fixo, é possível escrever qualquer $\boldsymbol{x} \in \mathbb{R}^n$ como uma combinação única de $\{\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}\}$ na forma

$$\boldsymbol{x} = \boldsymbol{x_0} + \sum_{i=1}^{n} \alpha_i \boldsymbol{p_i}, \tag{70}$$

sendo que α_i $(i = 1, 2, \cdots, n)$ é determinado unicamente por $\{\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}\}$, uma vez que esses vetores são linearmente independentes. Seja $\boldsymbol{x}^*$ a solução do sistema $A\boldsymbol{x} = \boldsymbol{b}$,

segundo a observação (70), tem-se

$$\boldsymbol{x}^* = \boldsymbol{x_0} + \sum_{i=1}^{n} \alpha_i \boldsymbol{p_i} \tag{71}$$

para um dado $\boldsymbol{x_0}$ fixo. Uma vez que $\boldsymbol{x}^*$ é solução de $A\boldsymbol{x} = \boldsymbol{b}$, então

$$A\left(\boldsymbol{x_0} + \sum_{i=1}^{n} \alpha_i \boldsymbol{p_i}\right) = \boldsymbol{b}, \tag{72}$$

portanto

$$A\boldsymbol{x_0} + \sum_{i=1}^{n} \alpha_i A\boldsymbol{p_i} = \boldsymbol{b}. \tag{73}$$

Seja $\boldsymbol{r}$ o vetor residual, ou seja, a diferença entre $\boldsymbol{b}$ e $A\boldsymbol{x}^*$. Sendo assim, tem-se

$$\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}^*. \tag{74}$$

Da Equação (72), tem-se

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{b} - A\boldsymbol{x_0} - \sum_{i=1}^{n} \alpha_i A\boldsymbol{p_i} && (75)\\ &= \boldsymbol{r_0} - \sum_{i=1}^{n} \alpha_i A\boldsymbol{p_i} && (76) \end{aligned}$$

sendo $\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$ o resíduo inicial. Como o objetivo do método LCD é fazer com que o resíduo se anule, $\boldsymbol{r} = \boldsymbol{0}$, é necessário encontrar os coeficientes $\alpha_i,\ i = 1, \cdots, n$ uma vez que os vetores $\boldsymbol{p_1}, \cdots, \boldsymbol{p_n}$ já deverão ser conhecidos. Como os vetores são linearmente independentes, o resíduo $\boldsymbol{r}$ será nulo caso ele seja ortogonal a todo $\boldsymbol{p_i},\ i = 1, \cdots, n$, ou seja,

$$\boldsymbol{p_j}^T \boldsymbol{r} = 0,\ \forall\ j = 1, \cdots, n. \tag{77}$$

Tem-se então

$$\begin{aligned} \boldsymbol{0} &= \boldsymbol{p_i^T}\left(\boldsymbol{r_0} - \sum_{i=1}^{n} \alpha_i A\boldsymbol{p_i}\right) && (78)\\ &= \boldsymbol{p_i^T}(\boldsymbol{r_0} - \alpha_1 A\boldsymbol{p_1} - \cdots - \alpha_i A\boldsymbol{p_i} - \cdots - \alpha_n A\boldsymbol{p_n}) && (79)\\ &= \boldsymbol{p_i^T}\left(\boldsymbol{r_0} - \sum_{j=1}^{i-1} \alpha_j A\boldsymbol{p_j} - \alpha_i A\boldsymbol{p_i} - \alpha_{i+1} A\boldsymbol{p_{i+1}} - \cdots - \alpha_n A\boldsymbol{p_n}\right). && (80) \end{aligned}$$

Sendo

$$\boldsymbol{r_{i-1}} = \boldsymbol{r_0} - \sum_{j=1}^{i-1} \alpha_j A\boldsymbol{p_j},$$

pode-se reescrever (80) como

$$\boldsymbol{p_i^T r_{i-1}} - \alpha_i \boldsymbol{p_i^T} A\boldsymbol{p_i} - \underbrace{\alpha_{i+1} \boldsymbol{p_i^T} A\boldsymbol{p_{i+1}}}_{=0} - \cdots - \underbrace{\alpha_n \boldsymbol{p_i^T} A\boldsymbol{p_n}}_{=0} = 0, \tag{81}$$

e concluir que

$$\boldsymbol{p_i^T r_{i-1}} - \alpha_i \boldsymbol{p_i^T} A \boldsymbol{p_i} = 0, \tag{82}$$

ou seja

$$\alpha_i = \frac{\boldsymbol{p_i^T r_{i-1}}}{\boldsymbol{p_i^T} A \boldsymbol{p_i}} \tag{83}$$

sendo

$$\boldsymbol{r_i} = \boldsymbol{b} - A\boldsymbol{x_i} = \boldsymbol{r_{i-1}} - \alpha_i A \boldsymbol{p_i}, \tag{84}$$

$$\boldsymbol{x_i} = \boldsymbol{x_0} + \sum_{k=1}^{i} \alpha_k \boldsymbol{p_k} = \boldsymbol{x_{i-1}} + \alpha_i \boldsymbol{p_i}. \tag{85}$$

Da discussão anterior, uma vez conhecidos os vetores $\{\boldsymbol{p_1}, \cdots, \boldsymbol{p_n}\}$, n direções conjugadas à esquerda de A, o algoritmo LCD é descrito abaixo.

Algoritmo 12: LCD

Entrada: Matriz A, vetores $\boldsymbol{x_0}$, $\boldsymbol{b}$ e os vetores conjugados à esquerda $\{\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_n}\}$
$\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$
para $k = 1$ *até* $pare$ **faça**
  $\alpha_k = \frac{\boldsymbol{p_k^T r_{k-1}}}{\boldsymbol{p_k^T} A \boldsymbol{p_k}}$
  se $\boldsymbol{p_k^T} A \boldsymbol{p_k} = 0$ **então**
    $Pare$
  fim
  senão
    $\boldsymbol{x_k} = \boldsymbol{x_{k-1}} + \alpha_k \boldsymbol{p_k}$
    $\boldsymbol{r_k} = \boldsymbol{b} - A\boldsymbol{x_k} = \boldsymbol{r_{k-1}} - \alpha_k A \boldsymbol{p_k}$
  fim
fim

Teorema 2.3.2. *Seja $A \in \mathbb{R}^{n \times n}$ uma matriz não singular e não antissimétrica e $\boldsymbol{p_1}, \cdots, \boldsymbol{p_n}$ um conjunto de vetores de direções conjugadas à esquerda de A. Na ausência de erros de arredondamentos, o algoritmo LCD obtém a solução exata do sistema $A\boldsymbol{x} = \boldsymbol{b}$ com no máximo n iterações.*

Demonstração: Pela equação (71), a solução do sistema $A\boldsymbol{x} = \boldsymbol{b}$ pode ser determinada por $\boldsymbol{p_1}, \cdots, \boldsymbol{p_n}$ e pelo vetor $\boldsymbol{x_0} \in \mathbb{R}^{n \times n}$ da seguinte maneira:

$$\boldsymbol{x^*} = \boldsymbol{x_0} + \sum_{i=1}^{n} \alpha_i \boldsymbol{p_i}. \tag{86}$$

O método LCD obtém a solução se existir $k \leq n$ tal que $\boldsymbol{r_k} = 0$. Suponha que $\boldsymbol{r_k} \neq 0, \forall k = 1, \cdots, n-1$. Assim,

$$\boldsymbol{r_n} = \boldsymbol{b} - A\boldsymbol{x_n} = \boldsymbol{b} - A\boldsymbol{x_0} - \sum_{i=1}^{n} \alpha_i A \boldsymbol{p_i}.$$

Escolhe-se α_i *de modo que* $\boldsymbol{r_n}$ *seja ortogonal a todo* $\boldsymbol{p_i}$ $(i = 1, 2, \cdots, n)$ *linearmente independente. Pode-se concluir que* $\boldsymbol{r_n}$ *deve ser nulo e, consequentemente,* $\boldsymbol{x_n}$ *é solução do sistema* $A\boldsymbol{x} = \boldsymbol{b}$*.* ■

2.3.1 Construção das Direções Conjugadas à Esquerda

Seja A uma matriz não singular e não antissimétrica. Uma das dificuldades do método LCD é determinar um conjunto de vetores de direções conjugadas à esquerda de A que sejam linearmente independentes. O modo como serão construídos esses vetores será apresentado a seguir. Suponha que

$$\boldsymbol{p_{k+1}} = \boldsymbol{r_k} + \sum_{i=1}^{k} \beta_i \boldsymbol{p_i}, \tag{87}$$

sendo $\boldsymbol{p_1}$ escolhido de forma que $\boldsymbol{p_1^T A p_1} \neq 0$. Segundo a Definição (2.0.1), para todo $j = 1, 2, \cdots, k$,

$$\boldsymbol{p_j^T A p_{k+1}} = 0,$$

ou seja,

$$\begin{bmatrix} \boldsymbol{p_1^T A p_1} & 0 & \cdots & 0 \\ \boldsymbol{p_2^T A p_1} & \boldsymbol{p_2^T A p_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \boldsymbol{p_k^T A p_1} & \boldsymbol{p_k^T A p_2} & \cdots & 0 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix} = - \begin{bmatrix} \boldsymbol{p_1^T A r_k} \\ \boldsymbol{p_2^T A r_k} \\ \vdots \\ \boldsymbol{p_k^T A r_k} \end{bmatrix} \tag{88}$$

Ao resolver esse sistema, cujas variáveis são $\beta_1, \beta_2, \cdots, \beta_k$, tem-se a seguinte relação de recorrência:

$$\boldsymbol{q_0} = \boldsymbol{r_k} \tag{89}$$

$$\beta_i = -\frac{\boldsymbol{p_i^T A q_{i-1}}}{\boldsymbol{p_i^T A p_i}} \tag{90}$$

$$\boldsymbol{q_i} = \boldsymbol{q_{i-1}} + \beta_i \boldsymbol{p_i} \tag{91}$$

$$\boldsymbol{p_{k+1}} = \boldsymbol{q_k} \tag{92}$$

$$\tag{93}$$

Portanto, se $\boldsymbol{p_1}$ é tal que $\boldsymbol{p_1^T A p_1} \neq 0$, a relação de recorrência acima fornece $\boldsymbol{p_2}, \boldsymbol{p_3}, \cdots, \boldsymbol{p_n}$ e, de acordo Dai e Yuan (2004), o método LCD é descrito pelo seguinte algoritmo:

Algoritmo 13: LCD2

Entrada: Matriz A, vetores $\boldsymbol{x_1}$, $\boldsymbol{b}$, $\boldsymbol{p_1}$ sendo que $\boldsymbol{p_1^T A p_1} \neq 0$.

$\boldsymbol{r_1} = \boldsymbol{b} - A\boldsymbol{x_1}$;

$\boldsymbol{q_1} = A\boldsymbol{p_1}$

para $k = 1$ *até* n **faça**

  $\alpha_k = \boldsymbol{p_k^T r_k} / \boldsymbol{p_k^T q_k}$

  $\boldsymbol{x_{k+1}} = \boldsymbol{x_k} + \alpha \boldsymbol{p_k}$

  $\boldsymbol{r_k + 1} = \boldsymbol{r_k} - \alpha_k \boldsymbol{q_k}$

  $\boldsymbol{p_{k+1}} = \boldsymbol{r_k}$

  $q_{k+1} = A\boldsymbol{p_{k+1}}$

  para $i = 1, 2, \cdots, n$ **faça**

    $\beta_i = \dfrac{\boldsymbol{p_i^T q_{k+1}}}{\boldsymbol{p_i^T q_i}}$

    $\boldsymbol{p_{k+1}} = \boldsymbol{p_{k+1}} - \beta_i \boldsymbol{p_i}$

    $\boldsymbol{q_{k+1}} = \boldsymbol{q_{k+1}} - \beta_i \boldsymbol{q_i}$

  fim

fim

Na primeira parte do algoritmo (13) há o armazenamento de n vetores $\boldsymbol{p_k}$ e n vetores $\boldsymbol{q_k}$, além de dois produtos matriz-vetor para encontrar $\boldsymbol{x_k}$. Um algoritmo reinicializado, como o GMRES(m), é proposto por Catabriga, Coutinho e Franca (2004). Nesse algoritmo, l_{max} é o número máximo de reinicializações, tol é a tolerância para o resíduo, m é o número de vetores de direções conjugadas à esquerda de A para cada processo de reinício. Catabriga, Coutinho e Franca (2004) escolhem $\boldsymbol{p_1} = \boldsymbol{r}$ para $l = 1$ e $\boldsymbol{p_1} = \boldsymbol{p_{k+1}}$ para $l = 2, 3, \cdots$ com base em experimentos numéricos.

Algoritmo 14: LCD(m)

Entrada: Matriz A, vetores $\boldsymbol{x_0}$ e $\boldsymbol{k}$, l_{max}, tol.

$\boldsymbol{r_0} = \boldsymbol{b} - A\boldsymbol{x_0}$;

Determine $\boldsymbol{p_1}$ tal que $\boldsymbol{p_1^T} A\boldsymbol{p_1} \neq 0$;

para $l = 1$ *até* l_{max} **faça**

  para $i = 1$ *até* m **faça**

    $\boldsymbol{q_i} = A^T\boldsymbol{p_i}$;

    $\boldsymbol{\alpha_i} = \dfrac{\boldsymbol{p_i^T r_{i-1}}}{\boldsymbol{q_i^T p_i}}$;

    $\boldsymbol{x_i} = \boldsymbol{x_{i-1}} + \boldsymbol{\alpha_i p_i}$;

    $\boldsymbol{r_i} = \boldsymbol{b} - A\boldsymbol{x_i} = \boldsymbol{r_{i-1}} - \boldsymbol{\alpha_i} A\boldsymbol{p_i}$;

    se $||\boldsymbol{r_i}|| < tol$ **então**

      Pare o loop e $\boldsymbol{x_i}$ é solução;

    fim

    senão

      $\boldsymbol{p_{i+1}} = \boldsymbol{r_i}$;

    fim

    para $j = 1$ *até* i **faça**

      $\boldsymbol{\beta_j} = -\dfrac{\boldsymbol{q_j^T p_{i+1}}}{\boldsymbol{q_j^T p_j}}$;

      $\boldsymbol{p_{i+1}} = \boldsymbol{p_{i+1}} + \boldsymbol{\beta_j p_j}$

    fim

  fim

  Escolha um novo $\boldsymbol{p_1}$ tal que $\boldsymbol{p_1^T} A\boldsymbol{p_1} \neq 0$.

fim

2.3.2 Análise Computacional

Seja A a matriz do sistema $A\boldsymbol{x} = \boldsymbol{b}$, suja dimensão é $(n \times n)$. Para calcular o vetor $\boldsymbol{q_i}$ e o resíduo $\boldsymbol{r_i}$, efetua-se produto matriz-vetor com custo n^2 e $n^2 + n$, respectivamente. Os vetores $\boldsymbol{\alpha_i}$ e $\boldsymbol{x_i}$ realizam $4n - 2$ e n operações. O vetor $\boldsymbol{\beta_j}$ realiza $4n - 2$ enquanto o vetor $\boldsymbol{p_i}$ executa $2n$ operações. Em média, cada iteração executa $2n^2 + 3n(3 + 3m)$ operações. Como todas as operações estão dentro de um loop que varia de 1 até m, ao término do algoritmo são executadas aproximadamente $2n^2m + 3mn(2 + 3m)$ operações, determinando que a complexidade do algoritmo seja $O(2n^2m)$.

Capítulo 3

A Equação de Convecção-Difusão-Reação Transiente

A equação de convecção-reação-difusão-transiente está presente em problemas onde pode ser observado, por exemplo, o transporte de um poluente em um fluido. Sem o termo transiente, podem ser vistas aplicações na equação de Helmholtz para modelagem de acústica exterior (HARARI; HUGHES, 1991), equações constitutivas viscoelásticas para modelar as tensões extras em escoamentos de fluidos não-newtonianos (CROCHET; DAVIES; WALTERS, 2012) e em equações constitutivas para modelar as quantidades de turbulência k e ε (ILINCA; PELLETIER, 1998). Um sistema que trata do escoamento de fluidos em meios porosos, sem o termo reativo é apresentado por (WERNER, 2011). Neste capítulo, será apresentada a formulação numérica dessa equação usando o método de Galerkin no contexto do MEF (SOLIN, 2006). Este capítulo foi baseado no trabalho (MULLER, 2014).

3.1 Caracterização do Problema

Seja $Q = \Omega \times (0,T)$, onde $\Omega \subset \mathbb{R}^2$ é um domínio aberto com fronteira poligonal Γ e $(0,T)$, com $T > 0$ e $T \in \mathbb{R}$, o intervalo de tempo. A equação de convecção-difusão-reação transiente consiste em encontrar a função $u = u(x,t)$, tal que

$$\begin{aligned} \partial_t u + \nabla \cdot (-\epsilon \nabla u + u\beta) + \sigma u &= f, \ em \ Q; && (94)\\ u &= g_D, \ em \ \Gamma_D \times (0,T); && (95)\\ \epsilon(\nabla u \cdot \eta) &= g_N, \ \Gamma_N \times (0,T); && (96)\\ u(x,0) &= u_0(x), \ em \ \Omega, && (97) \end{aligned}$$

sendo que $t \in (0,T)$ é o tempo, $\epsilon > 0$ é o coeficiente de difusão, considerado constante neste trabalho, $x = (x,y) \in \Omega$ é o vetor posição, $\sigma \geq 0$ é o coeficiente de reação, $\beta : Q \times Q \to \mathbb{R}^2$ é o campo de velocidades, $f : Q \to \mathbb{R}$ é o termo de fonte, $g_N : \Gamma_N \times (0,T) \to \mathbb{R}$

é a condição de contorno de Neumann e $u_0(x) : \Omega \to \mathbb{R}$ a solução inicial. A equação(94) representa uma equação diferencial parcial parabólica com condições de contorno prescritas na fronteira (Γ_D e Γ_N) em que Γ_D é a parte da fronteira onde as condições de Dirichlet são prescritas e $\Gamma_N = \Gamma \setminus \Gamma_D$ é a parte da fronteira onde as condições de Neumann são conhecidas. Na equação acima tem-se:

$$\begin{aligned} \text{Laplaciano de } u(x,y) &: \Delta(u) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}; \\ \text{Gradiente de } u(x,y) &: \nabla(u) = \left\langle \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right\rangle; \\ \text{Divergente de } \beta(x,y) &: \nabla \cdot (\beta) = \frac{\partial \beta}{\partial x} + \frac{\partial \beta}{\partial y} = \beta_x + \beta_y. \end{aligned}$$

Da equação (94), tem-se que $\nabla \cdot (-\epsilon \nabla u)$ é o termo difusivo, σu é o termo reativo, $\nabla \cdot (u\beta)$ é o termo convectivo e $\partial_t u$ é o termo transiente. Para que u seja solução da equação de convecção-difusão-reação-transiente, devem ser satisfeitas as seguintes condições: $u \in C^2(Q) \cap C(\overline{Q}), f \in C(\overline{Q}), \beta \in [C(\overline{Q})]^2, g_D \in C(\Gamma_D \times (0,T), g_N \in C(\Gamma_N \times (0,T))$ e $u_0 \in C(\overline{\Omega})$.

Assume-se que o campo de velocidades seja incompressível, ou seja, a densidade do fluido é constante e consequentemente, a massa específica não é função das coordenadas espaciais, tampouco do tempo (FOX, 1998). Esta condição pode ser descrita na seguinte equação:

$$\nabla \cdot \beta \;=\; 0.$$

Desta forma

$$\begin{aligned} \nabla \cdot (-\epsilon \nabla u + u\beta) &= -\epsilon \nabla \cdot (\nabla u) + \nabla \cdot (u\beta) \\ &= -\epsilon \triangle u + u \nabla \cdot \beta + \beta \cdot \nabla u \\ &= -\epsilon \triangle u + \beta \cdot \nabla u. \end{aligned}$$

Pode-se então reescrever o problema (94)-(97) da seguinte maneira:

$$\partial_t u - \epsilon \triangle u + \beta \cdot \nabla u + \sigma u = f, \ em\ \Omega \times (0,T); \tag{98}$$

$$u = g_D, \ em\ \Gamma_D \times (0,T); \tag{99}$$

$$\epsilon(\nabla u \cdot \eta) = g_N, \ em\ \Gamma_N \times (0,T); \tag{100}$$

$$u(x,0) = u_0(x), \ em\ \Omega. \tag{101}$$

3.2 Formulação Variacional

A fim de obter a solução do problema (98)-(101), passa-se à sua formulação variacional. Para isso, alguns espaços de funções, com seus respectivos produtos internos e normas usuais, serão apresentados para serem utilizados.

O espaço de Hilbert $L^2(\Omega)$ das funções quadrado integráveis em Ω é definido por

$$L^2(\Omega) = \left\{ v : \Omega \to \mathbb{R}; \int_\Omega |v|^2 dx < \infty \right\}$$

onde seu produto interno é

$$(u, v) = \int_\Omega u(x, y)v(x, y)d\Omega$$

e a norma por ele induzida é

$$||v||_{L^2(\Omega)} = \sqrt{(v, v)}.$$

Define-se também, para esse problema, os espaços de Hilbert $H^1(\Omega)$ e $H_0^1(\Omega)$ por

$$H^1(\Omega) = \left\{ v \in L^2(\Omega); \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \in L^2(\Omega) \right\};$$

$$H_0^1(\Omega) = \left\{ v \in H^1(\Omega); v = 0 \; em \; \partial\Omega \right\},$$

com produto interno

$$(u, v)_{H^1(\Omega)} = \int_\Omega [u(x, y)v(x, y) + \nabla u(x, y) \cdot \nabla v(x, y)]d\Omega$$

e norma

$$||v||_{H^1(\Omega)} = \sqrt{(u, v)_{H^1(\Omega)}},$$

onde as integrais presentes são definidas no sentido de *Lebesgue* e as derivadas são tomadas no sentido das distribuições (QUARTERONI, 2010).
Seja V um subespaço de H^1 onde

$$V = \left\{ v \in H^1(\Omega); v = 0 \text{ para todo } \mathrm{x} \in \Gamma_D \right\}, \mathrm{x} = (x, y). \tag{102}$$

As funções *teste* $v \in V$ satisfazem as condições de contorno homogêneas em Γ_D e não são dependentes do tempo. A função solução u pertence a outro espaço de funções, conhecido como espaço de funções *admissíveis*. Para o espaço das funções admissíveis é transferida a dependência do tempo da função solução u. Esse espaço de funções é definido por

$$S_{gD} = \left\{ u | u(., t) \in H^1(\Omega), t \in [0, T] \text{ e } u(\mathrm{x}, t) = g_D \text{ para todo } \in \Gamma_D \right\}. \tag{103}$$

Isso equivale a dizer que para cada $t \in [0, T]$ fixo, $u(t) \in H^1(\Omega)$ e satisfaz a condição de Dirichlet, ou seja, $u(t) = g_D$ em Γ_D. A formulação variacional de uma equação diferencial é obtida multiplicando-se a equação por uma função teste e integrando o resultado no domínio. Ao multiplicar a equação (98) por uma função *teste* $v \in V$ e integrando o resultado em Ω, obtém-se a seguinte equação integral:

$$\int_\Omega ((\partial_t u)v - \epsilon \triangle uv + (\beta \cdot \nabla u)v + \sigma uv)d\mathrm{x} = \int_\Omega fvd\mathrm{x}.$$

Ao usar a identidade de Green, pode-se expandir $-\epsilon \int_\Omega \triangle uvd\mathrm{x}$ da seguinte maneira:

$$\begin{aligned} -\epsilon \int_\Omega \triangle uvd\mathrm{x} &= \epsilon \int_\Omega \nabla u \cdot \nabla v d\mathrm{x} - \epsilon \int_\Gamma (\nabla u \cdot \eta) v ds \\ &= \epsilon \int_\Omega \nabla u \cdot \nabla v d\mathrm{x} - \epsilon \int_{\Gamma D} (\nabla u \cdot \eta) v ds - \epsilon \int_{\Gamma N} (\nabla u \cdot \eta) v ds \\ &= \epsilon \int_\Gamma \nabla u \cdot \nabla v d\mathrm{x} - \int_{\Gamma N} g_N v ds, \end{aligned}$$

pois, em Γ_D, $v = 0$ e, em Γ_N, $\epsilon(\nabla u \cdot \eta) = g_N$. A formulação variacional contínua do problema (98)-(101) é, portanto, definida da seguinte forma: dados f, g_D, u_0, para qualquer $t \in (0, T]$, encontrar $u(x, t) \in S_{gD}$, tal que

$$\int_\Omega (\partial_t u) v d\mathrm{x} + \int_\Omega (\epsilon \nabla u \cdot \nabla v + (\beta \cdot \nabla u) v + \sigma uv) d\mathrm{x} = \int_\Omega f v d\mathrm{x} + \int_{\Gamma N} g_N v ds, \ \forall v \in V. \quad (104)$$

É possível reescrever a equação (104) de maneira simplificada

$$(\partial_t u, v) + a(u, v) = l(v), \forall u \in V \quad (105)$$

onde $a(\cdot, \cdot)$ é uma forma bilinear definida por

$$a(u, v) = \int_\Omega (\epsilon \nabla u \cdot \nabla v + (\beta \cdot \nabla u) v + \sigma uv) d\mathrm{x}$$

e $l(\cdot)$ é o operador linear dado por,

$$l(v) = \int_\Omega f v d\mathrm{x} + \int_{\Gamma N} g_N v ds.$$

O problema (98)-(101) é chamado problema clássico ou formulação forte. A equação (105) é chamada formulação variacional. A forma fraca ou formulação variacional é a ferramenta básica para a discretização espacial via método de elementos finitos de um problema de valor de contorno (e inicial). Observe que a integração por partes (via teorema do divergente) do termo difusivo (operador com derivadas de segunda ordem) nos permite introduzir naturalmente a condição de fluxo prescrita em Γ_N, além de reduzir a regularidade imposta sobre u no problema clássico. Ou seja, pode-se buscar uma função u que pertença ao espaço $H^1(\Omega)$, para todo $t \in (0, T]$.

O Método de Galerkin baseia-se em converter o problema variacional contínuo em um problema discreto, cujas funções (*admissíveis e testes*) pertençam a espaços de dimensão finita. O método determina uma aproximação de valores de u em determinados pontos do domínio Ω. Assumindo que as condições de contorno de Dirichlet sejam homogêneas, ou seja, $g_D = 0$, para cada t fixo, os espaços S_{gD} e V são iguais. Nesse caso, considerando

V_h subespaço de dimensão finita de $V = s_0$, o Método de Galerkin consiste em restringir o problema (105) a esse espaço, ou seja, para qualquer $t \in (0, T]$ encontrar $u_h \in V_h$, tal que

$$(\partial_t u_h, v_h) + a(u_h, v_h) = l(v_h), \forall v_h \in V_h \tag{106}$$

sendo que u_h e v_h são aproximações de u e v, respectivamente. As funções u_h e v_h são construídas através de combinações lineares das funções que pertencem à base escolhida do espaço V_h.

Observação 3.2.1. *Os espaços V e V_h são vetoriais, enquanto que S_{gD}, com $g_D \neq 0$ e S_h (sendo que $S_h \subset S_{gD}$ tem dimensão finita) não são. De fato, se u_1 e $u_2 \in S_{gD}$, então $u_1|_{\Gamma D} + u_2|_{\Gamma D} = 2g_D \notin S_{gD}$. Para o método de Galerkin, considera-se o mesmo espaço de aproximação para as funções admissíveis e teste. Para aplicar o método no caso em que $g_D \neq 0$, estende-se $g_D \in C(\Gamma_D)$ para o restante da fronteira Γ introduzindo a função $\tilde{g}_D \in C(\Gamma)$ tal que*

$$\tilde{g}_D = g_D \; em \; \Gamma_D.$$

A função $\tilde{g}_D$ é uma extensão contínua da função g_D. Pode-se considerar para todo $t \in (0, T]$,

$$u_h = \tilde{u}_h + G_h,$$

tal que $\tilde{u}_h \in V_h$ e $G_h \in H^1(\Omega)$ satisfazendo $G_h = \tilde{g}_D$ em Γ. G_h é a extensão da função no domínio todo. Usando esse argumento, o problema (94) pode ser escrito da seguinte maneira: dado $t \in (0, T]$ encontrar $\tilde{u}_h \in V_h$, tal que

$$(\partial_t \tilde{u}_h, v_h) + a(\tilde{u}_h, v_h) = \tilde{l}(v_h), \forall v_h \in V_h, \tag{107}$$

no qual

$$\tilde{l}(v_h) = l(v_h) - [(\partial_t G_h, v_h) + a(G_h, v_h)]. \tag{108}$$

3.3 Elementos Finitos

Será agora feita uma apresentação sobre MEF. Considere uma partição τ_h do domínio $\overline{\Omega}$ em n_{el} subdomínios (triângulos ou quadriláteros) dada por $\tau_h = \{\Omega_e\} = \Omega_1, \Omega_2, \cdots, \Omega_{n_{el}}$ de modo que

$$\Omega = \bigcup_{e=1}^{n_{el}} \Omega_e \quad \text{e} \quad \Omega_i \cap \Omega_j = \emptyset, \quad i, j = 1, 2, \cdots, n_{el}, \; i \neq j.$$

Um exemplo de partição triangular no domínio $[0, 1] \times [0, 1]$ definida de maneira uniforme, ou seja, na qual todos os triângulos possuem o mesmo tamanho, pode ser observada na Figura 1.

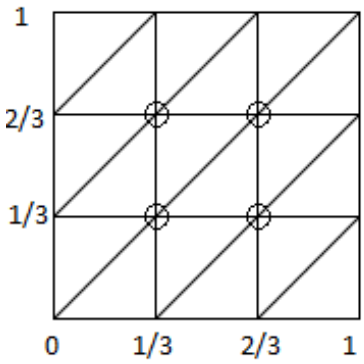

Figura 1 – Triangulação do domínio $[0,1] \times [0,1]$.

Os elementos da partição são chamados de *elementos finitos* e os vértices dos triângulos são chamados de nós da partição (ver Figura 1) . Seja h_e o diâmetro do elemento triangular Ω_e e o parâmetro característico da malha τ_h como sendo $h = max\{h_e\}$. A cada vértice x_j (na figura 1 marcado com o símbolo $\odot$) que não esteja prescrito com condições de contorno de Dirichlet associa-se uma função $\phi_j \in V$ satisfazendo

$$\varphi_j(x_i) = \begin{cases} 1, & \text{se } i = j; \\ 0, & \text{se } i \neq j. \end{cases} \tag{109}$$

Para determinar o espaço V_h, deve-se construir uma base para esse espaço. Seja, então

$$B = \phi_1, \phi_2, \cdots, \phi_n$$

uma base para o espaço V_h associada à partição/triangulação τ_h. Na equação (109), a função φ_j é igual a 1 no nó x_j e 0 nos demais nós da partição. Por definição, φ_j é uma função contínua em $\overline{\Omega}$ e afim $(a + bx + cy)$ em cada triângulo Ω_e, como pode ser observado na figura abaixo.

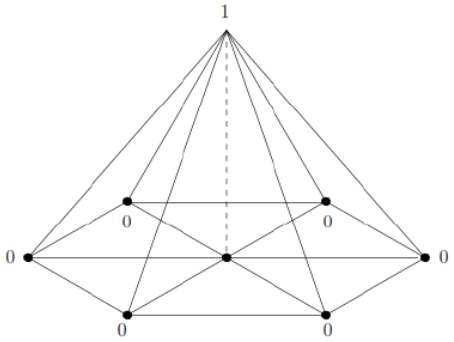

Figura 2 – Função base φ_j - Figura extraída de (SÜLI, 2002)

As funções $\varphi_1, \varphi_2. \cdots, \varphi_n$ são linearmente independentes. Define-se o espaço V_h como o espaço gerado pelas funções φ_j. Por isso, uma função arbitrária $u_h \in V_h$ pode ser escrita

como uma combinação linear dessas funções, ou seja, para cada $t \in (0, T]$,

$$u_h(\mathrm{x}, t) = \sum_{j=1}^{n} \alpha_j \varphi_j(\mathrm{x}). \tag{110}$$

A propriedade (109) implica que em cada vértice $x_i, i = 1, 2, 3, \cdots, n$ da malha,

$$\begin{aligned} u_h(\mathrm{x}_i, t) &= \alpha_1 \underbrace{\varphi_1(\mathrm{x}_i)}_{=0} + \cdots + \alpha_{i-1} \underbrace{\varphi_{i-1}(\mathrm{x}_i)}_{=0} + \alpha_i \underbrace{\varphi_i(\mathrm{x}_i)}_{=1} + \alpha_i \underbrace{\varphi_{i+1}(\mathrm{x}_i)}_{=0} + \cdots + \alpha_n \underbrace{\varphi_n(\mathrm{x}_i)}_{=0} \\ &= \alpha_i \underbrace{\varphi_i(\mathrm{x}_i)}_{=1} \end{aligned}$$

isto é, as coordenadas $\alpha_1, \cdots, \alpha_n$ da combinação linear são dadas por

$$\alpha_i = \alpha_i(t) = u_h(\mathtt{x}_i, t).$$

Ao substituir (110) em (106), obtém-se

$$\left(\partial_t \sum_{j=1}^{n} \alpha_j(t)\varphi_j(\mathtt{x}), v_h\right) + a\left(\sum_{j=1}^{n} \alpha_j(t)\varphi_j(\mathtt{x}), v_h\right) = l(v_h), \ \forall v_h \in V_h.$$

De outro modo

$$\sum_{j=1}^{n} (\varphi_j, \varphi_i)\dot{\alpha}_j(t) + \sum_{j=1}^{n} a(\varphi_j, \varphi_i)\alpha_j(t) = l(\varphi_i), \ i = 1, 2, \cdots, n,$$

que é simplesmente um sistema de equações diferenciais ordinárias que, escrito de forma matricial, é

$$\mathbf{M}\dot{U}(t) + \mathbf{K}U(t) = \mathbf{F}(t), \tag{111}$$

$$U(0) = U_0 \tag{112}$$

sendo que U_0 é a solução inicial (vetor), $\mathbf{M} = [m_{ij}]_{n\times n} \in \mathbb{R}^{n\times n}$ é a matriz com os seguintes elementos

$$m_{ij} = (\varphi_i, \varphi_j) = \int_\Omega \varphi_i \varphi_j d\mathtt{x}$$

$\mathbf{F} = [f_i]_n \in \mathbb{R}^n$ é o vetor com os elementos

$$f_i = (f, \varphi_i) + (g_N, \varphi_i)_{\Gamma_N} = \int_\Omega f\varphi_i d\mathtt{x} + \int_{\Gamma_N} f\varpi_i ds,$$

$\mathbf{K} = [k_{ij}]_{n\times n} \in \mathbb{R}^{n\times n}$ é a matriz com os elementos

$$k_{ij} = a(\varphi_i, \varphi_j) = \int_\Omega \left(\epsilon \nabla\varphi_i \cdot \nabla\varphi_j + (\beta \cdot \nabla\varphi_j)\varphi_i + \sigma\varphi_i\varphi_j\right) d\mathtt{x},$$

$U = [u_j]_n \in \mathbb{R}^n$ é o vetor incógnita com

$$u_j = \alpha_j = u_h(\mathtt{x}_j, t).$$

Pelas condições sobre a equação (111), o sistema (111)-(112) possui solução única devido ao Teorema de Existência e Unicidade para sistemas de EDO's (IÓRIO, 2001), (FIGUEIREDO, 2000). Ao resolver este sistema de equações diferenciais ordinárias, encontra-se uma aproximação para a solução desejada no instante t. Para resolver numericamente, será utilizado o método de diferenças finitas de Crank-Nicolson.

3.3.1 Método de Estabilização SUPG

Quando a convecção domina em relação à difusão em problemas de convecção-difusão, o método de elementos finitos de Galerkin não é adequado para resolvê-los (QUARTERONI, 2010), (BROOKS; HUGHES, 1982). A solução aproximada obtida pelo MEF apresenta oscilações que comprometem a estabilidade do método em situações como a descrita anteriormente. São então utilizados métodos estabilizados (métodos de Petrov-Galerkin) para contornar esses problemas no método de Galerkin. Adiona-se então um termo que está relacionado com o resíduo da equação e é balanceado por um parâmetro de estabilização. Neste trabalho utiliza-se o método de estabilização SUPG (*Streamline Upwind Petrov Galerkin*)(BROOKS; HUGHES, 1982), um dos mais utilizados e conhecidos na solução de problemas dessa natureza, uma vez que os experimentos serão realizados com problemas que possuem convecção dominante. Para resolver o sistema (98)-(101), a formulação SUPG consiste em encontrar $u_h \in V_h \forall t \in (0,T]$ tal que $u_h(\mathtt{x},0) = u_0(\mathtt{x})$ e

$$(\partial_t u_h, v_h) + a(u_h, v_h) + \sum_{\Omega_e \in \tau_h} \int_{\Omega_e} R(u_h)\delta_h \beta \cdot \nabla v_h d\mathtt{x} = l(v_h), \ \forall v_h \in V_h \tag{113}$$

sendo $R(\cdot)$ o resíduo da equação no elemento Ω_e , dado por

$$R(u_h) = \partial_t u_h - \epsilon \triangle u_h + \beta \cdot \nabla u_h + \sigma u_h$$

e o parâmetro de estabilização

$$\delta_h = \frac{h}{2||\beta||_\infty}\xi,$$

com

$$\xi = max\left[0, 1 - \frac{1}{Pe_h}\right]$$

onde

$$Pe_h = \frac{||\beta||_\infty h}{2\epsilon}$$

é o número de Peclet de malha. O acréscimo do termo de estabilização é o que diferencia as formulações (106) e (113) calculado no interior de cada elemento. Nem todas as oscilações são eliminadas pelos métodos de estabilização lineares, como é o caso do SUPG. Métodos de Captura e Descontinuidades são utilizados para eliminar oscilações próximas às regiões de cada limite ou de altos gradientes (métodos de estabilização não-lineares). (QUARTERONI, 2010) fornece maiores detalhes sobre métodos estabilizados.

Como mencionado anteriormente, o método de elementos finitos particiona o domínio Ω em n_{el} subdomínios Ω_e, formando uma malha estruturada. A partir desta fundamentação, serão construídas as matrizes M^e e K^e e o vetor F^e associados a cada elemento da malha, chamados de matrizes e vetor locais. As matrizes **M** e **K** do sistema (111)-(112) são chamandas de matrizes globais e o vetor **F** é chamado de vetor global e serão obtidas a partir das matrizes e vetores locais. A forma utilizada para montar esse sistema global será apresentada a seguir.

Para determinar a matriz **K** inicialmente deveria calcular uma integral sobre todo o domínio Ω. Será feito então o cálculo dessa integral sobre cada elemento Ω_e da seguinte maneira:

$$\begin{aligned} k_{ij} &= \int_{\Omega} (\epsilon \nabla\varphi_i \cdot \nabla\varphi_j \ + (\beta \cdot \nabla\varphi_j)\varphi_i + \sigma\varphi_i\varphi_j)\, d\mathtt{x} \\ &= \sum_{e=1}^{n_e l} \int_{\Omega_e} (\epsilon \nabla\varphi_i \cdot \nabla\varphi_j + (\beta \cdot \nabla\varphi_j)\varphi_i + \sigma\varphi_i\varphi_j)\, d\mathtt{x} \end{aligned}$$

em que

$$\int_{\Omega_e} (\epsilon \nabla\varphi_i \cdot \nabla\varphi_j + (\beta \cdot \nabla\varphi_j)\varphi_i + \sigma\varphi_i\varphi_j)\, d\mathtt{x} = 0$$

caso φ_i e/ou φ_j não sejam funções associadas aos pontos nodais do elemento Ω_e, uma vez que as funções φ_i ou φ_j serão nulas no restante do domínio. Como consequência, há a necessidade de se calcular apenas as integrais que não se anulam sobre os elementos e pode-se armazenar esse valores em uma matriz com dimensão 3×3, conhecida como matriz local. Serão então construídas todas as matrizes locais para montar a matriz global.

Para os cálculos necessários no nível de elementos, sejam $\varphi_j^e = \varphi_j^e(\mathtt{x}), j = 1, 2, 3$ as funções conhecidas como *de forma local* do método de elementos finitos de Galerkin. Como já mencionado, essas funções são lineares em cada elemento Ω_e e serão escritas a seguir. Uma vez que a partição adotada é triangular, a função linear em Ω_e é dada por

$$p_i(x, y) = a + bx + cy, \ \forall (x, y) = \mathtt{x} \in \Omega_e.$$

Com o objetivo de determinar as constantes a, b, c, faça-se:

$$v_1 = p_i(x_1, y_1) = a + bx_1 + cy_1 \tag{114}$$

$$v_2 = p_i(x_2, y_2) = a + bx_2 + cy_2 \tag{115}$$

$$v_3 = p_i(x_3, y_3) = a + bx_3 + cy_3 \tag{116}$$

em que $(x_i, y_i) = \mathtt{x}_j, j = 1, 2, 3$ são as coordenadas dos três vértices do triângulo Ω_e e v_1, v_2, v_3 é o valor da função em cada um dos vértices. No caso, a função possui valor 0 ou 1 dependendo do vértice que está sendo analisado no elemento Ω_e. Seja A^e a área do elemento Ω_e que pode ser calculada por:

$$2A^e = det \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} = (x_1y_2 - x_2y_1) + (x_3y_1 - x_1y_3) + (x_2y_3 - x_3y_2).$$

$$\begin{aligned} a &= \frac{1}{2A^e}[v_1(x_2y_3 - x_3y_2) + v_2(x_3y_1 - x_1y_3) + v_3(x_1y_2 - x_2y_1)]; \\ b &= \frac{1}{2A^e}[v_1(y_2 - y_3) + v_2(y_3 - y_1) + v_3(y_1 - y_2)]; \\ c &= \frac{1}{2A^e}[v_1(x_3 - x_2) + v_2(x_1 - x_3) + v_3(x_2 - x_1)] \end{aligned}$$

Eliminando a, b e c e escrevendo $p_i(x, y)$ em função de $v_1,\ v_2$ e v_3, tem-se

$$p_i(x, y) = v_1\varphi_1^e(x, y) + v_2\varphi_2^e(x, y) + v_3\varphi_3^e(x, y)$$

sendo que

$$\varphi_1^e(x, y) = \frac{1}{2A^e}[(x_2y_3 - x_3y_2) + (y_2 - y_1)x + (x_3 - x_2)y];$$
$$\varphi_2^e(x, y) = \frac{1}{2A^e}[(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y];$$
$$\varphi_3^e(x, y) = \frac{1}{2A^e}[(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y];$$

são as funções locais associadas aos três vértices do triângulo Ω_e que podem ser reescritas do seguinte modo:

$$\varphi_1^e(x, y) = \frac{1}{2A^e}[a_1 + b_1x + c_1y];$$
$$\varphi_2^e(x, y) = \frac{1}{2A^e}[a_2 + b_2x + c_2y];$$
$$\varphi_3^e(x, y) = \frac{1}{2A^e}[a_3 + b_3x + c_3y];$$

em que

$$a_1 = x_2y_3 - x_3y_2 \quad b_1 = y_2 - y_1 \quad c_1 = x_3 - x_2;$$
$$a_2 = x_3y_1 - x_1y_3 \quad b_2 = y_3 - y_1 \quad c_2 = x_1 - x_3;$$
$$a_3 = x_1y_2 - x_2y_1 \quad b_3 = y_1 - y_2 \quad c_3 = x_2 - x_1.$$

Para resolver as integrais presentes nas formulações, pode-se usar métodos de integração numérica. Como, nesse caso, as funções são lineares, essas integrais serão resolvidas de forma analítica usando-se a fórmula de integração sobre cada elemento Ω_e (ROLAND; PERUMAL; KANKANHALLI, 2004), definida por

$$\int_{\Omega_e} (\varphi_1^e)^m(\varphi_2^e)^n(\varphi_3^e)^r d\mathtt{x} = \frac{m!n!r!}{(m + n + r + 2)!}2A^e.$$

Seja Γ_{ij}^e aresta do elemento Ω_e que contenha dois vértices. As integrais que serão calculadas sobre as arestas $\Gamma_{12}^e, \Gamma_{13}^e$ e Γ_{23}^e de Ω_e são resolvidas por

$$\int_{\Gamma_{ij}^e} (\varphi_l)^m(\varphi_k)^n ds = \frac{m!n!}{(m + n + 1)!}|\Gamma_i^e|,$$

no qual $|\Gamma_{ij}^e|$ é o comprimento da aresta Γ_{ij}^e, caso os vértices l e k estejam sobre a aresta Γ_{ij} e 0 caso contrário.

A matriz **M**, associada ao termo transiente, é obtida a partir da contribuição de cada matriz local $M^e = [m_{ij}^e]_{3\times3}$ sendo

$$m_{ij} = \int_{\Omega_e} (\varphi_i^e)(\varphi_j^e) d\mathtt{x} = \begin{cases} \frac{A^e}{6}, & se,\ i = j; \\ \frac{A^e}{12}, & se,\ i \neq j. \end{cases} \tag{117}$$

Nesse caso, escrevendo em forma matricial os valores de $[m_{ij}^e]_{3\times 3}$, tem-se:

$$M^e = \frac{A^e}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}. \tag{118}$$

3.3.2 Matriz Local K

Sejam D^e, C^e e R^e as matrizes locais associadas aos termos de difusão, convecção e reação. Desta forma, a matriz local K^e pode ser escrita como

$$K^e = D^e + C^e + R^e \tag{119}$$

Observe que uma vez que $a(\cdot,\cdot)$ não é simétrica, a matriz K^e e portanto **K** não são matrizes simétricas. A matriz local associada ao termo de difusão é dada por $D^e = [d_{ij}^e]_{3\times 3}$ em que

$$d_{i,j}^e = \int_{\Omega_e} \epsilon \nabla\varphi_i^e(x,y)\cdot\nabla\varphi_j^e(x,y)d\mathtt{x},$$

sendo

$$\nabla\varphi_k^e(x,y) = \left\langle \frac{\partial\varphi_k^e}{\partial x}, \frac{\partial\varphi_k^e}{\partial y}\right\rangle = \frac{1}{2A^e}\left\langle b_k, c_k\right\rangle.$$

Pode-se então verificar que

$$\epsilon\nabla\varphi_i^e(x,y)\cdot\nabla\varphi_j^e(x,y) = \epsilon\left\langle \frac{b_i}{2A^e}, \frac{c_i}{2A^e}\right\rangle\left\langle \frac{b_j}{2A^e}, \frac{c_j}{2A^e}\right\rangle = \frac{\epsilon}{4(A^e)^2}(b_ib_j + c_ic_j)$$

e, então

$$d_{ij}^e = \int_{\Omega_e} \epsilon\nabla\varphi_i^e(x,y)\cdot\nabla\varphi_j^e(x,y)d\mathtt{x} = \frac{\epsilon}{4(A^e)^2}(b_ib_j + c_ic_j)\underbrace{\int_{\Omega_e} d\mathtt{x}}_{=A^e} = \frac{\epsilon}{4A^e}(b_ib_j + c_ic_j).$$

O elemento d_{11}, por exemplo, é:

$$d_{11}^e = \frac{1}{4A^e}\left[(y_2-y_3)(y_2-y_3) + (x_3-x_2)(x_3-x_2)\right].$$

A matriz correspondente ao termo de convecção é definida por $C^e = [c_{ij}^e]_{3\times 3}$ sendo

$$c_{ij}^e = \int_{\Omega_e} (\beta\cdot\nabla\varphi_j^e(x,y))\varphi_i^e(x,y)d\mathtt{x}.$$

Como $\beta : Q\times Q \to \mathbb{R}^2$, pode-se escrever $\beta = \langle\beta_x, \beta_y\rangle$. Assim, tem-se

$$\beta\cdot\nabla\varphi_j^e(x,y) = \langle\beta_x,\beta_y\rangle\left\langle \frac{b_j}{2A^e}, \frac{c_j}{2A^e}\right\rangle = \frac{1}{2A^e}(b_j\beta_x + c_j\beta_y).$$

Ao considerar as velocidades β_x e β_y constantes em Ω_e, tem-se:

$$c_{ij}^e = \frac{1}{2A^e}(b_j\beta_x + c_j\beta_y)\underbrace{\int_{\Omega_e}\varphi_i^e(x,y)dx}_{A^e/3} = \frac{1}{6}(b_j\beta_x + c_j\beta_y).$$

A matriz local referente ao termo de reação é definida por $R^e = [r_{ij}^e]_{3\times 3}$ sendo

$$r_{ij}^e = \int_{\Omega_e}\sigma(\varphi_i^e)(\varphi_j^e)d\mathtt{x} = \begin{cases} \frac{\sigma A^e}{6}, & se,\ i = j; \\ \frac{\sigma A^e}{12}, & se,\ i \neq j. \end{cases} \tag{120}$$

Pode-se, então, construir a matriz referente ao termo reativo

$$R^e = \frac{\sigma A^e}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

3.3.3 Vetor Local F

As condições de contorno de Dirichlet e Neumann e o termo de fonte estão relacionados com a construção do vetor global **F**. Da equação (108), segue que o vetor **F** está relacionado a

$$\int_\Omega f v d\mathtt{x} + \int_{\Gamma_N} g_N v ds - \int_\Omega (\partial_t G_h v_h dx + \epsilon\nabla G_h\cdot\nabla v_h + \beta\nabla G_h v_h + \sigma G_h v_h)d\mathtt{x} \tag{121}$$

A cada elemento Ω_e, assim como na construção das matrizes globais, tem-se um vetor **F** associado $F^e = [f_i^e]_{3\times 1}$ no qual

$$f_i^e = \int_{\Omega_e} f\varphi_i^e d\mathtt{x} + \int_{\Gamma_N^e} g_N\varphi_i^e ds - \int_{\Omega_e}(\partial_t G_h^e\varphi_i^e + \epsilon\nabla G_h^e\cdot\nabla\varphi_i^e + \beta\cdot\nabla G_h^e\varphi_i^e + \sigma G_h^e\varphi_i^e)d\mathtt{x},$$

$i = 1,2,3$, no qual G_h^e é a restrição da função G_h em Ω_e e Γ_N^e é a aresta de Ω com condições de Neumann. A função G_h é definida por (GOCKENBACH, 2006)

$$G_h(x_j,y_j) = \begin{cases} g_D, & \text{se } (x_j,y_j)\in\Gamma_D; \\ 0, & \text{caso contrário.} \end{cases}$$

Inicialmente, será calculada a primeira integral em (121). Usando interpolação, pode-se aproximar a função f, para cada $t\in(0,T]$ no elemento Ω_e, da seguinte maneira

$$f(\mathtt{x},t)|_{\Omega_e} = f_1\varphi_1^e(x,y) + f_2\varphi_2^e(x,y) + f_3\varphi_3^e(x,y),$$

sendo $f_i = f(x_i,y_i,t)$ o valor da função f no vértice $v_i = (x_i,y_i), i = 1,2,3$ do triângulo Ω_e, para cada $t\in[0,T]$. Nesse caso, ao calcular essas integrais para $i = 1,2,3$, tem-se o seguinte vetor local associado ao termo $\int_{\Omega_e} f v_h d\mathtt{x}$

$$\frac{A^e}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \frac{A^e}{12}\begin{bmatrix} 2f_1 + f_2 + f_3 \\ f_1 + 2f_2 + f_3 \\ f_1 + f_2 + f_3 \end{bmatrix}.$$

A integral $\int_{\Gamma_N^e} g_N \varphi_i^e ds$, que possui parte na fronteira de Ω, trata-se de uma integral de linha sobre um dos lados do triângulo. Considere um elemento do domínio Ω como o representado abaixo.

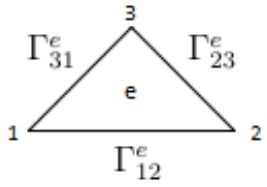

Figura 3 – Elemento Ω_e

A fronteira $\partial\Omega_e$ de um elemento pode ser descrita como:

$$\partial\Omega_e = \Gamma_{12}^e \cup \Gamma_{23}^e \cup \Gamma_{31}^e.$$

Como Ω_e pertence a partição do domínio Ω, deve-se notar que no máximo duas partes da fronteira estarão ativas em Γ_N^e. Neste caso, a função $\varphi_l^e, l \neq j$ e $l \neq k$ é tal que

$$\varphi_l^e|_{\Gamma_{jk}^e} \equiv 0. \tag{122}$$

Pode-se aproximar g_N usando interpolação da seguinte maneira

$$g_N = g_{N1}\varphi_1^e + g_{N2}\varphi_2^e + g_{N3}\varphi_3^e$$

no qual

$$g_{N_l} = \begin{cases} 0, \ se\ l \neq j \text{ e } l \neq k; \\ g_N(x_e, y_e)t, \ \text{caso contrário.} \end{cases}$$

Portanto

$$I_i = \int_{\Gamma_N}^e g_N \varphi_i^e ds = \int_{\Gamma_{jk}^e} \left(g_{N1}\varphi_1^e + g_{N2}\varphi_2^e + G_{g3}\varphi_3^e +\right) ds \tag{123}$$

$$= g_{N1}\int_{\Gamma_e} \varphi_1^e \varphi_i^e ds + g_{N2}\int_{\Gamma_e} \varphi_2^e \varphi_i ds + g_{N3}\int_{\Gamma_e} \varphi_3^e \varphi_i^e ds \tag{124}$$

Da equação (122), tem-se que se $i \neq j$ e $i \neq k$ então $I_i = 0$ uma vez que

$$\int_{\Gamma_{jk}^e} \varphi_l^e \varphi_i^e ds = 0, \ \forall\ l = 1, 2, 3.$$

$$I_i = 0 = G_{N1} \cdot 0 + G_{N2} \cdot 0 + G_{N3} \cdot 0$$

Contudo, se $i = j$ ou $i = k$ (não ocorrerá $i = j$ e $i = k$, uma vez que $j \neq k$), então I_i possuirá apenas uma de suas parcelas nulas devido à definição de g_{Nl}. Tal parcela

corresponde ao índice $l \neq j$ e $l \neq k$. As parcelas não nulas podem ser calculadas pela fórmula conhecida

$$\int_{\Gamma^e_{jk}} \varphi^e_l \varphi^e_i ds = \begin{cases} \frac{|\Gamma^2_i|}{3}, & se,\ i = j; \\ \frac{|\Gamma^e_i|}{6}, & se,\ i \neq j. \end{cases}$$

Admitindo uma malha uniforme em que Γ^e_{ij} possua sempre o mesmo comprimento na fronteira de Ω e $|\Gamma^e_{ij}| = L$, então:

$$\int_{\Gamma^e_{jk}} \varphi^e_l \varphi^e_i ds = \frac{L}{6} \begin{cases} 2,\ se\ l = i; \\ 1,\ sel\ \neq i. \end{cases}$$

Para exemplificar, assuma que o elemento Ω_e esteja na fronteira prescrita de Neumann e que os vértices 2 e 3 pertençam à fronteira. Nesse caso, pode-se escrever

$$\int_{\partial\Omega_e} Gn\varphi_i = \int_{\Gamma^e_i} Gn\varphi_1 + \int_{\Gamma^e_i} Gn\varphi_2 + \int_{\Gamma^e_i} Gn\varphi_3.$$

A fim de expressar de forma matricial local a contribuição na fronteira de Neumann, tem-se

$$\delta_{ij} + \delta_{ik} = \begin{cases} 0, \text{se } i \neq j \text{ e } i \neq k \\ 1, \text{caso contrário.} \end{cases}$$

$$\delta_{lj} + \delta_{lk} = \begin{cases} 0,\ \text{se } l \neq j \text{ e } l \neq k \\ 1,\ \text{caso contrário.} \end{cases}$$

$$1 + \delta_{il} = \begin{cases} 2,\ \text{se } i = j \\ 1,\ \text{caso contrário.} \end{cases}$$

Portanto, pode-se definir a matriz $N = (N_{il})_{3\times 3}$ como

$$n_{il} = \frac{L}{6}(\delta_{ij} + \delta_{jk})(\delta_{lj} + \delta_{lk})(1 + \delta_{il}) = \begin{cases} 0,\ \text{se } (i \neq j \text{ e } i \neq k) \text{ ou } (l \neq j \text{ e } l \neq k) \\ \frac{2L}{6},\ \text{se } i = l \\ \frac{1}{6},\ \text{se } i \neq l \end{cases} \quad (125)$$

Observe que se $i \neq j$ e $i \neq k$ a matriz N possui a linha i e a coluna j nulas e as demais entradas conforme a matriz

$$\frac{L}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

O vetor local F associado à integral $\int_{\Gamma_N} G_N \varphi_i ds$ é

$$N \cdot \begin{bmatrix} G_{N1} \\ G_{N2} \\ G_{N3} \end{bmatrix}$$

Para exemplificar, assuma que o elemento Ω é tal que a fronteira Γ^e_{23} é do tipo Neumann, então:

$$N \begin{bmatrix} G_{N1} \\ G_{N2} \\ G_{N3} \end{bmatrix} = \frac{L}{6} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} G_{N1} \\ G_{N2} \\ G_{N3} \end{bmatrix} = \frac{L}{6} \begin{bmatrix} 0 \\ 2G_{N_2} + G_{N_3} \\ G_{N_2} + 2G_{N_3} \end{bmatrix}.$$

Quando $\Gamma^e_N = \Gamma^e_{jk} \cup \Gamma^e_{kl}$, fica claro que soma-se as contribuições em cada parte da fronteira. Portanto

$$\begin{aligned}
\int_{\Gamma^e_1} G_N \varphi^e_1 d\mathtt{x} &= G_{N1} \int_{\Gamma^e_1} \varphi^e_1 \varphi^e_1 d\mathtt{x} + G_{N2} \int_{\Gamma^e_1} \varphi^e_2 \varphi^e_1 d\mathtt{x} + G_{N1} \int_{\Gamma^e_1} \varphi^e_1 \varphi^e_1 d\mathtt{x} \\
\int_{\Gamma^e_2} G_N \varphi^e_2 d\mathtt{x} &= G_{N1} \int_{\Gamma^e_2} \varphi^e_1 \varphi^e_2 d\mathtt{x} + G_{N2} \int_{\Gamma^e_2} \varphi^e_2 \varphi^e_2 d\mathtt{x} + G_{N1} \int_{\Gamma^e_2} \varphi^e_3 \varphi^e_2 d\mathtt{x} \\
\int_{\Gamma^e_3} G_N \varphi^e_3 \mathtt{x} &= G_{N1} \int_{\Gamma^e_3} \varphi^e_1 \varphi^e_3 d\mathtt{x} + G_{N2} \int_{\Gamma^e_3} \varphi^e_2 \varphi^e_3 d\mathtt{x} + G_{N1} \int_{\Gamma^e_3} \varphi^e_3 \varphi^e_3 d\mathtt{x}
\end{aligned}$$

Seja H a matriz associada ao termo $\displaystyle\int_{\Gamma^e_i} \varphi^e_i \varphi^e_j d\mathtt{x}$. Como o vértice 1 não pertence à fronteira, toda integral que possua a função φ^e_1 será nula. Portanto, para o termo $\displaystyle\int_{\Gamma_e} G_N \varphi^e_i ds$, com as características acima, tem-se o seguinte vetor associado

$$H \cdot \begin{bmatrix} G_{N1} \\ G_{N2} \\ G_{N3} \end{bmatrix} = \frac{|\Gamma^e_i|}{6} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} G_{N1} \\ G_{N2} \\ G_{N3} \end{bmatrix} = \frac{|\Gamma^e_i|}{6} \begin{bmatrix} 0 \\ 2G_{N2} + G_{N3} \\ G_{N2} + 2G_{N3} \end{bmatrix}$$

Caso o vértice não pertencente à fronteira seja o vértice 2, é fácil ver que o vetor associado ao termo $\displaystyle\int_{\Gamma_e} G_N \varphi_{\partial_e} ds$ será

$$\frac{|\Gamma^e_i|}{6} \begin{bmatrix} 2 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} G_{N1} \\ G_{N2} \\ G_{N3} \end{bmatrix} = \frac{|\Gamma^e_i|}{6} \begin{bmatrix} 2G_{N1} + G_{N3} \\ 0 \\ G_{N1} + 2G_{N3} \end{bmatrix}$$

Para o vértice 3, o vetor associado é

$$\frac{|\Gamma^e_i|}{6} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} G_{N1} \\ G_{N2} \\ G_{N3} \end{bmatrix} = \frac{|\Gamma^e_i|}{6} \begin{bmatrix} 2G_{N1} + G_{N2} \\ G_{N1} + 2G_{N2} \\ 0 \end{bmatrix}.$$

Usando interpolação, pode-se escrever a função G^e_h como

$$G^e_h = g_1 \varphi^e_1(x, y) + g_2 \varphi^e_2(x, y) + g_3 \varphi^e_3(x, y)$$

sendo

$$g_i = G^e_h(x_i, y_i, t)$$

o valor que, no tempo $t \in [0,T]$ fixo, G_h^e assume no vértice $(x_i, y_i),\ i = 1,2,3$ do triângulo Ω_e . Se $(x_i, y_i) \in \Gamma_D$, então $g_i = g_D(x_i, y_i),$ caso contrário $g_i = 0$. Assumindo que g_D é constante no tempo, tem-se que

$$\int_\Omega \partial_t G_h v_h d\mathtt{x} = 0$$

em (121), uma vez que esse vetor local representa a condição de contorno de Dirichlet. Consequentemente, o vetor local associado a

$$\int_\Omega (\partial_t G_h v_h dx + \epsilon \nabla G_h \cdot \nabla v_h + \beta \nabla G_h v_h + \sigma G_h v_h) d\mathtt{x}$$

pode ser escrito como

$$\begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} E_{11}g_1 + E_{12}g_2 + E_{13}g_3 \\ E_{21}g_1 + E_{22}g_2 + E_{23}g_3 \\ E_{31}g_1 + E_{32}g_1 + E_{33}g_1 \end{bmatrix}$$

sendo que a mesma matriz local $(E_{ij})_{3\times 3}$ é a mesma matriz local K^e, descrita em (119) associada ao termo $a(G_h, v_h)$. Somente quando o vértice (x_i, y_i) pertence à fronteira prescrita de Dirichlet, tem-se $g_i = G_h(x_i, y_i)$ diferente de zero. O vetor local F^e é, portanto, dado por

$$F^e = \begin{bmatrix} f_1^e \\ f_2^e \\ f_3^e \end{bmatrix} = \frac{A^e}{12} \begin{bmatrix} 2f_1 + f_2 + f_3 \\ f_1 + 2f_2 + f_3 \\ f_1 + f_2 + f_3 \end{bmatrix} - \begin{bmatrix} E_{11}g_1 + E_{12}g_2 + E_{13}g_3 \\ E_{21}g_1 + E_{22}g_2 + E_{23}g_3 \\ E_{31}g_1 + E_{32}g_1 + E_{33}g_1 \end{bmatrix} + \frac{|\Gamma_e|}{6} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}.$$

em que $[\ v_1\ v_2\ v_3\ ]^T$ é o vetor associado ao termo $\int_{\partial\Omega_e} G_N \varphi_i^e ds$.

3.3.4 Implementação Computacional

Para os algoritmos envolvidos na solução numérica do problema, considera-se:

- neq: número de equações, ou seja, a ordem do sistema global a ser resolvido.
- nel: número de elemento da malha τ_h.
- $nnos$: números de nós da malha τ_h.

Algumas estruturas também serão utilizadas. Entre elas

- COORD: matriz coordenada - matriz que armazena as coordenadas (x, y) dos pontos da malha.

- ID: matriz com $nnos$ linhas e 2 colunas. A primeira coluna identifica a equação que está associada a cada vértice global não prescrito (vértices livres e os que possuem condições de contorno de Neumann). A segunda coluna identifica se a condição de contorno é de Dirichlet, Neumann ou se o vértice é um vértice livre
- IEN: matriz de conectividade - matriz que relaciona o elemento com seus respectivos vértices globais.
- LM: matriz de localização - matriz que associa os vértices locais do elemento ao número da equação correspondente.
- BOUNDCOND: vetor que armazena os valores prescritos na fronteira (condições de contorno de Dirichlet).

Um exemplo que ilustra os elementos anteriores será mostrado a seguir. Considere a malha representada na figura (4).

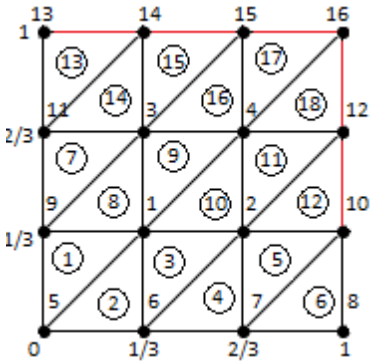

Figura 4 – Malha estruturada de elementos finitos com 16 vértices e 18 elementos de uma partição [0,1] $\times$ [0,1].

Essa malha possui 18 elementos triangulares e 16 vértices. A enumeração dos elementos, importante aspecto na estruturação do método de elementos finitos, é feita inicialmente com os elementos que não se encontram na fronteira. Definida a partição, pode-se associar cada vértice com sua respectiva coordenada (x, y). Os vértices 5, 6, 7, 8, 9 e 11 são prescritos com condições de contorno de Dirichlet. Os vértices 10, 12, 13, 14, 15 e 16 são prescritos com condições de Neumann. Aos demais vértices, dá-se o nome de vértices livres. Com isso, as variáveis nel, neq e $nnos$ terão os seguintes valores:

- nel: 18 (número de elementos);
- $nnos$: 16 (número total de vértices).

- neq: 4(número de vértices livres $=$ número total de vértices $-$ número de vértices prescritos);

Será agora construída a matriz COORD, que armazena as coordenadas dos vértices da malha. Como a malha é estruturada e está definida no domínio [0,1] $\times$ [0,1], tem-se uma matriz com 16 linhas e 2 colunas, associando cada ponto nodal à sua coordenada (x, y). Nesse caso, COORD$[i][j]$ representa a j-ésima coordenada (x, y) do i-ésimo vértice da malha. Tem-se então

$$\text{COORD}[i][j] = \begin{bmatrix} 0 & 0 \\ 1/3 & 0 \\ 2/3 & 0 \\ 1 & 0 \\ 0 & 1/3 \\ 0 & 2/3 \\ \vdots & \vdots \\ 2/3 & 1 \\ 1 & 1 \end{bmatrix}$$

Para a construção da matriz ID, que identifica a equação que está associada a cada vértice global $livre$ e se esse vértice está prescrito com condições de contorno de Neumann, tem-se uma matriz cuja dimensão é $nnos$ linhas e 2 colunas sendo que

$$ID[i][1] = \begin{cases} eq, \text{se } i \text{ é um ponto nodal livre;} \\ 0, \text{ se } i \text{ é um ponto nodal prescrito.} \end{cases}$$

e

$$ID[i][2] = \begin{cases} 0, \text{se vértice } i \text{ possui condição de contorno de Dirichlet;} \\ 1, \text{se vertice } i \text{ possui condição de contorno de Neumann;} \\ 2, \text{se vértice } i \text{ vértice livre.} \end{cases}$$

no qual eq é o número da equação associada ao nó i. Para a malha apresentada na Figura 4, tem-se

$$ID = \begin{bmatrix} 1 & 2 & 3 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$$

Para a construção da matriz IEN, que possui nel linhas e 3 colunas, deve-se associar cada elemento $e = 1,\ 2, \cdots, nel$ a seus respectivos vértices globais. Tem-se então

$$IEN = \begin{bmatrix} n_{1,2} & n_{1,2} & n_{1,3} \\ n_{2,1} & n_{2,2} & n_{2,3} \\ n_{3,1} & n_{3,2} & n_{3,3} \\ \vdots & \vdots & \vdots \\ n_{nel,1} & n_{nel,2} & n_{nel,3} \end{bmatrix}$$

sendo $n_{i,j}$ o vértice global do elemento Ω_i que está associado ao vértice local j, ou seja,

$$IEN[elemento][noLocal] = noGlobal.$$

Para construir a matriz de localização LM, com nel linhas e 3 colunas, associa-se os vértices locais de cada elemento ao número da equação correspondente. No caso da malha na Figura 4, tem-se as seguintes matrizes:

$$IEN = \begin{bmatrix} 5 & 1 & 9 \\ 5 & 6 & 1 \\ 6 & 2 & 1 \\ 6 & 7 & 2 \\ 7 & 10 & 2 \\ 7 & 8 & 10 \\ 9 & 3 & 11 \\ 9 & 1 & 3 \\ 1 & 4 & 3 \\ 1 & 2 & 4 \\ 2 & 12 & 4 \\ 2 & 10 & 12 \\ 11 & 14 & 13 \\ 11 & 3 & 14 \\ 3 & 15 & 14 \\ 3 & 4 & 15 \\ 4 & 16 & 15 \\ 4 & 12 & 16 \end{bmatrix}. \; LM = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 1 & 3 \\ 1 & 4 & 3 \\ 1 & 2 & 4 \\ 2 & 0 & 4 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 3 & 0 \\ 3 & 0 & 0 \\ 3 & 4 & 0 \\ 4 & 0 & 0 \\ 4 & 0 & 0 \end{bmatrix}.$$

Com a matriz ID e a matriz IEN, pode-se construir a matriz de localização LM da seguinte forma

$$LM[elem][noLocal] = ID[IEN[elem][noLocal]][1]$$

Os valores prescritos na fronteira formam o vetor $BOUNDCOND$ da seguinte maneira:

$$BOUNDCOND = [x \; x \; x \; x \; g_{D5} \; g_{D6} \; g_{D7} \; g_{D8} \; g_{D9} \; g_{N10} \; g_{D11} \; g_{N12} \; g_{N13} \; g_{N14} \; g_{N15} \; g_{N16} \;]^T,$$

sendo que $g_{Ni} = g_N(x_i, y_i)$ e $g_{Di} = g_D(x_i, y_i)$. Pode-se descrever como preencher o vetor $BOUNDCOND$ usando o vetor ID no seguinte algoritmo:

para $i = 1, 2, 3, \cdots, nnos$ **faça**
  se $ID[2][i] == 0$ **então**
    $BOUNDCOND[i] \leftarrow g_D(x_i, y_i)$
  fim
  se $ID[2][i] == 1$ **então**
    $BOUNDCOND[i] \leftarrow g_N(x_i, y_i)$
  fim
fim

Tais estruturas preenchidas formam a malha de elementos finitos computacionalmente. Pode-se então construir as matrizes locais do sistema global. Um procedimento para a construção das matrizes locais M^e, D^e, C^e e R^e é apresentado.

Algoritmo 15: Algoritmo para construção das matrizes locais D^e, C^e, R^e, M^e.

Entrada: matrizes $D^e, C^e, R^e, M^e, IEN, COORD$ e o elemento e.
Alocar vetores **x,y,b** e **c** de ordem 3.
(Armazenando as coordenadas dos vértices do elemento Ω_e nos vetores **x** e **y**)
para $nolocal = 1, 2, 3$ **faça**
  $noGlobal \leftarrow IEN[e][noLocal]$; (associação feita entre os nós local e global)
  $\mathbf{x}[noLocal] \leftarrow COORD[noGlobal][1]$; (coordenada x do nó global)
  $\mathbf{y}[noLocal] \leftarrow COORD[noGlobal][2]$; (coordenada y do nó global)
fim
$\mathbf{b}[1] \leftarrow \mathbf{y}[2] - \mathbf{y}[1]$; $\mathbf{b}[2] \leftarrow \mathbf{y}[3] - \mathbf{y}[1]$; $\mathbf{b}[3] \leftarrow \mathbf{y}[1] - \mathbf{y}[2]$;
$\mathbf{c}[1] \leftarrow \mathbf{x}[3] - \mathbf{x}[2]$; $\mathbf{c}[2] \leftarrow \mathbf{x}[1] - \mathbf{x}[3]$; $\mathbf{c}[3] \leftarrow \mathbf{x}[2] - \mathbf{x}[1]$;
$A^e \leftarrow (c[3] \cdot b[2] - c[2] \cdot b[3])/2$
(Construção das matrizes D^e, M^e, C^e e R^e.)
para $i = 1, 2, 3$ **faça**
  para $j = 1, 2, 3$ **faça**
    $D[i][j] \leftarrow (\epsilon/(4 \cdot Area)) \cdot (\mathbf{b}[i] \cdot \mathbf{b}[j] + \mathbf{c}[i] \cdot \mathbf{c}[j])$
    se $i = j$ **então**
      $M[i][j] \leftarrow A^e/6$
      else
        $M[i][j] \leftarrow A^e/12$
      end
    fim
    $R[i][j] \leftarrow \sigma \cdot M[i][j]$
  fim
  $C[1][i] \leftarrow C[2][i] \leftarrow C[3][i] \leftarrow (\beta_x/6) \cdot \mathbf{b}[i] + (\beta_y/6) \cdot \mathbf{c}[i]$
fim

Algoritmo 16: Construção do vetor local F^e.

Entrada: elemento e, vetor F^e, vetores $ID, BOUNDCOND$, matrizes IEN, e $COORD$, função de fonte f. Considere também que, no Algoritmo (1), os vetores **b** e **c** e as matrizes K^e e M^e já tenham sido calculados.

(Construindo parte do vetor F^e que está associado ao termo de fonte.)
para $i = 1, 2, 3$ **faça**
  $F^e[i] = 0;$
  para $j = 1, 2, 3$ **faça**
    $noGlobal = IEN[e][j]; x = COORD[noGlobal][1]; y = COORD[noGlobal][2];$
    $F^e[i] = F^e[i] + M[i][j] * f(x, y;)$
  fim
fim
(Construção da parte do vetor F^e que está associado a condição de contorno de Dirichlet.) **para** *i=1,2,3* **faça**
  $noGlobal = IEN[e][i];$
  se $ID[noGlobaal][1] \neq 0$ *e* $ID[noGlobaal][2] == 1$ **então**
    $F^e[i] = F^e[i] - (BOUNDCOND[IEN[e][1]][1]) * K^e[i][1] +$
    $BOUNDCOND[IEN[e][2]][1]) * K^e[i][2] + BOUNDCOND[IEN[e][3]][1]) *$
    $K^e[i][3]$
  fim
fim
$x = 0$
para $i = 1, 2, 3$ **faça**
  $noGlobal = IEN[e][i];$
  se $noGlobal[e][2] == 2$ **então**
    $x = x + 1$
  fim
fim
se $i = 1, 2, 3$ **então**
  se $x \geqslant 2$ **então**
    $F^e[i] = F^e[i] + H^e[i][1] * G_{N1} + H^e[i][2] * G_{N2} + H^e[i][3] * G_{N3}$
  fim
fim

Algoritmo 17: Construção das matrizes globais K e M e do vetor global F.

Entrada: nel, matriz LM.

M, K, F $\leftarrow$ 0;

para $e = 1, 2, 3. \cdots nel$ **faça**

  (Monta matrizes locais M^e e K^e e vetor local F^e para o elemento Ω_e;)

  para $i = 1, 2, 3$ **faça**

    if $LM[e][i] \neq 0$ **then**

      $F[LM[e][j]] \leftarrow F[LM[e][j]] + F^2[i]$; **para** *j=1,2,3* **faça**

        if $LM[e][j]$ **then**

          M$[LM[e][i]][LM[e][j]] \leftarrow$ **M**$[LM[e][i]][LM[e][j]] + M^e[i][j]$

          K$[LM[e][i]][LM[e][j]] \leftarrow$ **K**$[LM[e][i]][LM[e][j]] + K^e[i][j]$

        end

      fim

    end

  fim

fim

3.3.5 Estrutura de Dados Elemento-Por-Elemento

Como as matrizes provenientes do MEF são esparsas, ou seja, contêm muitos zeros, é usada a estrutura de dados Elemento-Por-Elemento para a implementação computacional, uma vez que essa estrutura é útil no armazenamento desse tipo de matriz. Nessa estrutura, as matrizes são armazenadas em uma estrutura tridimensional formada por todas as matrizes locais associadas aos elementos. No caso, a matriz **M** é armazenada com uma sequência de matrizes locais

$$M^1,\ M^1,\ \cdots\ M^{nel}.$$

Consequentemente, a matriz **K** é

$$K^1,\ ,K^2,\ \cdots\ ,K^{nel}.$$

O algoritmo abaixo apresenta uma maneira de como calcular o produto de uma dessas matrizes por um vetor V qualquer.

Algoritmo 18: Produto matriz-vetor usando estrutura EBE.

Entrada: nel, matriz IEN, matriz ID, matriz **K** tridimensional, vetor resposta V.
U (U = M V)
$U \leftarrow 0$;
para $e = 1, 2, \cdots, nel$ **faça**
  (armazena no vetor **eq** os número das equações associadas aos vértices do elemento e.) **para** $e = 1, 2, 3$ **faça**
    eq$[i] \leftarrow ID[IEN[e][i][1]$;
  fim
  (armazena em **u** os valores de U que estão associados aos vértices do elemento e.)
  para $i = 1, 2, 3$ **faça**
    se ***eq*** $== 0$ **então**
      $u[i] \leftarrow 0$;
    fim
    else
      $u[i] \leftarrow U[eq[i]]$;
    end
  fim
  (armazenar no vetor **v** o produto matriz-vetor correspondente a contribuição da matriz do elemento e.)
  para $i = 1, 2, 3$ **faça**
    $v[i] = 0$
    para $j = 1, 2, 3$ **faça**
      $v[i] \leftarrow v[i] + K[e][i][j] * u[j]$;
    fim
  fim adiciona as contribuições do elemento e no vetor resposta V
  para $i = 1, 2, 3$ **faça**
    se $eq \neq 0$ **então**
      $V[eq[i]] \leftarrow V[eq[i]] + v[i]$;
    fim
  fim
fim

3.4 O Método de Diferenças Finitas de Crank-Nicolson

A discretização temporal será apresentada nesta seção. Para isso, usa-se o método de diferenças finitas de Crank-Nicolson para resolver o sistema de equações diferenciais ordinárias (111)-(112). Esse método possui ordem de convergência quadrática (SMITH, 1985). Seja uma discretização temporal do intervalo [0,T] em m subintervalos de tamanhos iguais, de modo que

$$0 = t_0 < t_1 < t_2 < \cdots < t_m \tag{126}$$

no qual $\triangle\, t = t_k - t_{k-1}, t = 1, \cdots, m$. Portanto, os valores de U no sistema (111)-(112) serão calculados em cada ponto $t_k \in (0, T]$ da discretização (126). Dado $U(t_k)$, $U(t_{k+1})$ é obtido através da solução de um sistema linear ao considerar uma aproximação no ponto médio do intervalo $[t_k, t_{k+1}]$ (diferenças finitas de segunda ordem), ou seja, no ponto $t_{k+\frac{\triangle t}{2}}$.

O cálculo de diferenças finitas de segunda ordem nos permite fazer a discretização de $U(t)$ no ponto $t_{k+\frac{\triangle t}{2}}$ da seguinte maneira

$$U'\left(t_{k+\frac{\triangle t}{2}}\right) \cong \frac{U^{k+1}-U^k}{\triangle t} \tag{127}$$

no qual $U^k \cong U(t_k)$. Em $t_{k+\frac{\triangle t}{2}}$, os valores de $U(t)$ e $F(t)$ são calculados fazendo a média aritmética entre seus valores nos pontos t_k e t_{k+1}, ou seja,

$$U(t_k+\frac{\triangle t}{2}) \cong \frac{U^{k+1}+U^k}{2}, \tag{128}$$

$$F(t_k+\frac{\triangle t}{2}) \cong \frac{F^{k+1}+F^k}{2}, \tag{129}$$

no qual $F^k = F(t_k)$. Ao substituir (127), (128) e (129) em (111)-(112), tem-se

$$\mathbf{M}U\left(t_{k+\frac{\triangle t}{2}}\right) \cong \frac{U^{k+1}-U^k}{\triangle t} + \mathbf{K}\frac{U^{k+1}+U^k}{2} = \frac{F^{k+1}+F^k}{2}$$

resultando no seguinte sistema linear

$$\left(\mathbf{M}+\frac{\triangle t}{2}\mathbf{K}\right)U^{k+1} = \left(\mathbf{M}-\frac{\triangle t}{2}\mathbf{K}\right)U^k + \frac{\triangle t}{2}\left(F^{k+1}+F^k\right) \tag{130}$$

Os valores de F^{k+1} e F^k são conhecidos em todo o domínio. Dado U^k, a solução aproximada U^{k+1} é obtida, no tempo $t_{k+1}, k = 0, 1, 2, \cdots, m-1$, ao resolver o sistema linear (130) através de alguma estratégia iterativa para resolver sistemas de grande porte como os métodos GMRES, LCD e CMRH.

Capítulo 4

Experimentos Numéricos

Neste capítulo são apresentados os resultados dos experimentos numéricos realizados para:

- matrizes genéricas e;
- problema de convecção-difusão transiente.

O objetivo dos testes realizados é comparar o desempenho computacional dos métodos estudados, através do tempo de processamento e da quantidade de iterações.

4.1 Comparação entre os Métodos para Matrizes Genéricas

Nesta seção são apresentados testes similares aos encontrados em (SADOK, 1999), (SADOK; SZYLD, 2012) e (HEYOUNI; SADOK, 2008). As matrizes utilizadas para os testes são não simétricas. A configuração do computador utilizado é processador Intel Core i5-4210U, 1.70 GHz, sistema operacional Windows 64 bits e 8 GB de memória RAM. Os métodos encontram-se implementados em linguagem C, utilizando a biblioteca GSL.

O vetor $\boldsymbol{b}$ do sistema $A\boldsymbol{x} = \boldsymbol{b}$ é calculado conhecendo-se previamente o vetor solução $\boldsymbol{x}^*$. Isso é feito com o intuito de comparar a solução encontrada pelos métodos com a solução exata do sistema. Para os experimentos realizados neste trabalho tem-se solução do sistema $\boldsymbol{x}_{\boldsymbol{i}}^* = i$, solução inicial $\boldsymbol{x_0} = [0, \cdots, 0]^T$, dimensão da matriz $n = 3000$, tolerância para norma residual $tol = 10^{-10}$ e quantidade máxima de reinicializações $l_{max} = 8$. A análise será feita considerando-se a quantidade de iterações necessárias para calcular a solução aproximada $\boldsymbol{x}$, a norma residual $||\boldsymbol{x} - \boldsymbol{x}^*||$, a norma infinito do vetor solução $||\boldsymbol{x} - \boldsymbol{x}^*||_\infty$ e o tempo de processamento. Em cada exemplo será mostrada uma matriz com dimesão (5×5) com o objetivo de facilitar a observação de sua estrutura.

4.1.1 Exemplo 1

Para este exemplo foi usada a matriz de Riemann, presente em (SADOK; SZYLD, 2012):

$$A(i,j) = \begin{cases} i, & \text{se } i+1 \text{ divide } j+1; \\ -1, & \text{caso contrário.} \end{cases}$$

Com dimensão (5×5), a matriz de Riemann tem a seguinte estrutura:

$$A = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 \\ -1 & 2 & -1 & -1 & 2 \\ -1 & -1 & 3 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 5 \end{bmatrix}.$$

Para a execução dos métodos é necessário definir o valor de k_{max}. Esse parâmetro irá determinar a quantidade de vetores que irão compor a base para o espaço de $Krylov$ nos métodos GMRES e CMRH, assim como a quantidade de vetores de direções conjugadas no método LCD. Caso esse valor seja grande, o custo computacional para armazenar esses vetores também será. No entanto, se k_{max} é pequeno, os métodos necessitarão reinicializar diversas vezes aumentando significativamente a quantidade de iterações. Portanto, é necessário ponderar o valor desse parâmetro. Para a matriz em questão, foram testados diversos valores para k_{max} e a quantidade de iterações necessárias para encontrar a solução do sistema para cada um desses valores está presente na Figura 5.

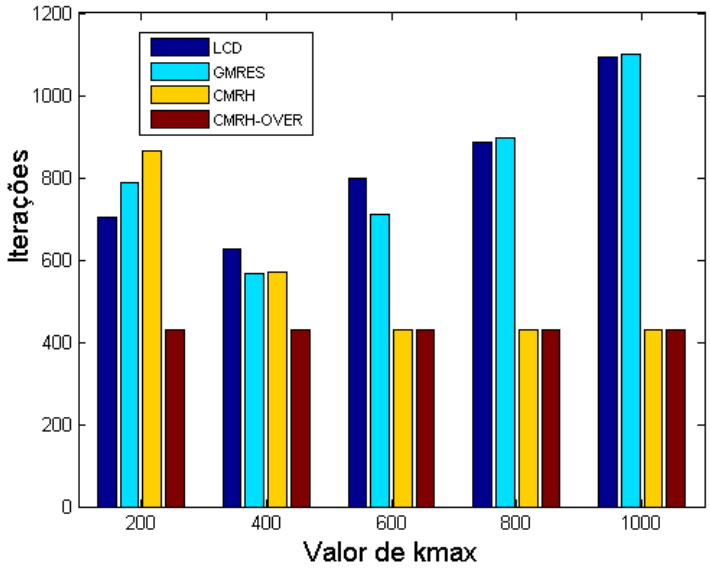

Figura 5 – Quantidade de iterações em função da quantidade de vetores da base.

Como o método CMRH-OVER não reinicializa, a quantidade de iterações em todos os casos é a mesma. Nos testes em que k_{max} assume valores muito pequenos, a solução aproximada

não é encontrada. Como a quantidade de iterações varia para diferentes valores de k_{max}, isso comprova que esse parâmetro influencia diretamente na convergência dos métodos. De acordo com os resultados presentes na Figura 5, neste exemplo será adotado $k_{max} = 400$. O número de condicionamento da matriz em questão é $3.14 \cdot 10^4$. Os resultados sobre a quantidade de iterações, a norma residual, a norma infinito e o tempo de processamento para a execução de cada um dos métodos em estudo estão presentes na Tabela 1.

Tabela 1 – Desempenho Computacional para a Matriz de Riemann.

Método	LCD	GMRES	CMRH	CMRH-OVER
Iterações	625	566	571	430
$\|\|\boldsymbol{x_i} - \boldsymbol{x^*}\|\|$	8.1×10^{-11}	9.5×10^{-11}	5×10^{-11}	1.6×2.2^{-11}
$\|\|\boldsymbol{x_i} - \boldsymbol{x^*}\|\|_\infty$	6.5×10^{-10}	1.76×10^{-10}	4.4×10^{-10}	6.6×10^{-10}
Tempo	$9.74\ s$	$7.65\ s$	$46.58\ s$	$9.10\ s$

O método CMRH-OVER convergiu com menos iterações, porém o LCD levou menos tempo de processamento. Como $k_{max} = 400$, todos os métodos foram reinicializados apenas uma vez. Os valores para norma residual e para norma infinito, em todos os casos, são bem similares. A norma infinito mostra que, no pior caso, a maior diferença entre um elemento da solução exata e da solução aproximada é da ordem de 10^{-10}, portanto a solução encontrada é realmente muito próxima da solução exata. O tempo por iteração do método LCD é $0.0155\ s$, do GMRES é $0.0135\ s$, do CMRH é $0.0815\ s$ e do CMRH-OVER é $0.0211\ s$. Portanto, o método que apresentou o menor custo porm iteração é o LCD. A Figura 6 apresenta a convergência da norma residual para todos métodos.

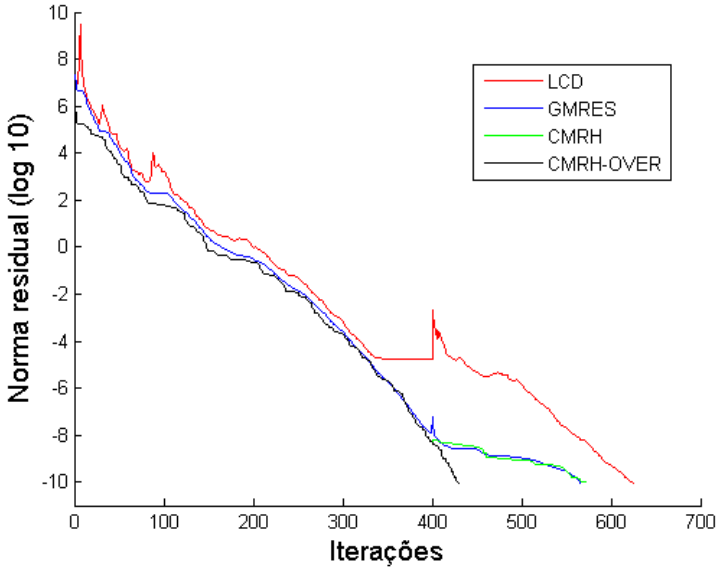

Figura 6 – Convergência da Norma Residual

O método LCD apresenta oscilações no início do processo iterativo, inclusive quando houve reinicialização. Quando os métodos reinicializaram, a norma residual do método LCD era

aproximadamente 10^{-4} enquanto dos demais métodos era próximo de 10^{-8}. A partir dessa nova solução inicial, os métodos levaram cerca de 200 iterações para encontrar a solução do sistema. Isso mostra que a solução inicial próxima da solução do sistema garante que os métodos convergem mais rapidamente.

4.1.2 Exemplo 2

A matriz para este exemplo está presente em (HEYOUNI; SADOK, 2008) e é definida como:

$$A(i,j) = \frac{2\min(i,j) - 1}{n - i + j}.$$

Novamente, para exemplificar uma matriz com dimensão 5×5 tem a seguinte esturutura:

$$A = \begin{bmatrix} 0.2000 & 0.1667 & 0.1429 & 0.1250 & 0.1111 \\ 0.2500 & 0.6000 & 0.5000 & 0.4286 & 0.3750 \\ 0.3333 & 0.7500 & 1.0000 & 0.8333 & 0.7143 \\ 0.5000 & 1.0000 & 1.2500 & 1.4000 & 1.1667 \\ 1.0000 & 1.5000 & 1.6667 & 1.7500 & 1.8000 \end{bmatrix}$$

O número de condicionamento da matriz com dimensão 3000×3000, utilizada para os testes é 8.10^9. Neste exemplo também será adotado $k_{max} = 400$. O desempenho computacional dos métodos é descrito na Tabela 2.

Tabela 2 – Desempenho Computacional.

Método	LCD	GMRES	CMRH	CMRH-OVER
Iterações	526	554	260	260
$\|\|\boldsymbol{x_i} - \boldsymbol{x^*}\|\|$	6.1×10^{-6}	5.9×10^{-6}	7.4×10^{-6}	7.4×10^{-6}
$\|\|\boldsymbol{x_i} - \boldsymbol{x^*}\|\|_\infty$	7.2×10^{-7}	7.8×10^{-7}	8.5×10^{-7}	8.5×10^{-7}
Tempo	$6.90\ s$	$7.03\ s$	$20.69\ s$	$4.16\ s$

Observe que os métodos CMRH e CMRH-OVER apresentaram a mesma quantidade de iterações, contudo o tempo de execução do método CMRH-OVER é quase 80% menor que o método CMRH. Isso já era esperado, conforme discutido na seção 2.2.4, uma vez que o método CMRH executa mais operações que o método CMRH-OVER. Neste exemplo, o método que leva mais iterações para convergir é o GMRES e o que apresenta o maior tempo de processamento é p CMRH. Por iteração, o método LCD leva $0.0131\ s$, o GMRES $0.0126\ s$, o CMRH $0.0795\ s$ enquanto o CMRH-OVER $0.016\ s$. A Figura 7 traz a convergência da norma residual dos métodos.

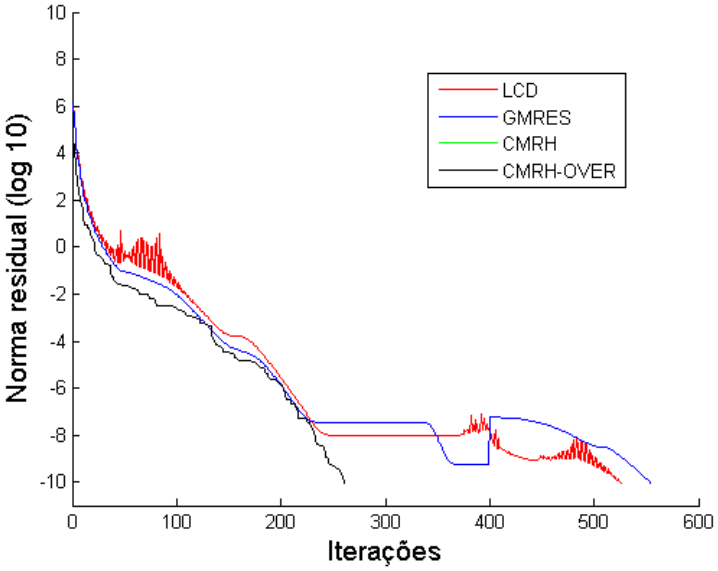

Figura 7 – Convergência da Norma Residual.

Como $k_{max} = 400$ e o método CMRH encontrou a solução do sistema com 260 iterações, não houve reinicialização. Portanto, a convergência da norma residual dos métodos CMRH e CMRH-OVER é a mesma, por isso na Figura 7 uma curva se sobrepõe a outra. Assim como o GMRES, o método LCD apresentou um período de estabilização da norma residual antes do processo de reinicialização, aproximadamente entre 220 e 360 iterações. Novamente, o método CMRH-OVER apresenta melhores resultados que os demais, uma vez que o método CMRH não precisou reinicializar para encontrar a solução do sistema.

4.1.3 Exemplo 3

Neste caso, a matriz foi determinada pelo próprio autor e é definida como:

$$A(i,j) = \begin{cases} 1 + i \cdot \varepsilon, & \text{se } i > j; \\ 1, & \text{caso contrário.} \end{cases}$$

Para os experimentos, tem-se $\varepsilon = 0.1$. Com dimensão 5×5, essa matriz possui a seguinte estrutura:

$$A = \begin{bmatrix} 1 & 1.1 & 1.1 & 1.1 & 1.1 \\ 1 & 1 & 1.2 & 1.2 & 1.2 \\ 1 & 1 & 1 & 1.3 & 1.3 \\ 1 & 1 & 1 & 1 & 1.3 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Com dimensão 3000×3000, o número de condicionamento desta matriz é $3.79 \cdot 10^6$. Devido aos resultados obtidos, adotou-se neste exemplo $k_{max} = 600$. O desempenho computacional encontra-se presente na Tabela 3.

Tabela 3 – Desempenho Computacional.

Método	LCD	GMRES	CMRH	CMRH-OVER
Iterações	4152	4792	3378	1034
$\|\|\boldsymbol{x_i} - \boldsymbol{x^*}\|\|$	9.71×10^{-10}	1.04×10^{-10}	7.85×10^{-10}	1.1×10^{-10}
$\|\|\boldsymbol{x_i} - \boldsymbol{x^*}\|\|_\infty$	9.81×10^{-9}	1.34×10^{-9}	6.69×10^{-11}	1.08×10^{-10}
Tempo	$61.37\ s$	$75.45\ s$	$353.70\ s$	$29.89\ s$

Este é o exemplo em que os métodos levaram mais iterações e mais tempo de processamento para convergirem. O método LCD reinicializou sete vezes, o GMRES seis e o método CMRH cinco. Contudo, o método CMRH-OVER apresentou resultados muito melhores que os demais, levando aproximadamente 39% do tempo de processamento apresentado pelo GMRES e 31% da quantidade de iterações do método CMRH. Em cada iteração, o método LCD leva $0.0147\ s$, o GMREs leva $0.0157\ s$, o CMRH $0.1047\ s$ e o CMRH-OVER 0.0289. A convergência da norma residual encontra-se na Figura 8.

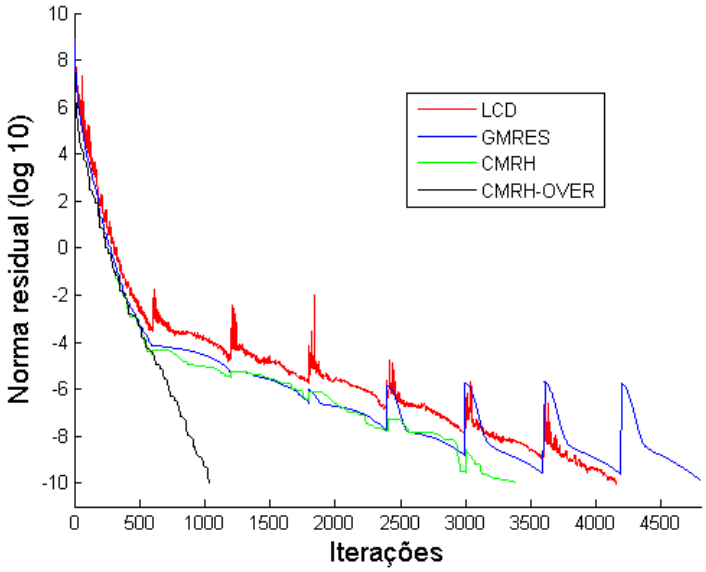

Figura 8 – Convergência da Norma Residual.

Novamente, o método LCD apresenta fortes oscilações, principalmente no momento em que o método usa o processo de reinicialização. A convergência da norma residual dos métodos GMRES e CMRH são similares, conforme diversos casos na literatura. Como esperado, a norma residual e a norma infinito apresentam valores bem similares, devido ao critério de parada dos métodos.

4.1.4 Discussão dos Resultados

Com exceção do primeiro exemplo em que o GMRES apresenta menor tempo de convergência, em todos os casos de matrizes genéricas estudados, o método que converge

mais rápido para a solução do sistema, seja analisando a quantidade de iterações ou o tempo de execução, é o CMRH-OVER. Esse comportamento pôde ser observado em outros experimentos realizados que não estão apresentados neste trabalho. Contudo, como esse método armazena na própria matriz do sistema os vetores que formam a base para o espaço de $Krylov$, ao fim da execução, essa matriz estará destruída. Essa é a desvantagem desse método com relação aos demais.

Como a norma residual é um critério de parada dos métodos, a diferença dos valores encontrados entre cada um deles é mínima. A norma infinito também apresenta valores satisfatórios, evidenciando que a maior diferença calculada entre os elementos da solução aproximada e da solução exata do sistema é pequena. Esses resultados mostram que a solução encontrada pelos métodos realmente é próxima da solução exata. Em todos os exemplos, quando os métodos reinicializam, a solução já está mais próxima da solução exata do problema. Isso corrobora o fato de que, quanto mais próxima da solução exata do sistema é a solução inicial, mais rápido os métodos convergem.

Uma variável importante na execução dos métodos reinicializados é o parâmetro k_{max}. Esse parâmetro determina a quantidade de vetores que irão compor a base para o subespaço de $Krylov$. Caso esse parâmetro seja pequeno, o método pode não convergir ou executar mais iterações quando comparado com um valor adequado. Contudo, se k_{max} tem valor alto, haverá um custo computacional maior para o armazenamento desses vetores. Por isso, baseando-se nos testes realizados, definiu-se para este trabalho diferentes valores para esse parâmetro que, garantindo a convergência, não acarreta em um custo computacional tão alto para efetuar esse armazenamento, isso porque não foi encontrado na literatura nenhum trabalho que determine um valor ideal para k_{max}.

Acreditamos que a diferença significativa na quantidade de iterações entre os métodos CMRH e CMRH-OVER, seja devido ao cálculo de uma norma infinito durante a execução do método, cujo custo é O(n).

Assim, com base nos experimentos realizados, pode-se inferir que, se a matriz do sistema não é significante ao término da execução, o método CMRH-OVER é o mais indicado. Caso contrário, se é necessário manter a matriz do sistema, o método LCD é o menos recomendado e, apesar da convergência dos métodos GMRES e CMRH serem próximas, o tempo de execução do GMRES, salvo algumas exceções, é melhor que o CMRH.

4.2 Equação de Convecção-Difusão Transiente

Para testar o desempenho computacional dos métodos em um problema aplicado, resolveu-se numericamente o sistema presente no Capítulo 2:

$$\partial_t u + \nabla \cdot (-\epsilon \nabla u + u\beta) = f, \ em \ Q; \quad (131)$$

$$u = g_D, \ em \ \Gamma \times (0, T); \quad (132)$$

$$u(x, 0) = 0, \ em \ \Omega. \quad (133)$$

A implementação do problema de Elementos Finitos foi disponibilizada para este trabalho pelo prof. Dr. Isaac Pinheiro dos Santos e sua orientanda Laisa Karoline Muller, fruto do trabalho (MULLER, 2014). Modificações na implementação original foram feitas, como implementar o campo de direções variável. Não foi considerado, neste trabalho, o termo reativo, portanto $\sigma = 0$. Em todos os experimentos tem-se domínio $\Omega : (0, 1) \times (0, 1)$, o coeficiente de difusão $\varepsilon = 10^{-6}$, tolerância residual 10^{-3} e condições de contorno e iniciais homogêneas. A malha utilizada é particionada em 50×50, fazendo com que a quantidade de equações seja $49 \times 49 = 2401$. Como esse é um problema dependente do tempo t, tem-se tempo inicial $t_0 = 0$. Em cada passo do processo iterativo, o tempo é acrescido de 0.0025. Os resultados apresentados são para 200 iterações, ou seja, o tempo final $t_{final} = 200 \cdot 0.0025 = 0.5$. Dentro de cada laço temporal haverá um sistema linear para ser resolvido, resultando em 200 sistemas lineares. A quantidade de iterações apresentada para os exemplos a seguir é o resultado do somatório de todas as iterações executadas pelos métodos em cada uma das 200 iterações dentro do laço temporal. São apresentados quatro exemplos com:

- termo de fonte f e campo de velocidade β constantes;
- termo de fonte f constante e campo de velocidade β variável;
- termo de fonte f variável e campo de velocidade β constante e;
- termo de fonte f e campo de velocidade β variáveis.

Devido à estrutura do método de elementos finitos, que usa a estratégia elemento por elemento, não foi possível testar esse problema usando o método CMRH-OVER. Por isso, os exemplos abaixo trazem a solução encontrada para os métodos LCD, GMRES e CMRH.

4.2.1 Exemplo 1

Neste primeiro exemplo, tem-se o termo de fonte constante $f = 1$ e o vetor de direções constante $\beta = \langle 1, 1/2 \rangle$. O campo de velocidade para este exemplo encontra-se na Figura 9.

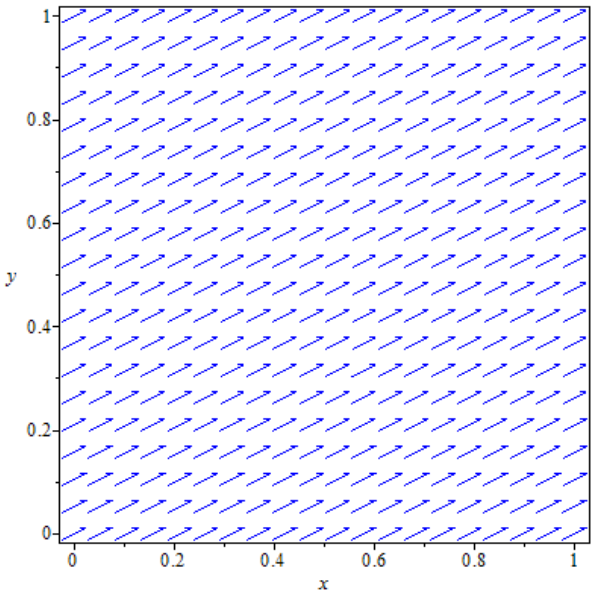

Figura 9 – Campo de Velocidades $\beta =< 1, 1/2 >$.

Como o campo de velocidade é constante, a matriz do sistema não varia com o tempo. A solução encontrada para cada um dos três métodos está presente na Figura 10.

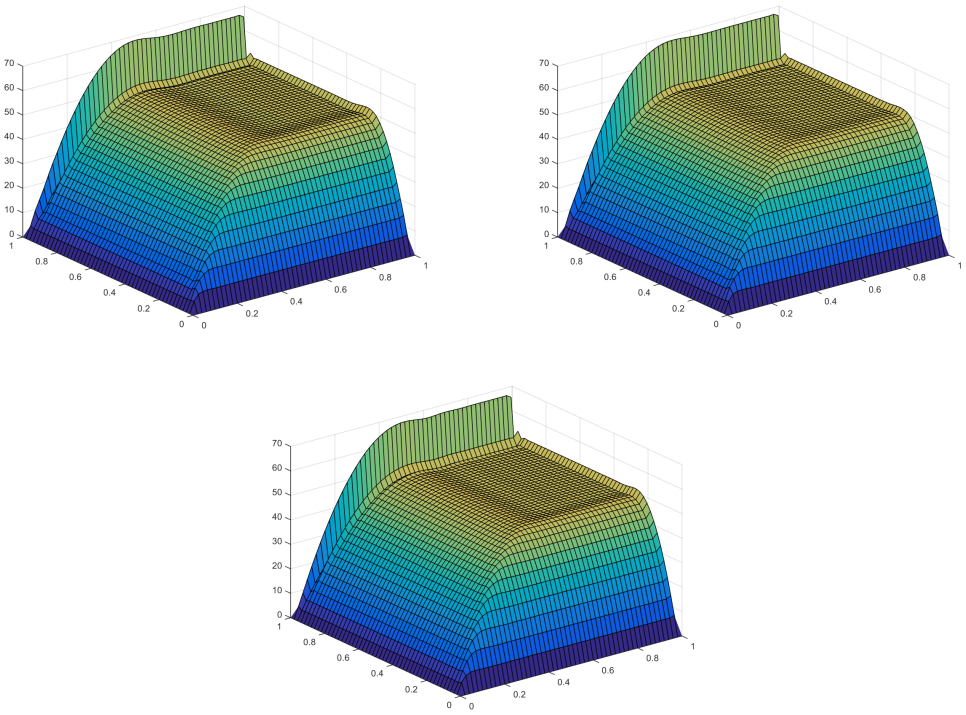

Figura 10 – Solução numérica utilizando os métodos LCD, GRMES e CMRH.

Em todos os casos há fortes oscilações próximas ao eixo $y = 1$. Esse problema não é decorrente da aplicação dos métodos para a resolução de sistemas lineares, mas sim do

Método de Elementos Finitos. O tempo de processamento e a quantidade de iterações na execução de cada um dos métodos está presente na Tabela 4.

Tabela 4 – Desempenho Computacional.

	LCD	GMRES	CMRH
Iterações	875	863	616
Tempo	$16.37\ s$	$18.31\ s$	$14.66\ s$

A quantidade de iterações e o tempo de execução do método CMRH é menor que os demais. Apesar do método GMRES levar menos iterações que o LCD, seu tempo de execução é maior. O tempo médio de execução de cada iteração do método LCD é $0.0187\ s$, do método GMRES $0.0212\ s$ e do método CMRH é $0.0237\ s$. A execução de cada iteração do método CMRH leva mais tempo que nos outros métodos. Porém, esse tempo é compensado pela quantidade menor de iterações. Neste exemplo, o método mais indicado é o CMRH. Os resultados estão de acordo com a literatura.

4.2.2 Exemplo 2

Neste caso, a função de fonte constante $f(x, y) = 1$ e campo de velocidade $\beta(t, x, y) = \langle -e^{10t}(x - 0.5),\ e^{10t}(x - 0.5)\rangle$. Como o campo de velocidade depende exponencialmente do tempo, a medida que t aumenta, o módulo da velocidade também aumenta. Para $t = 0$, o campo de velocidade está presente na Figura 11.

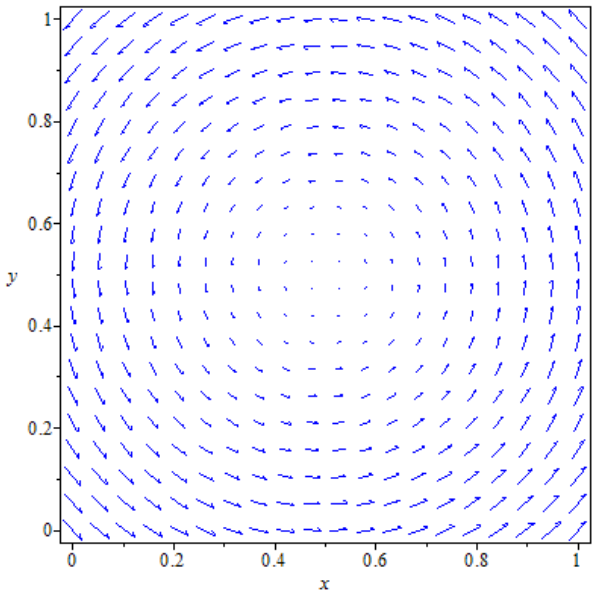

Figura 11 – Solução numérica utilizando os métodos LCD, GRMES e CMRH.

A solução encontrada para a solução do sistema encontra-se na Figura 12.

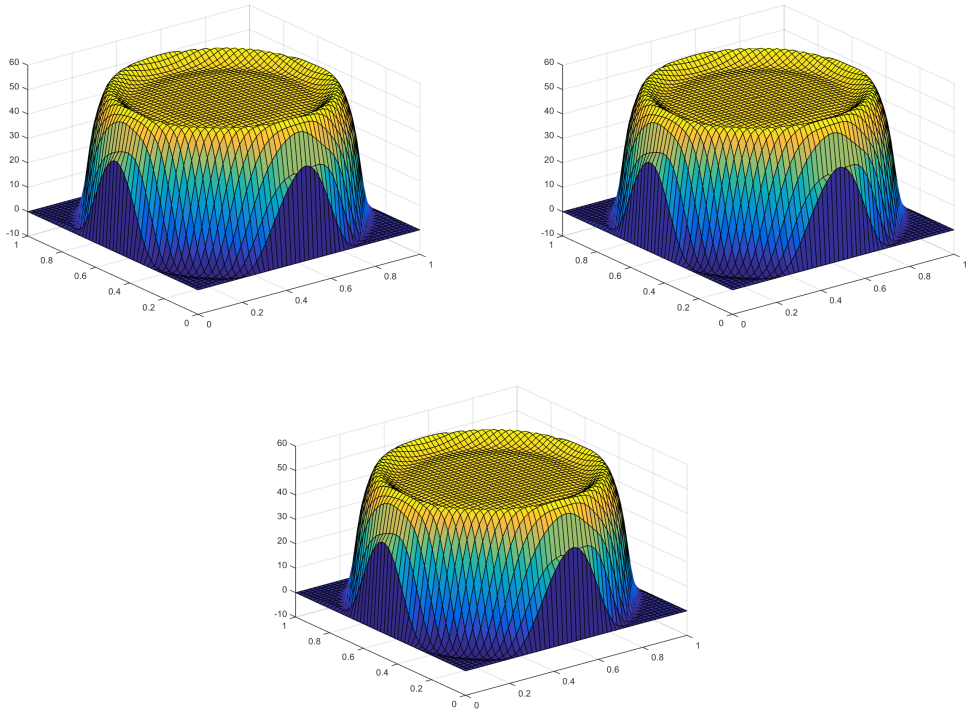

Figura 12 – Solução numérica utilizando os métodos LCD, GRMES e CMRH.

A Tabela 5 mostra o tempo de processamento e a quantidade de iterações para esse experimento.

Tabela 5 – Desempenho Computacional.

	LCD	GMRES	CMRH
Iterações	1526	1413	1144
Tempo	$16.67\ s$	$16.49\ s$	$16.13\ s$

Assim como no exemplo anterior, o método que leva menos iterações é o CMRH. Apesar disso, o tempo de execução é similar aos demais. Isso evidencia que o tempo médio por iteração é maior que os outros métodos. Em média, cada iteração do método LCD levou $0,0109\ s$, do método GMRES $0,0116\ s$ e do método CMRH $0,0140\ s$. Novamente, o método que leva mais tempo por iteração é o CMRH.

4.2.3 Exemplo 3

Neste exemplo, tem-se função de fonte $f(x,y) = e^{-100((x-0.3)^2+(y-0.3)^2)}$ e campo de velocidade $\beta = \langle 1, 1/2 \rangle$. A solução encontrada está na Figura 13.

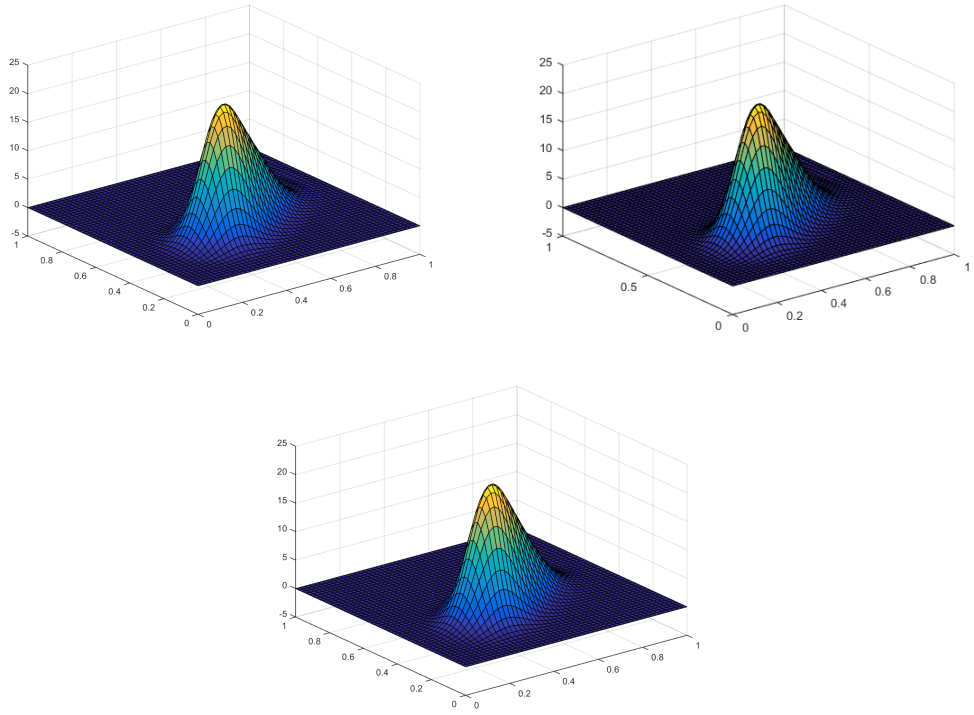

Figura 13 – Solução numérica utilizando os métodos LCD, GRMES e CMRH.

Os resultados encontrados para a quantidade de iterações e para o tempo de processamento está presente na Tabela 6.

Tabela 6 – Desempenho Computacional.

	LCD	GMRES	CMRH
Iterações	651	650	529
Tempo	$21.09\ s$	$18.22\ s$	$17.93\ s$

A quantidade de iterações dos métodos LCD e GMRES é praticamente a mesma, tendo o GMRES tempo de execução melhor. O método CMRH apresentou os melhores resultados, seja para tempo de execução ou quantidade de iterações. Nesse exemplo, o tempo médio por iteração do LCD é $0.032\ s$, do GMRES é $0.028\ s$ e do CMRH é $0.033\ s$. Apesar de levar menos tempo para encontrar a solução, cada iteração do método CMRH leva mais tempo que dos demais métodos.

4.2.4 Exemplo 4

Neste exemplo, tem-se o campo de velocidade dependente do tempo e da posição do elemento $\beta = \langle -e^{10t}(x - 0.5),\ e^{10t}(y - 0.5)\rangle$. A função de fonte também é variável, dependendo da posição $f(x, y) = e^{-100((x-0.3)^2+(y-0.3)^2)}$. A Figura 14 mostra o resultado após 200 iterações no laço temporal.

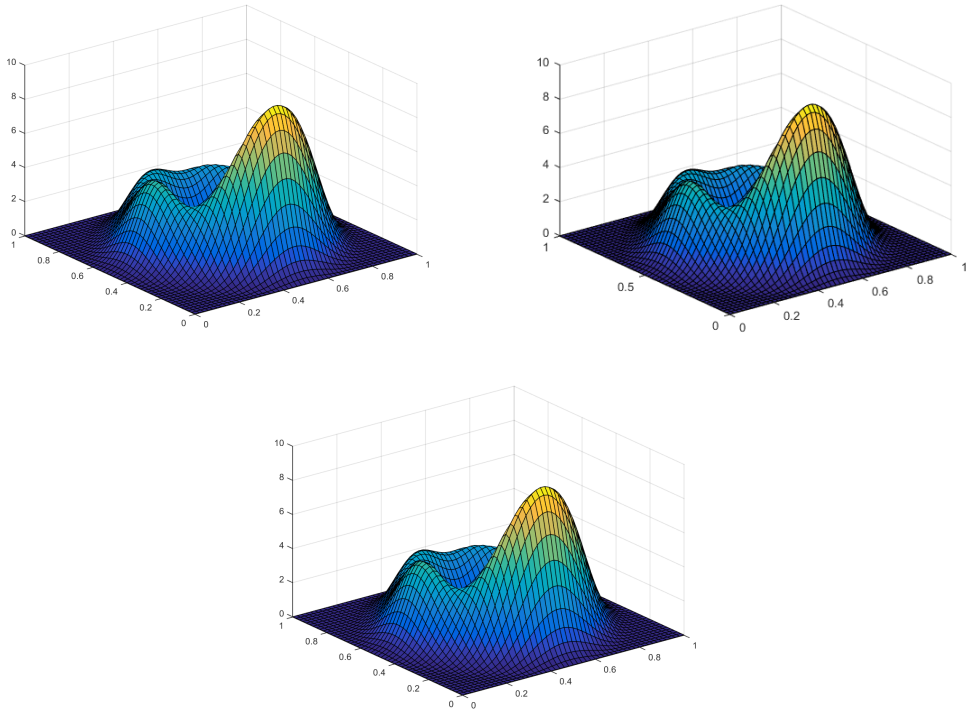

Figura 14 – Solução numérica utilizando os métodos LCD, GRMES e CMRH.

A Tabela 7 apresenta os resultados para a quantidade de iterações e para o tempo de processamento.

Tabela 7 – Desempenho Computacional.

	LCD	GMRES	CMRH
Iterações	1152	1091	1387
Tempo	$20.15\ s$	$19.63\ s$	$35.40\ s$

A quantidade de iterações e o tempo de execução do CMRH é maior que os outros métodos. Por iteração, o método LCD leva $0.0174\ s$, o GMRES $0.0179\ s$ enquanto o CMRH $0.0255\ s$. Novamente, o CMRH é o que leva mais tempo para executar uma iteração. Neste exemplo, o método mais indicado é o GMRES.

4.2.5 Discussão dos Resultados

Nos exemplos que envolvem a equação de convecção-difusão transiente é necessário resolver 200 sistemas lineares, isso porque o laço temporal é executado 200 vezes. Pelo fato de a matriz do sistema ser em banda e esparsa, cada um desses sistemas lineares necessita de poucas iterações para ser resolvido.

Nos exemplos em que o campo de velocidades é variável $\beta = \langle -e^{10t}(x - 0.5),\ e^{10t}(y - 0.5)\rangle$, os métodos levaram mais iterações para convergir. Nestes exemplos, a matriz do sistema

varia dentro do laço temporal, uma vez que o campo de direções é dependente do tempo. Assim, em cada iteração há um sistema com uma matriz diferente para ser resolvido. Isso não ocorre quando o campo de direções é constante $\beta = \langle 1, 1/2 \rangle$. Neste caso, a matriz não varia com o tempo.

Analisando o tempo que cada método leva para executar uma iteração, o CMRH é o mais lento em todos os exemplos. Esse tempo maior é compensado pela quantidade de iterações que, com exceção do Exemplo 14, é menor que todos os outros métodos.

Nesse exemplo aplicado, não é possível comparar a solução obtida pelos métodos com a solução exata, uma vez que não se tem solução exata para o sistema em questão. Para avaliar quão próxima estava as soluções encontradas ao fim de 200 iterações, calculou-se a norma da diferença entre as soluções dos três métodos. O resultado está presente na Tabela 8.

Tabela 8 – Comparação entre a norma da diferença dos métodos.

	LCD e CMRH	LCD e GMRES	GMRES e CMRH
Exemplo 1	14.2394	12.4885	19.1229
Exemplo 2	8.6017	5.5665	6.5563
Exemplo 3	1.1356	4.3715	4.4976
Exemplo 4	1.3470	2.1140	1.0007

Como são 2041 vetores e a tolerância para a norma residual é 10^{-3}, a Tabela 8 mostra que a diferença entre os resultados é pequena, acentuando-se mais no primeiro exemplo.

Capítulo 5

Conclusão

O presente trabalho trata de métodos iterativos para resolução de sistemas lineares de grande porte não simétricos. Três métodos foram estudados: GMRES, CMRH e LCD, todos baseados em espaços de $Krylov$. Uma versão do método CMRH (CMRH-OVER), incluindo processo de sobre-armazenamento, também é abordado.

Enquanto o GMRES usa o método de Arnoldi, através do processo de ortogonalização de Gram-Schmidt, para construir a base para o espaço de $Krylov$, o método CMRH usa o processo de Hessenberg via fatoração LU. Essa é a principal diferença entre esses métodos. O método CMRH-OVER é uma adaptação do CMRH, que armazena na própria matriz A os vetores que formam a base para o espaço de $Krylov$. O método LCD consiste em encontrar um conjunto de vetores de direções conjugadas a esquerda da matriz A, formando uma base para o espaço vetorial, de modo que a solução exata possa ser escrita como uma combinação linear dos vetores dessa base.

Em cada método é definido um processo iterativo $\boldsymbol{x_k} = \boldsymbol{x_0} + \boldsymbol{z_k}$, sendo que $\boldsymbol{z_k}$ é escolhido com o objetivo de minimizar a norma residual. Como a matriz do sistema pode ser grande, armazenar os vetores que formam a base para esses espaços pode ter um custo computacional significativo. Por isso, os métodos utilizam um processo de reincialização, que assumem como solução inicial do sistema a última solução encontrada. Esse processo não ocorre no método CMRH-OVER.

Os métodos foram aplicados em um problema prático, através da solução numérica da equação de convecção-difusão transiente. A resolução dessa equação recai na solução de sistemas lineares, oriundos da discretização por diferenças finitas no tempo e elementos finitos no espaço. A implementação computacional da discretização dessa equação foi apresentada em um capítulo específico.

Primeiramente, os métodos LCD, GMRES, CMRH e CMRH-OVER foram utilizados para resolverem sistemas lineares com matrizes genéricas. Nestes testes, o método CMRH-

OVER apresentou resultados expressivamente melhor que os demais, tanto na quantidade de iterações quanto no tempo de execução. Esse método executa menos produtos matriz-vetor que os outros. Além disso, como ele armazena na própria matriz do sistema os vetores para a base do espaço de $Krylov$, não é necessário criar uma nova matriz para fazer esse armazenamento, resultando em uma redução no espaço de memória utilizado. Isso faz com que o método destrua a matriz do sistema. Portanto, caso não seja necessário preservar a matriz, o melhor método é o CMRH-OVER. Caso contrário, o LCD é o menos indicado. A quantidade de iterações do GMRES e do CMRH é similar, mas na maioria dos casos o tempo do GMRES é melhor. Quando se analisa o tempo por iteração, o método mais lento é o CMRH. Os valores da norma residual e da norma infinito obtidos pelos métodos são aproximados. Isso evidencia que os resultados encontrados são semelhantes e próximos da solução exata do sistema.

Uma característica importante nos métodos iterativos é que, quanto mais próximo da solução exata é a aproximação inicial, menos iterações os métodos levarão para encontrar a solução aproximada para o sistema. Diferentemente dos métodos diretos, o custo computacional varia dependendo da aproximação inicial, uma vez que essa aproximação inicial leva o método a gastar mais ou menos iterações.

No caso dos métodos aplicados para resolverem a equação de convecção-difusão transiente, como o CMRH-OVER não foi testado no problema de Elementos Finitos, devido à sua estrutura, não é possível compará-lo com os outros métodos. Com exceção do exemplo 4.2.4, o método que apresenta os melhores resultados, seja para a quantidade de iterações ou para o tempo de execução é o CMRH. Embora o tempo por iteração desse método seja maior os demais, a diferença do tempo é compensada pela quantidade de iterações que são necessárias para calcular a solução.

5.1 Trabalhos Futuros

Como trabalhos futuros, deseja-se desenvolver as seguintes etapas:

- Buscar uma forma de definir o valor ideal para o parâmetro k_{max}, seja por um estudo teórico ou por testes estatísticos.
- Analisar se existe uma relação entre o número de condicionamento da matriz e o melhor método a ser aplicado.
- Modificar a implementação do método CMRH afim de melhorar seu tempo de execução, verificando em qual parte do loop o método demora mais tempo para executar.

- Avaliar a viabilidade de utilizar o método CMRH-OVER para o problema de Elementos Finitos.
- Estudar a eficiência dos métodos aplicados em sistemas oriundos de outros problemas.
- Buscar outros tipos de resolução de sistemas lineares e comparar com os métodos que encontram-se implementados.
- Estudar e implementar o termo reativo e as condições de contorno de Neumann na equação de convecção-difusão-reação transiente .

Referências

BROOKS, A. N.; HUGHES, T. J. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. **Computer Methods in Applied Mechanics and Engineering**, Elsevier, v. 32, n. 1, p. 199–259, 1982. Citado na página 40.

BURDEN, R. L.; FAIRES, D. J. **Numerical Analysis**. 8. ed. Boston, MA: Cengage, 2011. Citado na página 2.

CATABRIGA, L.; COUTINHO, A. L.; FRANCA, L. P. Evaluating the LCD algorithm for solving linear systems of equations arising from implicit SUPG formulation of compressible flows. **International Journal for Numerical Methods in Engineering**, Wiley Online Library, v. 60, n. 9, p. 1513–1534, 2004. Citado na página 31.

CROCHET, M. J.; DAVIES, A. R.; WALTERS, K. **Numerical Simulation of Non-Newtonian Flow**. New York: Elsevier, 2012. v. 1. Citado 2 vezes nas páginas 2 e 33.

DAI, Y.-H.; YUAN, J. Study on semi-conjugate direction methods for non-symmetric systems. **International journal for numerical methods in engineering**, Chichester, New York, Wiley [etc.] 1969-, v. 60, n. 8, p. 1383–1399, 2004. Citado 4 vezes nas páginas 2, 25, 27 e 30.

EYMARD, R.; GALLOUËT, T.; HERBIN, R. Finite volume methods. **Handbook of numerical analysis**, Elsevier, v. 7, p. 713–1018, 2000. Citado na página 2.

FIGUEIREDO, D. G. de. **Análise de Fourier e Equações Diferenciais Parciais**. [S.l.]: Instituto de Matemática Pura e Aplicada, 2000. Citado na página 39.

FOX, R. W. **Introdução à Mecânica dos Fluidos**. 4. ed. Rio de Janeiro: LTC, 1998. Citado na página 34.

GOCKENBACH, M. S. **Understanding and Implementing the Finite Element Method**. Houghton, Michigan: Siam, 2006. Citado na página 44.

GOLUB, G. H.; LOAN, C. F. V. **Matrix Computations**. Baltimore: JHU Press, 1989. v. 2. Citado 3 vezes nas páginas 5, 10 e 15.

HARARI, I.; HUGHES, T. J. Finite element methods for the helmholtz equation in an exterior domain: Model problems. **Computer Methods in Applied Mechanics and Engineering**, Elsevier, v. 87, n. 1, p. 59–96, 1991. Citado na página 33.

HEYOUNI, M.; SADOK, H. A new implementation of the cmrh method for solving dense linear systems. **Journal of Computational and Applied Mathematics**, Elsevier, v. 213, n. 2, p. 387–399, 2008. Citado 5 vezes nas páginas 2, 14, 22, 58 e 61.

HOFFMAN, K.; KUNZE, R. **Linear Algebra**. [S.l.]: Prentince, 1971. Citado na página 5.

ILINCA, F.; PELLETIER, D. Positivity preservation and adaptive solution for the k- ϵ model of turbulence. **AIAA journal**, v. 36, n. 1, p. 44–50, 1998. Citado 2 vezes nas páginas 2 e 33.

IÓRIO, V. **EDP Um Curso de Graduação**. Rio de Janeiro: IMPA, 2001. v. 2. Citado na página 39.

MULLER, L. K. **Estudo de Métodos de Krylov Para Resolver Sistemas Lineares no Contexto do Método de Elementos Finitos.** 2014. Monografia (Bacharel em Matemática), UFES (Universidade Federal do Espítito Santo), São Mateus, Brazil. Citado 2 vezes nas páginas 33 e 65.

QUARTERONI, A. **Numerical Models for Differential Problems**. New York: Springer, 2010. v. 2. Citado 2 vezes nas páginas 35 e 40.

ROLAND, W. L.; PERUMAL, N.; KANKANHALLI, N. **Fundamentals of the Finite Element Method for Heat and Fluid Flow**. Chichester: John Wiley and Sons Ltd., 2004. Citado na página 42.

SAAD, Y. **Iterative Methods for Sparse Linear Systems**. 2. ed. [S.l.]: Siam, 2003. Citado 2 vezes nas páginas 5 e 12.

SAAD, Y.; SCHULTZ, M. H. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. **SIAM Journal on Scientific and Statistical Computing**, SIAM, v. 7, n. 3, p. 856–869, 1986. Citado 2 vezes nas páginas 2 e 14.

SADOK, H. CMRH: A new method for solving non-symmetric linear systems based on the hessenberg reduction algorithm. **Numerical Algorithms**, Springer, v. 20, n. 4, p. 303–321, 1999. Citado 5 vezes nas páginas 2, 14, 18, 22 e 58.

SADOK, H.; SZYLD, D. B. A new look at cmrh and its relation to gmres. **BIT Numerical Mathematics**, Springer, v. 52, n. 2, p. 485–501, 2012. Citado 3 vezes nas páginas 3, 58 e 59.

SMITH, G. D. **Numerical Solution of Partial Differential Equations: Finite Difference Methods**. Oxford: Oxford University Press, 1985. Citado na página 56.

SOLIN, P. **Partial Differential Equations and the Finite Element Method**. New Jersey: Wiley-Interscience, 2006. v. 1. Citado 2 vezes nas páginas 2 e 33.

SÜLI, E. **Finite Element Methods for Partial Differential Equations**. Oxford: Oxford University Computing Laboratory, 2002. Citado 2 vezes nas páginas vi e 38.

THOMAS, J. W. **Numerical Partial Differential Equations: Finite Difference Methods**. New York: Springer Science & Business Media, 2013. v. 22. Citado na página 2.

WERNER, S. Método de estabilizaçao submalha difusao dinâmica aplicado na simulação de escoamentos miscíveis em meios porosos. **Dissertação de mestrado, Programa de Pós-Graduação em Informática/UFES, Vitória, ES**, 2011. Citado 2 vezes nas páginas 2 e 33.

ZIENKIEWICZ, O. C. et al. **The Finite Element Method**. London: McGraw-hill, 1977. v. 3. Citado na página 2.