ADILSON DA SILVEIRA

MÉTODOS ITERATIVOS HÍBRIDOS PARA RESOLUÇÃO DE SISTEMAS LINEARES

CURITIBA
2003

ADILSON DA SILVEIRA

MÉTODOS ITERATIVOS HÍBRIDOS PARA RESOLUÇÃO SISTEMAS LINEARES

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciências, no Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Área de Concentração em Programação Matemática, Setor de Tecnologia, Departamento de Construção Civil e Setor de Ciências Exatas, Departamento de Matemática da Universidade Federal do Paraná.

Orientador: Prof. Dr. Luis Carlos Matioli.

CURITIBA

2003

TERMO DE APROVAÇÃO

ADILSON DA SILVEIRA

MÉTODOS ITERATIVOS HÍBRIDOS PARA RESOLUÇÃO SISTEMAS LINEARES

Dissertação aprovada como requisito parcial para a obtenção do grau de Mestre em Ciências no Programa de Pós-Graduação em Métodos Numéricos em Engenharia da Universidade Federal do Paraná, pela Comissão:

Orientador:

Prof. Dr. Luis Carlos Matioli
Departamento de Matemática, UFPR

Prof. Dr. Celso Carnieri
Programa de Pós-Graduação em Métodos Numéricos
em Engenharia, UFPR

Prof. Dr. Christian José Quintana Pinedo
Centro Federal de Educação Tecnológica do Paraná,
CEFET - PR.

Curitiba, Dezembro de 2003

A minha filha Roberta
A minha esposa Beatriz

AGRADECIMENTOS

A realização deste trabalho só foi possível devido à grande ajuda que recebi do professor Dr. Luiz Carlos Matioli, durante todo o trabalho de orientação. Sua capacidade, paciência e incentivo não faltaram em momento algum e grande parte do mérito deste trabalho é seu.

Também quero deixar registrado meu agradecimento aos colegas do Departamento de Matemática do Centro Federal de Educação Tecnológica do Paraná - CEFET, de Pato Branco, professores: Fredy M. S. Suarez, Santos R. W. S. Bejarano e Christian J. Q. Pinedo, pelo incentivo e apoio durante os estudos.

Ao professor Yuan Jin Yun que sugeriu os caminhos iniciais para os estudos e pesquisa.

Ao programa de Pós-Graduação em Métodos Numéricos em Engenharia - PPGMNE, da Universidade Federal do Paraná - UFPR, pela oportunidade de realizar este trabalho.

Conteúdo

Resumo

O objetivo deste trabalho será apresentar métodos iterativos híbridos para resolver sistemas de equações lineares $Ax = b$, $x, b \in \mathbb{R}^n$. Para isso apresentaremos alguns conceitos inciais de álgebra linear, descreveremos o método de eliminação de Gauss e os métodos iterativos de Jacobi, Gauss-Seidel e SOR. Em seguida proporemos métodos iterativos híbridos que são baseados em combinar uma iteração do método de eliminação de Gauss com os iterativos de Jacobi, Gauss-Seidel e SOR. Tanto os algoritmos iterativos clássicos quanto os híbridos foram implementados em Matlab e analisados numericamente a partir de testes com matrizes dadas em The Matrix Computational Toolbox for Matlab [8].

Apresentaremos na parte final, conclusões sobre os resultados numéricos dos algoritmos referentes aos métodos propostos comparativamente aos métodos tradicionais, com indicações para estudos futuros relacionados a este trabalho.

Abstratct

In this work we presented hibrid iterative methods for solving linear systems $Ax = b$, $x, b \in \mathbb{R}^n$. For that we will present some initial concepts of linear algebra, we will describe the Gauss elimination method and iterative methods of Jacobi, Gauss-Seidel and SOR. Further we propose hibrids iterative methods that consist in to combine an iteration of the Gauss elimination method with classic iteratives methods of Jacobi. Gauss-Seidel and SOR. As classical iteratives algorithms and the hibrids are coded in Matlab and then numericaly analised to start of test with matrices given in The Matrix Computational Toolbox for Matlab [8].

In the last part we present conclusions about the numerical results to algorithms concerning to propose methods comparatively of the traditional ones, with indication for future studys related to this work.

Capítulo 1

Introdução

Um dos grandes desafios da álgebra linear numérica é a resolução de sistemas lineares. Na maioria dos métodos computacionais a solução de sistemas lineares aparece como um subproblema de um problema mais amplo, como é o caso por exemplo, em otimização, dos métodos de Newton, os métodos da família Newton, assim como muitos outros.

Duas famílias de métodos bastante conhecidos dentro da área de álgebra linear numérica são os métodos diretos e os métodos iterativos. Estes métodos já foram extensivamente estudados. Os métodos diretos são baseados nas operações elementares de matrizes, adição de linhas, multiplicação de uma linha por um escalar, para transformar a matriz original em uma matriz triangular, como é o caso da eliminação de Gauss, ou então decompor a matriz dos coeficientes do sistema em produto de outras matrizes que tem alguma estrutura especial, como é o caso da decomposição LU, em que L é triangular inferior com diagonal unitária e U é triangular superior, e a decomposição de Cholesky (para matrizes simétricas e positiva definidas). Quando o sistema linear considerado tem solução os métodos diretos encontram esta solução em um número finito de passos, a menos de erros de arredondamentos.

Por outro lado, existem os métodos iterativos que, opostamente aos métodos diretos, geram uma seqüência, partindo de um ponto inicial arbitrário, que aproxima a solução do sistema considerado. Os algoritmos gerados por métodos iterativos são simples de entender e fáceis de implementar em linguagens de programação. São indicados para matrizes de dimensões altas e esparsas, pois preservam a esparsidade da matriz, ao passo que os métodos diretos não. Os métodos iterativos clássicos e mais difundidos para resolver sistemas lineares são: o método de Jacobi, o método de Gauss-Seidel e os métodos de aceleração conhecidos na literatura como SOR (sucessive overrelaxation).

Neste trabalho estamos propondo uma classe de métodos iterativos, baseados nos métodos clássicos de Jacobi, Gauss-Seidel e SOR combinados com uma iteração do método de eliminação gaussiana, os quais estaremos chamando de métodos híbridos. O mecanismo de funcionamento destes métodos é bastante simples. Inicialmente é feita uma iteração do método de eliminação de Gauss e então partindo da matriz determinada por esta iteração geraramos os processos iterativos baseados em Jacobi, Gauss-Seidel e SOR. Em termos de sistemas lineares a explicação dos métodos híbridos é a seguinte: Transforma-se o sistema linear original em um outro equivalente, através da eliminação da variável x_1 da segunda até a última equação. E neste novo sistema aplicam-se os métodos iterativos clássicos.

A teoria de convergência para os métodos clássicos pode ser considerada para estes novos métodos híbridos, uma vez que será utilizada a matriz dos coeficientes do sistema linear transformado pela primeira iteração da eliminação de Gauss. Tanto os algoritmos dos métodos sugeridos quanto os clássicos, foram implementados em Matlab. Testes numéricos foram realizados para comparar o desempenho e eficiência dos métodos propostos. Constatamos através dos testes computacionais que os algoritmos propostos são promissores, no sentido que são competitivos com os métodos clássicos, principalmente no que se refere ao tempo de processamento, o que já era esperado de antemão devido ao fato de os métodos híbridos serem aplicados em um sistema cuja matriz de coeficientes tem zeros na primeira coluna, a partir da segunda linha. Estes métodos utilizam-se desta estrutura para economizar tempo, o que de fato foi verificado através dos testes que serão apresentados no capítulo 5.

Como os resultados foram promissores, para a implementação feita em Matlab, deixamos como sugestão outras alternativas de geração de métodos híbridos. Por exemplo. gerar os métodos iterativos após a terceira ou quarta iteração do método de elininação de Gauss. Além disso, estudar o efeito da esparsidade, pois sabe-se que o método da eliminação de Gauss não preserva a esparsidade da matriz, e também dos problemas de condicionamento. Para uma matriz formada somente pela unidade os métodos híbridos não podem ser aplicados pelo fato de zerar os denominadores que aparecem nos métodos iterativos.

A dissertação está dividida em 6 capítulos que são descritos em pormenores nos parágrafos seguintes.

No capítulo 1, discutimos introdutoriamente o trabalho, indicando as principais características do mesmo e sua relevância.

No capítulo 2, apresentamos os fundamentos essenciais de álgebra linear, prin-

cipalmente, os resultados referentes a matrizes com estruturas especiais.

No capítulo 3, descrevemos o método de eliminação de Gauss e alguns métodos baseados na fatoração da matriz dos coeficientes do sistema linear considerado.

A principal contribuição deste trabalho será apresentada no capítulo 4. A partir dos métodos iterativos clássicos de Jacobi, Gauss-Seidel e SOR, para resolver sistemas lineares. propomos métodos híbridos (Jacobi híbrido, Gauss-Seidel híbrido e SOR híbrido), cuja metodologia consiste em combinar uma iteração do método de eliminação de Gauss com os iterativos clássicos citados.

No capítulo 5 realizamos testes computacionais referentes aos métodos propostos, bem como para os algoritmos dos métodos iterativos clássicos. Serão feitas várias simulações de soluções de sistemas lineares, gerados utilizando matrizes de testes dadas em The Matrix Computational Toolbox for Matlab, ver refência [8]. As matrizes deste toolbox estão descritas na tabela 5.1, deste capítulo. Serão feitas duas baterias de testes. sendo que a primeira é para matrizes de dimensão 3 e a segunda para matrizes de dimensão 40. Estas matrizes são, na maioria, mal condicionadas e a condição destas é apresentada junto com as tabelas que descrevem os resultados. Comparamos os métodos aos pares, Jacobi com Jacobi híbrido, Gauss-Seidel com Gauss-Seidel hibrído e SOR com SOR híbrido e depois comparamos os três métodos híbridos entre si para as duas baterias de testes. Finalmente conluímos o capítulo fazendo uma análise dos resultados.

Capítulo 2

Algébra Linear - Conceitos Fundamentais

2.1 Introdução

Inicialmente apresentaremos conceitos básicos relativos à álgebra matricial. A idéia é introduzir um apanhado geral das principais estruturas matriciais que serão empregadas em todo o trabalho. Para leituras mais demoradas e aprofundadas são sugeridas algumas referências na parte final do trabalho, em especial,([4], [7], [11], [15], [16], [17]).

2.2 Vetores e Matrizes

O conjunto $\mathbb{R}^{m\times n}$ denota o espaço vetorial de todas as matrizes reais $m \times n$:

$$A \in \mathbb{R}^{m\times n} \Leftrightarrow A = [a_{ij}] = \begin{pmatrix} a_{11} & . & . & . & a_{1n} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ a_{m1} & . & . & . & a_{mn} \end{pmatrix},$$

onde $a_{ij} \in \mathbb{R}$ representa o elemento da i-ésima linha e j-ésima coluna de A.

Consideremos as operações básicas com matrizes, definidas a seguir:

a) (**Adição**) S: $\mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$.

$$C = A + B, \quad \text{onde} \quad c_{ij} = a_{ij} + b_{ij} \quad i = 1, ..., m \quad j = 1..., n.$$

b) (**Multiplicação por Escalar**) M: $\mathbb{R} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$,

$$\text{onde} \quad c = \alpha A,\ c_{ij} = \alpha a_{ij},\ \alpha \in \mathbb{R},\ i = 1, ..., m,\ j = 1, ..., n.$$

c) (**Multiplicação de Matrizes**) P: $\mathbb{R}^{m\times n} \times \mathbb{R}^{n\times p} \to \mathbb{R}^{m\times p}$,

$$\text{onde} \quad C = AB \quad \text{e} \quad c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj},\ i = 1, ..., m,\ j = 1, ..., n.$$

d) (**Transposição**) T: $\mathbb{R}^{m\times n} \to \mathbb{R}^{n\times m}$,

$$\text{onde} \quad C = A^T \quad \text{e} \quad c_{ij} = a_{ji},\ i = 1, ..., n,\ j = 1, ..., n.$$

As matrizes $n \times n$ são ditas quadradas. A matriz identidade $n \times n$ é denotada por I_n e a sua k-ésima coluna, por $e_k^{(n)}$. Desta forma temos:

$$I_n = \begin{pmatrix} 1 & . & . & . & 0 \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ 0 & . & . & . & 1 \end{pmatrix}, \quad e_k^{(n)} = (0, ..., 0, 1_k, 0, ..., 0).$$

Quando a dimensão estiver clara no contexto, nós podemos escrever somente I e e_k, respectivamente.

Se $A, B \in \mathbb{R}^{n\times n}$, satisfazem $AB = I$, então B é dita a inversa de A e é denotada por A^{-1}. Se A^{-1} existe, então A é dita não singular; caso contrário é dita singular. Também, A^{-T} denota $(A^{-1})^T = (A^T)^{-1}$. Para um estudo mais detalhado, veja ([7], [11]), por exemplo.

Se $A = (a) \in \mathbb{R}^{1\times 1}$, então seu determinante ($D : \mathbb{R}^{n\times n} \to \mathbb{R}$) é dado por $det(A) = a$. Para $A \in \mathbb{R}^{n\times n}$ tem-se

$$det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} det(A_{ij}), \quad i \text{ fixo},$$

onde A_{ij} é uma matriz $(n-1) \times (n-1)$, obtida a partir de A, retirando-se desta, a i-ésima linha e a j-ésima coluna.

As propriedades usuais dos determinantes incluem

(i) $det(AB) = det(A)det(B) \qquad A, B \in I\!R^{n\times n}$

(ii) $det(A^T) = det(A) \qquad A \in I\!R^{n\times n}$

(iii) $det(cA) = c^n det(A) \qquad c \in I\!R,\ A \in I\!R^{n\times n}$

(iv) $det(A) \neq 0 \Leftrightarrow A$ é não singular, $\quad A \in I\!R^{n\times n}$.

Para $I\!R^{n\times 1}$ podemos escrever $I\!R^n$ e para nomear este vetor coluna serão usadas letras minúsculas. Já suas componentes individuais serão nomeadas com símbolos subscritos. Então, se $A \in I\!R^{m\times n}$, $x \in I\!R^n$ e $y = Ax$, tem-se

$$y_i = \sum_{j=1}^{n} a_{ij}x_j, \qquad i = 1, ..., m.$$

O produto externo de $x \in I\!R^m$ e $y \in I\!R^n$ é dado por

$$xy^T = \begin{pmatrix} x_1y_1 & . & . & . & x_1y_n \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ x_my_1 & . & . & . & x_my_n \end{pmatrix}$$

e o produto interno ($se \quad m = n$) é definido por

$$x^Ty = \sum_{i=1}^{n} x_iy_i = y^Tx.$$

Se $A \in I\!R^{m\times n}$, podemos escrever

$$A = [a_1, ..., a_n],$$

onde a_k representa a k-ésima coluna de A. Da mesma forma,

$$A = \begin{pmatrix} r_1^T \\ \vdots \\ r_m^T \end{pmatrix},$$

onde r_k^T é a k-ésima linha de A. Veja também ([10], [11]).

2.3 Independência, Ortogonalidade, Subespaços

Um conjunto de vetores $\{a_1, ..., a_n\}$ em $\mathbb{R}^m$ é linearmente independente se

$$\sum_{j=1}^{n} \alpha_j a_j = 0 \Leftrightarrow \alpha_1 = ... = \alpha_n = 0.$$

Caso contrário, uma combinação não trivial de $a_1, ..., a_n$ é zero e o conjunto de vetores $\{a_1, ..., a_n\}$ é dito linearmente dependente.

Um subespaço do $\mathbb{R}^m$ é um conjunto que também é um espaço vetorial. O conjunto de todas as combinações lineares de $a_1, ..., a_n \in \mathbb{R}^m$ é um subespaço denotado $span\ \{a_1, ..., a_n\}$:

$$span\{a_1, ..., a_n\} = \{\sum_j \beta_j a_j \mid \beta_1, \cdots, \beta_n \in \mathbb{R}\},$$

e se $\{a_1, ..., a_n\}$ é linearmente independente, para $b \in span\{a_1, ..., a_n\}$, b é construído, via combinação linear, de forma única a partir de $a_1, ..., a_n$.

Se S_1, S_2,.....,S_k são subespaços do $\mathbb{R}^n$, então sua soma S é definida por

$$S = \{a_1 + a_2 + ... + a_k \mid a_i \in S_i, \quad i = 1, ..., k\},$$

também é um subespaço. S é dito ser uma soma direta de cada $v \in S$, tendo uma única representação $v = a_1 + ... + a_k$, $a_i \in S$. Neste caso escrevemos $S = S_1 \oplus .... \oplus S_k$. Também a intersecção de uma coleção de subespaços é um subespaço, isto é, $S = S \cap S_2 \cap .... \cap S_k$.

O subconjunto $\{a_{i_1}, ..., a_{i_k}\}$ é um subconjunto maximal linearmente independente de $\{a_1, ..., a_k\}$ se $\{a_{i_1}, ..., a_{i_k}\}$ é linearmente independente e não é parte própria de algum subconjunto linearmente independente de $\{a_1, ..., a_n\}$. Se $\{a_{i_1}, ..., a_{i_k}\}$ é maximal, então

$$span\{a_1, ..., a_n\} = span\{a_{i_1}, ..., a_{i_k}\}$$

e $\{a_{i_1}, ..., a_{i_k}\}$ é uma base para $span\{a_1, ..., a_n\}$.

Se $S \subset \mathbb{R}^m$ é um subespaço, então existem vetores independentes básicos $a_1, ...a_k$ em S tal que $S = span\{a_1, ..., a_k\}$. Todas as bases para um subespaço S tem o mesmo número de elementos. Este número é a dimensão de S e é denotado por $dim(S)$. Há dois importantes subespaços associados a uma matriz $A \in \mathbb{R}^{m \times n}$. O espaço linha de A é definido por

$$\mathcal{R}(A) = \{y \in I\!R^m \mid y = Ax, \text{ para algum } x \in I\!R^n\},$$

e o espaço nulo de A, definido por

$$\mathcal{N}(A) = \{x \in I\!R^n \mid Ax = 0\}.$$

O posto de A é definido por

$$posto(A) = dim[\mathcal{R}(A)].$$

Pode-se mostrar que $posto(A) = posto(A^T)$, veja [7] e também, o posto de uma matriz é igual ao número máximo de linhas ou colunas independentes. Para a matriz $A \in I\!R^{m\times n}$, tem-se

$$dim[\mathcal{N}(A)] + posto(A) = n.$$

Se $m = n$, então as seguintes implicações são logicamente equivalentes:

1) A é não singular

2) $\mathcal{N}(A) = \{0\}$

3) $posto(A) = n$.

Um conjunto de vetores $\{x_1, ..., x_p\}$ em $I\!R^m$ é ortogonal se $x_i^T x_j = 0$, sempre que $i \neq j$ e ortonormal se $x_i^T x_j = \delta_{ij}$, onde $\delta_{ij} = \begin{cases} 1, & se \quad i = j \\ 0, & se \quad i \neq j \end{cases}$.

Mais geralmente, uma coleção de subespaços $S_1, ..., S_p$ do $I\!R^m$ é mutualmente ortogonal se $x^T y = 0$, sempre que $x \in S_i$ e $y \in S_j$, $\forall i \neq j$. O complemento ortogonal de um subespaço $S \subset I\!R^m$ é definido por

$$S^{\perp} = \{y \in I\!R^m \mid y^T x = 0, \quad \forall x \in S\}.$$

Também pode-se mostrar que $\mathcal{R}(A)^{\perp} = \mathcal{N}(A^T)$, [11]. Os vetores $v_1, ..., v_k$ formam uma base ortonormal para um subespaço $S \subset I\!R^m$ se eles são ortonormais e geram S. É sempre possível se estender uma base para uma base ortonormal completa $v_1, ..., v_m$ do $I\!R^m$. Note que neste caso, $S^{\perp} = span\{v_{k+1}, ..., v_m\}$.

2.4 Matrizes Especiais - I

Matrizes com formas especiais de elementos nulos, com simetria especial e com outras propriedades são listadas a seguir. Podemos encontrar maiores detalhes, bem como exemplos elucidativos de tais matrizes, principalmente em ([3], [4], [7], [10], [11], [12], [13], [15], [17]).

Matrizes Diagonais e Triangulares

Uma matriz $A_{n\times n}$ é dita triangular superior se $a_{ij} = 0$, sempre que $i > j$; ela é triangular inferior se $a_{ij} = 0$ sempre que $i < j$. Além disso, A é simplesmente triangular se for triangular superior ou inferior.

Uma matriz $A_{n\times n}$ é diagonal se $a_{ij} = 0$ sempre que $i \neq j$. Uma matriz diagonal é, ao mesmo tempo, triangular superior e inferior.

Matrizes Tridiagonais

Em muitas aplicações a matriz associada é esparsa, isto é, a quantidade de elementos não nulos é pequena, quando comparada com o número total de elementos da matriz. Se, além de esparsa, a matriz tem os elementos não nulos concentrados em torno da diagonal, ela é chamada matriz de banda. Podemos formalizar a definição escrevendo que $A \in \mathbb{R}^{n\times n}$ é uma matriz de banda $p + q + 1$, se $a_{ij} = 0$ para $i > j + q$ ou $i < j - p$. Agora, se $p = q = 1$, os elementos não nulos estão nas três diagonais centrais e a matriz é dita tridiagonal. Algumas implementações envolvendo matrizes tridiagonais são apresentadas em [3].

Matrizes Ortogonais

Uma matriz $Q_{n\times n}$ é dita ortogonal se seus vetores coluna formam um conjunto ortonormal, ou seja, $q_i^T q_j = \delta_{ij}$.

Também, sabe-se que uma matriz $Q_{n\times n}$ é ortogonal se e somente se $Q^T Q = I$. Sobre este detalhe pode-se consultar [7].

Para exemplificar consideremos qualquer θ fixo, então a matriz

$$Q = \begin{pmatrix} \cos\theta & -sen\theta \\ sen\theta & \cos\theta \end{pmatrix} \text{ é ortogonal e } Q^{-1} = Q^T = \begin{pmatrix} \cos\theta & sen\theta \\ -sen\theta & \cos\theta \end{pmatrix}$$

Matrizes Simétricas

Uma matriz $A_{n\times n}$ é dita simétrica se $a_{ij} = a_{ji}$, $\forall i, j = 1, ..., n$. $(A = A^T)$

Matrizes Anti-Simétricas

Uma matriz $A_{n\times n}$ é dita anti-simétrica se $a_{ij} = -a_{ji}$, $\forall i, j = 1, ..., n$.

Uma matriz simétrica real A é dita positiva definida se $x^T Ax > 0$, $\forall x \in I\!R^n$, $x \neq 0$. Discussões mais detalhadas podem ser encontradas em [10].

2.5 Normas e Raio Espectral

Para estudarmos os aspectos básicos da análise iterativa matricial, precisamos de alguns conceitos, tais como: norma de vetor, norma de matriz e suas extensões, além do conceito de raio espectral.

2.5.1 Norma de Vetor

Podemos pensar, de certo modo, a norma como sendo uma extensão do comprimento de vetor em $I\!R^2$ ou do valor absoluto, em $I\!R$, veja ([1], [17]).

É bem conhecido que $\forall x \in I\!R$, $|x| = \sqrt{x^2}$ e tem-se as seguintes propriedades:

1. $|x| \geq 0$, $\forall x$, e $|x| = 0 \Leftrightarrow x = 0$,
2. $|\alpha x| = |\alpha||x|$,
3. $|x + y| \leq |x| + |y|$.

Pode-se generalizar as três propriedades anteriores para o espaço vetorial $I\!R^n$, como a seguir.

Definição 2.1 $\|.\| : \mathbb{R}^n \to \mathbb{R}$ *é uma norma de vetor sobre o* $\mathbb{R}^n$ *se:*

1. $\|x\| \geq 0$, $\forall x$, *e* $\|x\| = 0 \Leftrightarrow x = 0$,
2. $\|(\alpha x)\| = |\alpha|\|x\|$
3. $\|(x + y)\| \leq \|x\| + \|y\|$

Exemplo 2.1 *Existem três normas muito utilizadas sobre o* $\mathbb{R}^n$, *definidas a seguir,*

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} = \sqrt{x^T x},$$

e

$$\|x\|_\infty = max\{|x_i| : i = 1, 2, ..., n\}.$$

A norma-2 é a generalização do comprimento Euclidiano de vetor conhecido, em $\mathbb{R}^2$ e $\mathbb{R}^3$ e é chamada de norma Euclidiana ([7], [17]). A norma-∞ é, às vezes, chamada de norma máximo ou norma de Chebyshev. Na verdade estas três normas são casos especiais da norma-p, assim definida

$$\|x\|_p = \{\sum_{i=1}^{n} |x_i|^p\}^{1/p}, \qquad p \geq 1. \tag{2.1}$$

Um resultado clássico relativo à norma-p é a desigualdade de *Hölder*,

$$|x^T y| \leq \|x\|_p \|y\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1.$$

Um caso muito especial desta é a desigualdade de *Cauchy-Schwartz*:

$$|x^T y| \leq \|x\|_2 \|y\|_2.$$

Note que a norma-2 é invariante sob uma transformação ortogonal, pois se $Q^T Q = I$, então

$$\|Qx\|_2^2 = x^T Q^T Q x = x^T x = \|x\|_2^2.$$

Todas as normas sobre o $\mathbb{R}^n$ são equivalentes, isto é, se $\|.\|_\alpha$ e $\|.\|_\beta$ são normas sobre o $\mathbb{R}^n$, existem $c_1, c_2 \in \mathbb{R}$, tais que

$$c_1\|x\|_\alpha \leq \|x\|_\beta \leq c_2\|x\|_\alpha, \quad \forall x \in \mathbb{R}^n.$$

Exemplo 2.2 *Para qualquer* $x \in \mathbb{R}^n$ *tem-se:*

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$$

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$$

$$\|x\|_\infty \leq \|x\|_1 \leq \sqrt{n}\|x\|_\infty.$$

As normas servem para alguns propósitos relacionados à conceituação de distância. Mais precisamente, o $\mathbb{R}^n$ juntamente com a norma sobre o $\mathbb{R}^n$ definem um espaço métrico [9]. Além disso, nós temos a notação familiar de vizinhança, conjuntos abertos, convergência e continuidade, quando trabalhamos com vetores e funções de valores vetoriais.

Suponha que $\widehat{x} \in \mathbb{R}^n$ seja uma aproximação para $x \in \mathbb{R}^n$. $x \neq 0$. Para uma dada norma de vetor, $\|.\|$, define-se o erro absoluto como

$$\epsilon_a = \|\widehat{x} - x\|,$$

e o erro relativo como

$$\epsilon_r = \frac{\|\widehat{x} - x\|}{\|x\|}. \tag{2.2}$$

O erro relativo na norma $-\infty$ pode ser traduzido em uma informação sobre o número correto de dígitos significativos em x. Em particular, se

$$\frac{\|\widehat{x} - x\|_\infty}{\|x\|_\infty} \cong 10^{-p}$$

então a maior componente de $\widehat{x}$ terá aproximadamente $p \in \mathbb{N}$ dígitos significativos corretos. Uma discussão mais detalhada pode ser encontrada em ([3], [14]).

É de nosso interesse saber se a seqüência gerada pelo método iterativo que vamos utilizar converge para a solução, quando nós estudamos tais métodos. Em razão disso, daremos a seguir, alguns conceitos sobre limites de seqüências em espaços vetoriais.

Definição 2.2 *Seja* $\{x^{(k)}\}$ *uma seqüência de n vetores e seja* $x \in \mathbb{R}^n$. *Então* x *é um limite da seqüência* $\{x^{(k)}\}$, *denotado* $x = \lim_{k\to\infty} x^{(k)}$, *se*

$$\lim_{k\to\infty} x_i^{(k)} = x_i, \quad (i = 1, 2, ..., n),$$

onde x_i *são componentes de* x.

Assim, por definição,

$$x = \lim_{k\to\infty} x^{(k)} \Leftrightarrow \lim_{k\to\infty} \|x - x^{(k)}\|_\infty = 0.$$

Além disso, tem-se a partir da equivalência de normas que

$$x = \lim_{k\to\infty} x^{(k)} \Leftrightarrow \lim_{k\to\infty} \mu(x - x^{(k)}) = 0,$$

onde μ é uma norma sobre o $\mathbb{R}^n$. Para maiores detalhes ver ([1], [4], [17]).

2.6 Norma Matricial

Seu emprego é útil, por exemplo, nos casos em que se deseja determinar a sensibilidade do sistema linear $Ax = b$. Inicialmente consideremos a seguinte definição. Apresentamos a seguir coneitos e resultados que serão utilizados na seqüência do trabalho e são baseados nas referências ([7], [15], [17]).

Definição 2.3 $\|.\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ *é uma norma se satisfizer as seguintes propriedades:*

1. $\|A\| \geq 0$, $\forall A \in \mathbb{R}^{m\times n}$, *e* $\|A\| = 0 \Leftrightarrow A = 0$,
2. $\|\alpha A\| = |\alpha|\|A\|$, $A \in \mathbb{R}^{m\times n}$, $\alpha \in \mathbb{R}$.
3. $\|(A + B)\| \leq \|A\| + \|B\|$, $A, B \in \mathbb{R}^{m\times n}$.

Exemplo 2.3 *Seja* $A = (a_{ij})$. *Suponha que* $\|A\| = max_{i,j}|a_{ij}|$. *Então,* $\|A\| \geq 0$ *e* $\|A\| = 0$ *se e somente se* $A = 0$. *Também* $\|\alpha A\| = |\alpha|\|A\|$. *Como* $|a_{ij}+b_{ij}| \leq |a_{ij}|+|b_{ij}|$, $\|(A + B)\| \leq \|A\| + \|B\|$. *Portanto,* $\|.\|$ *apresentada é uma norma matricial.*

A norma matricial mais freqüentemente usada em análise numérica é a norma de Frobenius ou norma-F,

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2} \tag{2.3}$$

e a norma-p

$$\|A\|_p = sup_{x\neq 0}\frac{\|Ax\|_p}{\|x\|_p}. \tag{2.4}$$

A norma-p é também conhecida como norma espectral quando $p = 2$. Então a desigualdade

$$\|AB\|_p \leq \|A\|_p\|B\|_p, \quad A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times q} \tag{2.5}$$

fornece uma informação sobre a relação entre três diferentes normas. Mas nem todas as normas cumprem a desigualdade em (2.5).

Exemplo 2.4 *Defina a norma* $||A||$ *como*

$$||A|| = max\{|a_{ij}| : i = 1, 2, ..., m; j = 1, 2, ... n\},\ e\ A = B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix},$$

então $||A||||B|| = 1 < 2 = ||AB||$.

Isto nos motiva a introduzir a seguinte definição.

Definição 2.4 *Dizemos que as normas* μ *em* $\mathbb{R}^{l\times m}$, ν *em* $\mathbb{R}^{m\times n}$ *e* ρ *em* $\mathbb{R}^{l\times n}$ *são mutualmente compatíveis, se para todo* $A \in \mathbb{R}^{l\times m}$, $B \in \mathbb{R}^{m\times n}$,

$$\rho(AB) \leq \mu(A)\nu(B)\ \textit{é mantida.}$$

Uma norma matricial sobre o $\mathbb{R}^{n\times n}$ *é compatível se ela é compatível com si mesma.*

A partir desta definição podemos considerar o seguinte resultado, relativo à norma-F.

Teorema 2.1 *A norma de Frobenius* $||.||_F$ *é compatível.*

Prova: Para a prova, temos que considerar $||A||_F^2 = \sum ||A_i||_2^2$, onde A_i é a i-ésima coluna de A. Desde que $C = AB = (AB_1, ..., AB_n)$, o que nós precisamos mostrar é que $||AB_i||_2 \leq ||A||_F||B_i||_2$ onde B_i é a i-ésima coluna de B.
Primeiro mostraremos que

$$||Ax||_2 \leq ||A||_F||x||_2, \tag{2.6}$$

$\forall A \in \mathbb{R}^{m\times n}$ e $x \in \mathbb{R}^m$. Seja A particionada por linhas: $A^T = (a_1, ..., a_l)$. Então

$$Ax = \begin{pmatrix} a_1^T x \\ . \\ . \\ . \\ a_l^T x \end{pmatrix}.$$

Portanto

$$||Ax||_2^2 = \sum_{i=1}^{l} |a_i^T x|^2.$$

Segue da desigualdade de *Cauchy*, $|a_i^T x| \leq ||a_i||_2||x||_2$, que

$$\|Ax\|_2^2 \leq \|x\|_2^2 \sum_{i=1}^{l} \|a_i\|_2^2.$$

Então (1.5) é satisfeita pois $\|Ax\|_F^2 = \sum_{i=1}^{l} \|a_i\|_2^2$. Agora seja $C = AB$, onde $A \in \mathbb{R}^{l \times m}$ e $B \in \mathbb{R}^{m \times n}$. Se $B = (b_1, ..., b_n)$ é particionada por colunas, então

$$\|C\|_F^2 = \|AB\|_F^2 = \|(Ab_1, ..., Ab_n)\|_F^2 = \sum_{j=1}^{n} \|Ab_j\|_2^2 \leq$$

$$\leq \|A\|_F^2 \sum_{j=1}^{n} \|b_j\|_2^2 = \|A\|_F^2 \|B\|_F^2.$$

∎

Para a norma-p nós temos a seguinte propriedade

$$\|Ax\|_p \leq \|A\|_p \|x\|_p,$$

onde $A \in \mathbb{R}^{m \times n}$ e $x \in \mathbb{R}^n$.

Similar ao caso de vetores, há algumas definições para seqüências de matrizes $\{A^k\}$, extraídas de ([1], [11]).

Definição 2.5 *Seja $\{A^k\}$ uma seqüência de matrizes $m \times n$ e seja $A \in \mathbb{R}^{m \times n}$. Então A é um limite da seqüência $\{A^k\}$ (escreve-se $A = \lim_{k \to \infty} A^{(k)}$) se*

$$\lim_{k \to \infty} a_{ij}^k = A_{ij}, \quad (i = 1, 2, ..., m; j = 1, 2, ..., n).$$

Por definição tem-se,

$$A = \lim_{k \to \infty} A^{(k)} \Leftrightarrow \lim_{k \to \infty} \|A - A^{(k)}\| = 0.$$

Proposição 2.6.1 *Seja $\lim_{k \to \infty} A_k = A$ e $\lim_{k \to \infty} B_k = B$. Então*

1. $\lim_{k \to \infty} (A_k + A_k) = A + B$
2. $\lim_{k \to \infty} A_k B_k = A.B$
3. *Se A é não singular, então para todo k suficientemente grande, A_k é não singular e*

$$\lim_{k \to \infty} A_k^{-1} = A^{-1}.$$

Prova: Ver [16].

∎

Teorema 2.2 *Se* $\|P\| < 1$, *então* $I - P$ *é não singular e*

$$\|(I-P)^{-1}\| \leq (1-\|P\|)^{-1} \tag{2.7}$$

e

$$\|I-(I-P)^{-1}\| \leq \frac{\|P\|}{1-\|P\|}. \tag{2.8}$$

Prova: Primeiro provaremos que $(I-P)$ é não singular. O caminho é provar que $(I-P)x$ é não nulo $\forall x \neq 0$.

Seja $x \neq 0$. Então,

$$\|(I-P)x\| = \|x - Px\| \geq \|x\| - \|Px\| \geq$$

$$\|x\| - \|Px\|\|x\| \geq (1-\|P\|)\|x\| > 0,$$

como $1-\|P\| > 0$ e $\|x\| > 0$, então $(I-P)x \neq 0$, $\forall x \neq 0$, e $I-P$ é não singular. Na seqüência mostramos que a inequação (2.8) segue de

$$(I-P)^{-1}(I-P) = I,$$

e que

$$(I-P)^{-1} = I + P(I-P)^{-1}$$

ou

$$I-(I-P)^{-1} = -P(I-P)^{-1}.$$

Então,

$$\|I-(I-P)^{-1}\| \leq \|I\| + \|P\|\|(I-P)^{-1}\|,$$

isto é,

$$\|(I-P)^{-1}\| \leq (1-\|P\|)^{-1},$$

pois $\|I\| = 1$, e

$$\|I-(I-P)^{-1}\| \leq \|P\|(1-\|P\|)^{-1}.$$

∎

O resultado a seguir é importante quando se quer estabelecer o quão próximo uma matriz está da singularidade.

Teorema 2.3 *Seja A não singular e seja $\|A^{-1}E\| < 1$. Então $A+E$ é não singular e*

$$(A+E)^{-1} = (I+F)A^{-1} \tag{2.9}$$

onde

$$\|F\| \leq \frac{\|A^{-1}E\|}{1-\|A^{-1}E\|}. \tag{2.10}$$

Além disso,

$$\frac{\|A^{-1}-(A+E)^{-1}\|}{\|A^{-1}\|} \leq \frac{\|A^{-1}\|}{1-\|A^{-1}E\|}. \tag{2.11}$$

Se $\|A^{-1}\|\|E\| < 1$, então

$$\|F\| \leq \frac{Cond(A)\frac{\|E\|}{\|A\|}}{1-Cond(A)\frac{\|E\|}{\|A\|}},$$

e

$$\frac{\|A^{-1}-(A+E)^{-1}\|}{\|A^{-1}\|} \leq \frac{Cond(A)}{\|A\|-Con(A)\|E\|},$$

onde $Cond(A) = \|A\|^{-1}\|A\|$ é o número de condição da matriz A.

Prova: Ver [1]. ∎

2.7 Autovalores e Autovetores

Vamos discutir a equação $Ax = \lambda x$. Essa equação ocorre em muitas aplicações da álgebra linear. Se a equação tiver uma solução não trivial x, dizemos que λ é um autovalor de A e que x é um autovetor associado a λ. Uma das principais aplicações de autovalores é a solução de sistemas de equações diferenciais lineares ([2], [4], [10]). Se A é uma matriz $n \times n$, então A representa uma transformação linear de $\mathbb{R}^n$ em si mesmo. Os autovalores e autovetores nos proporcionam a chave para entender como o operador funciona. Por exemplo, se $\lambda > 0$, o efeito do operador em qualquer autovetor associado a λ é simplesmente o de esticar ou encolher por um fator constante. De fato, o efeito do operador é determinado facilmente em qualquer combinação linear de vetores. Em particular, se for possível encontrar uma base de autovetores para o $\mathbb{R}^n$, o operador pode ser representado por uma matriz diagonal D, formada a partir dos autovalores. Em relação a essa base a matriz A pode ser fatorada em um produto PDP^{-1}, onde P tem colunas formadas pelos autovetores de A.

Para matrizes com elementos complexos, estaremos interessados em matrizes cujos autovetores formam uma base para $\mathcal{C}^n$ (o espaço vetorial de todas as n-uplas de números complexos). Consideraremos também as matrizes simétricas positivas definidas, cujos autovalores são sempre reais e positivos.

Daremos atenção ao problema de encontrar um escalar λ tal que o sistema $n \times n$

$$Ax = \lambda x$$

tenha uma solução não trivial.

Definição 2.6 *Seja A uma matriz $n \times n$. Um escalar λ é um autovalor ou valor característico de A se existe um vetor não nulo x tal que $Ax = \lambda x$. O vetor x é um autovetor ou vetor característico associado a λ.*

A equação $Ax = \lambda x$ pode ser colocada na forma

$$(A - \lambda I)x = 0. \tag{2.12}$$

Logo, λ é um autovalor de A se e somente se a equação (2.12) tem solução não trivial. O conjunto das soluções de (2.12) é $\mathcal{N}(A - \lambda I)$, que é um subespaço do $\mathbb{R}^n$. Logo, se λ é um autovalor de A, então $\mathcal{N}(A - \lambda I) \neq \{0\}$ e qualquer vetor não nulo em $\mathcal{N}(A - \lambda I)$ é um autovetor associado a λ. O subespaço $\mathcal{N}(A - \lambda I)$ é chamado de auto-espaço associado a λ.

A equação (2.12) terá uma solução não trivial se e somente se $(A - \lambda I)$ é singular ou, equivalentemente,

$$\det(A - \lambda I) = 0. \tag{2.13}$$

Expandindo-se o determinante em (2.13), obtemos um polinômio de grau n na variável λ,

$$p(\lambda) = \det(A - \lambda I).$$

Esse polinômio é chamado de polinômio característico e a equação (2.13) é a equação característica para a matriz A. Se contarmos as raízes de acordo com sua multiplicidade, o polinômio característico tem exatamente n raízes. Então, A tem exatamente n autovalores, alguns dos quais podem estar repetidos e alguns podem ser números complexos. Neste último caso, vai ser necessário aumentar nosso corpo de escalares para

os números complexos e permitir que nossos vetores e matrizes tenham coeficientes complexos.

Já estabelecemos algumas condições equivalentes para que λ seja um autovalor de A.

Seja A uma matriz $n \times n$ e seja λ um escalar. As seguintes afirmações são logicamente equivalentes:

a) λ é um autovalor de A;

b) $(A - \lambda I)x = 0$ tem uma solução não trivial;

c) $\mathcal{N}(A - \lambda I) \neq \{0\}$;

d) $A - \lambda I$ é singular;

e) $\det(A - \lambda I) = 0$.

2.7.1 Raio Espectral

Consideremos a seguinte definição.

Definição 2.7 *Seja A uma matriz $n \times n$ com autovalores λ_i, $1 \leq i \leq n$. Então*

$$\rho(A) = max_{1 \leq i \leq n} |\lambda_i|$$

é o raio espectral da matriz A.

2.8 Matrizes Especiais - II

Apresentamos, a seguir, mais algumas matrizes especiais de interesse. Entre elas: matrizes não negativas, as matrizes redutíveis e irredutíveis, as matrizes diagonalmente dominantes . Dentre os motivos principais para apontarmos tais matrizes, está o conceito de dominância em relação à diagonal da matriz. Este fato será usado, em momento oportuno, para estabelecer uma condição suficiente em relação à convergência de métodos iterativos para resolução sistemas lineares. Autores como ([4], [7]) e principalmente [16], apresentam mais detalhes sobre tais matrizes.

2.8.1 Matrizes não negativas

Em sistemas lineares que aparecem em muitas aplicações, os elementos da matriz de coeficientes representam quantidades não negativas. Apresentaremos algumas destas matrizes e suas propriedades.

Definição 2.8 *Uma matriz $A \in \mathbb{R}^{n\times n}$ com elementos reais é dita não negativa se $a_{ij} \geq 0$, $\forall i, j = 1, ..., n$ e é dita positiva se $a_{ij} > 0$, $\forall i, j = 1, ..., n$. Analogamente, um vetor $x = (x_1, ..., x_n)$ é não negativo se $x_i \geq 0$ para todo i e positivo se $x_i > 0$ para todo $i = 1, ..., n$.*

Teorema 2.4 ***(Perron)*** *Se A é uma matriz positiva $n \times n$, então A tem um autovalor real positivo λ com as seguintes propriedades:*

i) λ é uma raiz simples da equação característica;

ii) λ tem um autovetor positivo associado ;

iii) se r é outro autovalor qualquer de A, então $|r| < \lambda$.

Prova: Ver [7] ∎

2.8.2 Matrizes Redutíveis e Irredutíveis

Definição 2.9 *Uma matriz $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ é redutível se existe um subconjunto $S \subset N = \{1, 2, ..., n\}$, não vazio, com $S \neq N$, tal que $a_{ij} = 0$ para todo par de índices (i, j) onde $i \in S$ e $j \in N \backslash S$ ou equivalentemente, existe uma matriz de permutação P tal que*

$$P^T AP = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

onde A_{11} e A_{22} são matrizes quadradas. A matriz A é irredutível se A não é redutível.

Exemplo 2.5 *Seja* $A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & -1 & 0 & 3 & 0 & 1 \\ 2 & -1 & 5 & 3 & 1 & -1 \\ 0 & 1 & 0 & -1 & 0 & 2 \\ -6 & 3 & 4 & -2 & -5 & 1 \\ 0 & 3 & 0 & 1 & 0 & -1 \end{pmatrix}$.

Se tomarmos a matriz de permutação P como sendo $P = [e_1, e_5, e_3, e_4, e_2, e_6]$, onde e_i é o i-ésimo vetor coordenada, então $P^T AP = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$, onde A_{11} e A_{22} são matrizes 3×3.

Se A é redutível, então o sistema linear $Ax = b$ pode ser escrito da seguinte forma

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

o que implica em

$$A_{11}x_1 + A_{12}x_2 = b_1$$

$$A_{22}x_2 = b_2.$$

Então, se A_{22} é não singular, nós podemos primeiro resolver x_2 a partir de um sistema linear reduzido e então substituir a solução x_2 no primeiro subsistema para resolver x_1.

2.8.3 Matriz Diagonal Dominante

Definição 2.10 *Uma matriz $A = (a_{ij})$, $n \times n$, é diagonalmente dominante se*

$$|a_{ii}| \geq \sum_{j=1, j \neq i}^{n} |a_{ij}| = \Lambda_i \tag{2.14}$$

para todo $1 \leq i \leq n$.

A é estritamente diagonal dominante se a inequação em (2.14) é estrita para todo i. A é irredutível diagonal dominante se A é irredutível e diagonalmente dominante, com a inequação estrita em (2.14) para até no mínimo i, ([4], [7], [11]).

Exemplo 2.6 *As seguintes matrizes são diagonalmente dominante.*

$$A_1 = \begin{pmatrix} -4 & 1 & -2 \\ 5 & 10 & -2 \\ 1 & -1 & 6 \end{pmatrix}, \quad \begin{pmatrix} 3 & -1 & 1 \\ -1 & 6 & -2 \\ 1 & -2 & 4 \end{pmatrix}.$$

A_1 e A_I^T são linha diagonal dominante e coluna diagonal dominante, respectivamente. A_2 é linha e coluna diagonal dominante. Todas as matrizes diagonais são diagonalmente dominantes. Esta característica pode ser aproveitada para se estabelecer uma condição suficiente em relação a convergência dos métodos iterativos, discutidos no capítulo 4.

Capítulo 3

Métodos Diretos Para Resolução de Sistemas Lineares

3.1 Introdução

As equações lineares aparecem com grande freqüência, tanto na modelagem, quanto na resolução de problemas em diversas áreas das ciências ligadas às engenharias, economia, biologia, matemática aplicada, entre outras.

Uma equação linear, nas variáveis $x_1, x_2, \cdots, x_n$ é uma equação que pode ser escrita na forma

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b \tag{3.1}$$

onde b e os coeficientes $a_1, ..., a_n$ são números reais (ou complexos), geralmente já conhecidos. O subíndice n pode ser qualquer inteiro positivo.

Podemos também dizer que um sistema de equações lineares é uma coleção de uma ou mais equações lineares envolvendo as mesmas variáveis, digamos $x_1, ..., x_n$.

Dizemos que uma solução de tal sistema é uma lista $(s_1, s_2, ..., s_n)$ de números que torna verdadeira, simultaneamente, cada equação que compõe o sistema. Também, o conjunto de todas as soluções possíveis é chamado de conjunto solução do sistema linear e dois sistemas são ditos equivalentes, quando apresentam o mesmo conjunto solução. Geralmente, representamos um sistema linear retangular de m equações com n incógnitas escrevendo-o sob a forma abreviada $Ax = b$, com $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$,

$b \in \mathbb{R}^m$, ou na forma estendida,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots\ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots\ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots\ldots + a_{mn}x_n = b_m \end{cases} \tag{3.2}$$

Quando $m = n$ dizemos que o sistema é quadrado e terá a forma,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots\ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots\ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots\ldots + a_{nn}x_n = b_n \end{cases} \tag{3.3}$$

Neste trabalho estaremos interessados em sistemas como o que aparece em (3.3).

Na próxima seção destacamos os sistemas triangulares, bem como o algoritmo para resolver tal sistema. Vários autores, como ([1], [3], [4], [6], [15]) apresentam extensas discussões sobre este assunto.

3.2 Sistemas Triangulares

Na seção (2.4), caracterizamos as matrizes triangulares. Agora vamos detalhar sistemas de equações lineares onde aparecem tais sistemas. Então, seja $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, de forma que a matriz A seja triangular superior, ou seja, $a_{ij} = 0$ se $i > j$. Neste caso dizemos que o sistema está na forma triangular ou, simplesmente, sistema triangular.

Como sabemos, a solução de um sistema triangular superior é obtida por retro substituição, desde que $a_{nn} \neq 0$.

Para generalizar, tomemos o sistema triangular

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ \vdots \\ a_{nn}x_n = b_n \end{cases}$$

Admitindo-se que $a_{ii} \neq 0$, $i = 1, ..., n$, podemos calcular as incógnitas $x_1, ..., x_n$ através do seguinte procedimento

$$\begin{cases} x_n = b_n/a_{nn} \\ x_{n-1} = (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1} \\ \vdots \\ x_1 = (b_1 - a_{1n}x_n - \cdots - a_{13}x_3 - a_{12}x_2)/a_{11} \end{cases}$$

A partir desse processo geral, podemos estabelecer um algoritmo sistemático para a resolução de sistema triangulares.

Algoritmo 3.1 *Solução de Sistema Triangular Superior*

Dados a_{ij}, $j \geq i$, b_i, $1 \leq i, j \leq n$

Faça $x_n = \dfrac{b_n}{a_{nn}}$, *soma=0*

Para $k = n - 1, ..., 1,$

Faça soma=b_k

Para $j = k + 1, ..., n,$

Faça soma=soma$-a_{kj}x_j$

$x_k = \dfrac{soma}{a_{kk}}$

Fim

Fim

Para o caso em que a matriz seja triangular inferior, procedemos de modo análogo.

Na próxima seção descrevermos o Método de Eliminação Gaussiana, para resolvermos sistemas lineares como o que aparece em (3.3).

3.3 Método de Eliminação Gaussiana

O método de eliminação Gaussiana será usado para resolver um sistema com n equações e n incógnitas. Este método é considerado, em geral, o método computacional mais eficiente, já que envolve o menor número de operações aritméticas. Antes de abordarmos este método, porém, vamos estudar a sua construção e fundamentação. Inicialmente definimos sobre a matriz completa, associada a este sistema, um conjunto de operações sobre linhas, ditas elementares, usadas com o objetivo de produzir um sistema equivalente, mais simples. São descritas a seguir:

1) Trocar duas linhas ($l_i \leftrightarrow l_j \quad i \neq j$);

2) Multiplicar uma linha por um número real não nulo ($\alpha L_i \rightarrow L_j \quad \alpha \neq 0$);

3) Substituir uma linha por sua soma com um múltiplo de outra linha ($\alpha L_i + L_j \rightarrow L_j \quad i \neq j$).

3.3.1 Matrizes Elementares

Uma matriz elementar é uma matriz obtida, a partir da matriz identidade I, por uma das operações elementares sobre linhas acima descritas.

Em geral, suponha que E é uma matriz elementar $n \times n$. Podemos pensar em E como sendo obtida de I por uma operação elementar sobre as linhas ou sobre as colunas. Se A é uma matriz $n \times r$, multiplicar A por E à esquerda tem o efeito de efetuar a mesma operação sobre as linhas de A. Se B é outra matriz $m \times n$, multiplicar B por E à direita equivale a efetuar a mesma operação sobre as colunas de B.

Teorema 3.1 *Se E é uma matriz elementar, então E é inversível e E^{-1} é uma matriz elementar do mesmo tipo.*

Prova: Ver [10]. ∎

Definição 3.1 *Uma matriz B é equivalente por linhas a A se existe uma seqüência finita de matrizes elementares $E_1, E_2, ..., E_k$ tal que*

$$B = E_k E_{k-1} ... E_1 A.$$

Teorema 3.2 *Seja A uma matriz $n \times n$. Então as seguintes afirmações são logicamente equivalentes:*

(a) A é inversível;

(b) $Ax = 0$ tem apenas a solução trivial $\mathbf{0}$;

(c) A é equivalente por linhas a I.

Prova: Ver [11]. ∎

Corolário 3.2.1 *O sistema de equações lineares com n incógnitas $Ax = b$ tem uma única solução se e somente se A é inversível.*

Prova: Ver [11]. ∎

3.3.2 Método de Gauss sem Mudanças de Ordem das linhas

É possível definir, de forma sistemática, uma seqüência conveniente de operações elementares do tipo $1, 2, 3$, de modo a transformar um sistema linear, como o que aparece em (3.3), num outro equivalente, escrito na forma triangular. A discussão apresentada nesta subseção está baseada, principalmente, nos estudos apresentados em ([3], [4], [11]).

A solução de sistemas lineares usando uma eliminação ordenada de suas variáveis é atribuída ao matemático alemão Carl Friedrich Gauss (1777-1855).

De certa forma, é importante que a eliminação seja feita de forma sistemática, pois assim podemos elaborar algoritmos que poderão ser programados.

Para fixarmos a idéia geral do método e estabelecer o processo sistemático de triangularização de um sistema com n equações e n incógnitas, consideremos a matriz completa $[A \quad b]$, associada ao sistema $Ax = b$, onde $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$.

$$\begin{pmatrix} a_{11} & a_{12} & . & . & . & a_{1n} & b_1 \\ a_{21} & a_{22} & . & . & . & a_{2n} & b_2 \\ . & . & . & & & . & . \\ . & . & & . & & . & . \\ . & . & & & . & . & . \\ a_{n1} & a_{n2} & . & . & . & a_{nn} & b_n \end{pmatrix}$$

na qual cada linha l_i^1, $i = 1 : n$ representa uma equação do sistema linear que queremos triangularizar.

Inicialmente fazemos a eliminação da primeira coluna, cujo processo é descrito a seguir.

Supondo $a_{11} \neq 0$, para eliminar a incógnita x_1 das $(n-1)$ últimas equações vamos subtrair a primeira linha multiplicada pelo fator

$$m_{i1} = a_{i1}/a_{11},$$

de todas as outras linhas l_i, $i = 2 : n$.

Dessa meneira, estamos modificando os elementos das $n-1$ últimas linhas da seguinte maneira:

Para $i = 2 : n$

$$a_{ij}^2 = a_{ij} - m_{i1}a_{1j}, \quad j = 2 : n$$

$$b_i^2 = b_i - m_{i1}b_1.$$

O índice superior 2 em a_{ij}^2 e b_i^2 indica que vamos usar um segundo valor para a_{ij} e b_i.

No fim deste estágio, os coeficientes da matriz completa foram modificados de modo que a matriz assume a seguinte configuração:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & . & . & a_{1n} & b_1 \\ 0 & a_{22}^2 & a_{23}^2 & . & . & a_{2n}^2 & b_2^2 \\ . & . & . & & & . & . \\ . & . & & . & & . & . \\ . & . & & & . & . & . \\ 0 & a_{n2}^2 & a_{n3}^2 & . & . & a_{nn}^2 & b_n^2 \end{pmatrix}.$$

Na seqüência para eliminarmos a segunda coluna temos o seguinte procedimento:

para eliminarmos a incógnita x_2 das $n-2$ últimas equações repetimos o procedimento anterior tomando, agora, a segunda linha como auxiliar no processo de eliminação, isto é, faremos

$$l_i^3 = l_i^2 - m_{i2} l_2^2, \quad i = 3:n, \quad \text{onde} \quad m_{i2} = \frac{a_{i2}}{a_{22}}, \quad i = 3:n.$$

nesta etapa, estamos supondo $a_{22} \neq 0$.

Os coeficientes serão modificados segundo as relações.

Para $i = 3:n$,

$$a_{ij}^3 = a_{ij}^2 - m_{i2} a_{2j}^2, \quad j = 3:n$$

$$b_i^3 = b_i^2 - m_{i2} b_2^2.$$

Continuamos eliminando os elementos das colunas que estão abaixo da diagonal, repetindo a idéia descrita.

No processo de eliminação, os elementos $a_{11}, a_{22}^2, a_{33}^3, ..., a_{jj}^j$, que aparecem na diagonal de A são chamados pivôs. Se eles não se anulam, após a eliminação chegamos à forma triangular do sistema:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ \qquad a_{22}^2 x_2 + ... + a_{2n}^2 x_n = b_2^2 \\ \qquad\qquad\qquad . \\ \qquad\qquad\qquad . \\ \qquad\qquad\qquad . \\ \qquad\qquad\qquad a_{nn}^n x_n = b_n^n \end{cases}$$

Se no processo de eliminação um dos pivôs se anular, devemos trocar linhas (sempre escolhendo equelas abaixo da diagonal para não perder a eliminação anterior), de modo que escolhamos elementos não nulos para pivôs. Este processo será descrito na próxima seção.

A sistemática do Método de Eliminação de Gauss sem troca de ordem das linhas, é descrita pelo algoritmo a seguir, que aparece, por exemplo em ([3], [4], [11]).

Algoritmo 3.2 *Método de Gauss sem troca de linhas*

Para $k = 1, 2, ..., n-1$

Para $i = k+1, ..., n$

Defina $m_{ik} = \dfrac{a_{ik}}{a_{kk}}$, *(desde que* $a_{kk} \neq 0$*)*

Para $j = k+1, ..., n$

Defina $a_{ij} = a_{ij} - m_{ik}a_{kj}$

Fim

Fim

Fim

3.4 Estratégia do Pivoteamento

Nesta seção vamos apresentar um algoritmo para o método de Gauss com troca de ordem entre linhas. Este procedimento é necessário, principalmente para evitar que um pivô mal condicionado interfira na precisão e até mesmo na condição de existência de soluções de um sistema.

Em cada etapa do algoritmo será necessário escolher a linha do pivô. Podemos, muitas vezes, evitar acúmulo de erros desnecessários escolhendo a linha do pivô de maneira razoável.

Podemos chamar então, esta estratégia, como sendo Método de Gauss com troca de ordem entre linhas e sistematizá-la pelo algoritmo a seguir, veja ([3], [11]).

Algoritmo 3.3 *Método de Gauss com Troca de Linhas*

Para $i = 1, ..., n$

Defina $p(i) = i$

Fim

Para $i = 1, ..., n$

(1) Escolha um pivô $a_{p(j),i}$ *entre os elementos*

$$a_{p(i),i}, a_{p(i+1),i}, ..., a_{p(n),i}$$

(2) Troque as i*-ésimas e* j*-ésimas elementos de* p

(3) Para $k = i = 1, \cdots, n$

Defina $m_{p(k),i} = a_{p(k),i}/a_{p(i),i}$

Para $j = i+1, \cdots, n$

Defina $a_{p(k),j} = a_{p(k),j} - m_{p(k),i}a_{p(i),j}$

Fim

Fim

Fim

Na estratégia do pivoteamento procedemos a troca sistemática de linhas, de modo que o pivô seja o maior elemento, em valor absoluto, da coluna que estamos eliminando. Com esta estratégia, no k-ésimo passo procuramos o elemento de maior valor absoluto, dentre aqueles que estão na mesma coluna, abaixo da diagonal.

3.5 Fatoração LU

O método de eliminação pode ser usado economicamente quando precisamos resolver vários sistemas com a mesma matriz dos coeficientes. Em algumas situações práticas, os diversos termos independentes, b, não estão disponíveis simultaneamente ou podem depender da própria solução do sistema associada a outro b.

Uma opção seria guardar os coeficientes m_{ij} calculados no processo de eliminação e usá-los na atualização de novos termos independentes, b. Outra alternativa computacionalmente equivalente é o que chamamos *fatoração* ou *decomposição* **LU** da matriz **A**. A base do método também está apoiada na simplicidade de resolução de sistemas triangulares.

Suponhamos que seja possível fatorar a matriz **A** num produto de uma matriz triangular inferior, (com os elementos da diagonal principal iguais a 1) **L** e uma matriz triangular superior **U**, isto é,

$$\mathbf{A}=\mathbf{LU}. \tag{3.4}$$

Nestas condições, o sistema $Ax = b$ pode ser escrito na forma $LUx = b$, o que permite o desmembramento em dois sistemas triangulares

$$Ly = b \qquad \text{e} \qquad Ux = y.$$

Resolvendo o primeiro sistema, calculamos y que, usado no segundo sistema, fornecerá o vetor procurado x.

Dessa maneira, conhecidas L e U, o sistema será resolvido com $2n^2$ operações (dois sistemas triangulares), o que representa um ganho substancial comparado com as $2n^3/3$ operações do método de eliminação.

A questão básica da existência dos fatores L e U é tratada em livros especializados em análise matricial, como por exemplo. ([4], [7]. [15]).

Dada uma matriz A, os fatores L e U são únicos se exigirmos que todos os elementos da diagonal de L sejam iguais a 1.

Dessa maneira, se chamarmos m_{ij} os elementos de L, e de u_{ij} os elementos de U, obtemos expressões que viabilizam uma implementação eficiente do método em questão (lembrando que $m_{ij} = 1$, para todo i):

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} m_{ik} u_{kj} \qquad i \leq j$$

e

$$m_{ij} = (a_{ij} - \sum_{k=1}^{j-1} m_{ik} u_{kj})/u_{jj} \qquad i > j.$$

O algoritmo abaixo sintetiza esta importante fatoração:

Algoritmo 3.4 *Fatoração LU*

Dado $A = [a_{ij}]$

Para $i = 1 : n$,

Para $j = i : n$,

$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} m_{ik} u_{kj}$

Para $j = i + 1 : n$,

$m_{ij} = (a_{ij} - \sum_{k=1}^{i-1} m_{jk} u_{ki}/u_{ii})$

Fim

Fim

Fim

Na seqüência aproveitamos a estrutura da matriz A associada ao sistema linear, para explorar uma fatoração matricial, conhecida como Fatoração de Cholesky ([4], [7]).

3.6 Fatoração de Cholesky

Se A é uma matriz simétrica positiva definida, então A pode ser fatorada em um produto LL^T, onde L é triangular inferior com elementos diagonais positivos.

Teorema 3.3 *Seja A uma matriz simétrica $n \times n$. As seguintes afirmações são equivalentes:*

a) A é positiva definida;

b) Todas as submatrizes principais $A_1, ..., A_n$ têm determinante positivo;

c) A pode ser reduzida por linhas a uma forma triangular superior usando apenas operações elementares do tipo 3 com todos os pivôs positivos;

d) A tem uma decomposição de Cholesky LL^T (onde L é triangular inferior com todos os elementos diagonais positivos);

d) A pode ser fatorada em um produto B^TB para alguma matriz invertível B.

Prova: Ver [11]. ∎

Esta é a *decomposição de Cholesky*, sistematizada pelo algoritmo abaixo:

Algoritmo 3.5 *Fatoração de Cholesky*

Dado $A = [a_{ij}]$, matriz simétrica e positiva definida

Para $j = 1 : n$,

$d_j = a_{jj} - \sum d_k l_{jk}$

Para $i = j + 1 : n$,

$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} d_k l_{ik} l_{jk})/d_j$

Fim

Fim

A existência da decomposição de Cholesky é uma condição necessária e suficiente para que uma matriz simétrica seja positiva definida. Assim, o Algoritmo acima também pode ser usado para verificarmos se uma matriz simétrica é positiva definida (ver [10]).

A caracterização apresentada, para uma matriz simétrica positiva definida, nos será útil futuramente para estabelecermos condições suficientes para convergência dos métodos iterativos a serem estudados.

Capítulo 4

Métodos Iterativos para Resolução de Sistemas Lineares

4.1 Introdução

Os métodos numéricos são ferramentas úteis para resolver problemas reais. Muitos destes problemas exigem a solução de sistemas de equações lineares $Ax = b$, grandes e esparsos, os quais aparecem, por exemplo, em análise não linear, problemas de programação linear, entre outros.

Os métodos diretos usuais não podem ser normalmente aplicados, por causa da precisão e custo de armazenamento. Por estas razões, normalmente prefere-se os métodos iterativos para resolver sistemas onde a matriz de coeficientes é esparsa. Podem ser usados também para reduzir os erros de arredondamento na solução obtida pelos métodos exatos. Um método é iterativo quando fornece uma seqüência de aproximantes da solução, cada um dos quais sendo obtido dos anteriores pela repetição do mesmo tipo de processo. Nos métodos iterativos com aproximação inicial, uma seqüência $\{x^{(k)}\}$ é gerada para aproximar a solução do sistema linear, tal que, $x^{(k)}$ converge para a solução desejada, x. De fato, Gauss (1823), foi quem primeiro sugeriu um método com estrutura iterativa para resolver sistemas lineares. Recentemente o estudo de métodos iterativos para resolver sistemas lineares é muito ativo e desperta a atenção de muitos pesquisadores pelo mundo. Fatores como esparsidade, custo computacional. armazenamento, convergência e precisão, são os mais estudados. Naturalmente, a notação empregada em todo o trabalho, estará de acordo com àquelas apresentadas nas seções do primeiro capítulo. A necessidade de se resolver sistemas de equações lineares aparece numa grande quantidade de problemas científicos, seja como

problema principal ou como subproblema do problema original. Para a sua resolução são apresentados, na literatura, uma grande variedade de métodos ([3], [4], [7], [14] [15], [17]).

Respeitando-se a distinção dos métodos, é comum catalogá-los em dois grupos:

i) Métodos Diretos

ii) Métodos Iterativos

Os Métodos Diretos são aqueles que conduzem à solução exata, a menos de erros de arredondamento introduzidos pela máquina, após um número finito de passos. Se os métodos diretos tem a vantagem de fornecer a solução após um número finito de passos e não depender de condições de convergência, eles podem ser inviáveis, por exemplo, quando o sistema é muito grande ou mal condicionado. Essas desvantagens são superadas pelos métodos iterativos, os quais se baseiam na construção de seqüências de aproximações onde, a cada passo, os valores calculados anteriormente são usados para melhorar a aproximação corrente. É claro que o método iterativo será útil e de interesse, somente se a seqüência de aproximações construídas pelo método convergir para a solução do sistema. Esta convergência esperada, está diretamente vinculada à matriz de coeficientes do sistema, sobre a qual deve-se observar algumas condições para se garantir tal convergência da seqüência de aproximações obtida.

Os métodos iterativos são, em síntese, uma grande variedade de técnicas usando aproximações sucessivas em cada passo, com a finalidade de se obter uma solução mais apurada para um sistema linear

$$Ax = b, \tag{4.1}$$

onde $A = (a_{ij})$ é uma matriz em $\mathbb{R}^{n\times n}$, não singular.

4.2 Conceitos Básicos

Nesta seção apresentaremos conceitos básicos relativos a métodos iterativos, que serão descritos na proximas seções, principalmente resultados de convergência.

Há dois tipos de métodos iterativos: os métodos estacionários

$$x^{(k+1)} = Gx^{(k)} + c, \quad k = 0, 1, ..., \tag{4.2}$$

e os métodos não estacionários

$$x^{(k+1)} = G_k x^{(k)} + c, \quad k = 0, 1, ... \tag{4.3}$$

A matriz G que aparece em (4.2) é chamada matriz de iteração do método, sobre a qual reside grande parte das características de cada um destes.

Os métodos estacionários são clássicos, simples de serem estruturados e implementados e muito utilizados. Os métodos não-estacionários são relativamente novos e usualmente difíceis para se estabelecer sua análise e fundamentação, mas podem ser bem efetivos.

Nosso objeto de interesse serão os métodos ditos estacionários, entre os quais: o Método de Jacobi, o Método de Gauss-Seidel e o Método SOR. A discussão relativa a esta seção baseia-se principalmente em ([3], [4], [7], [15], [17]).

Inicialmente, consideremos a seguinte definição:

Definição 4.1 *O método iterativo (4.2) é convergente se*

$$\lim_{k\to\infty} x^{(k)} = x^*$$

onde $\{x^{(k)}\}$ *é gerada por (4.2) e* x^* *é a solução exata do sistema linear* $Ax = b$

Se o método (4.2) é convergente, então

$$x^* = Gx^* + c, \tag{4.4}$$

isto é, $(I - G)x^* = c$. Além disso, se x^* é a solução do sistema referido em (4.1), então existe uma matriz Q tal que

$$I - G = QA \quad e \quad c = Qb \tag{4.5}$$

e neste caso, o método definido em (4.2) é dito consistente.

Exemplo 4.1 *Para $Ax = b$, nós tomamos $Q = \alpha I$ onde α é um escalar. Então, substituindo-se em (4.2), tem-se:*

$$x_{k+1} = (I - \alpha A)x_k + \alpha b$$

sendo consistente pois

$$I - GA = I - I + \alpha A = QA \quad e \quad c = \alpha b = Qb$$

onde $G = I - \alpha A$.

Teorema 4.1 *O método definido em (4.2) é convergente se e somente se*

$$\rho(G) < 1, \tag{4.6}$$

onde $\rho(G)$ é o raio espectral da matriz iterativa G.

Prova: Ver [7] ∎

Note que se $\rho(G) = 1$, o método (4.2) é convergente somente para algum vetor especial $x^{(0)}$ inicial. Caso contrário, este método é divergente [17].

Exemplo 4.2 *O correspondente método iterativo para $2Ix = b$ é $x^{(k+1)} = -x^{(k)} + b$. Neste caso, o método é consistente por causa de $G = -I$, $I - G$ é não singular e $c = b = (I - G)A^{-1}b$. Mas se nós escolhemos o vetor inicial $x^{(0)} = 0$, o método é divergente por que $\rho(G) = 1$. Neste caso, nós temos o conceito de convergência fraca, que será definida a seguir.*

A seguir descrevemos os métodos iterativos que abordaremos neste trabalho e que, juntamente com os métodos diretos abordados no capítulo anterior, darão origem à principal contribuição dessa dissertação.

4.3 O Método Iterativo de Jacobi

Carl Gustav Jacobi (1804-1851) apresentou o método iterativo que recebe o seu nome em 1845. Para introduzi-lo, consideremos o sistema linear $Ax = b$ onde $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ é não singular. Podemos reescrever o sistema anterior sob a forma

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{4.7}$$

Suponha que todas as entradas da diagonal de A sejam não nulas. isto é, $a_{ii} \neq 0$. $\forall \ i = 1, 2, ...n$. Então podemos resolver cada incógnita x_i da i-ésima equação de (4.7) considerando que as outras entradas de x permanecem fixas. Então nós temos

$$(J) \begin{cases} x_1 = [b_1 - a_{12}x_2 - \cdots - a_{1n}x_n]/a_{11} \\ x_2 = [b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n]/a_{22} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_i = [b_i - a_{i1}x_1 - ... - a_{i,i-1}x_{i-1} - a_{i,i+1}x_{i+1} - \cdots - a_{in}x_n = b_n]/a_{ii} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_n = [b_n - a_{n1}x_1 - \cdots - a_{n,n-1}x_{n-1}]/a_{nn} \end{cases} \tag{4.8}$$

isto é,

$$x_i = [b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j]/a_{ii}, \quad (1 \leq i \leq n). \tag{4.9}$$

Veja que (4.9) sugere o método iterativo definido por

$$x_i^{(k+1)} = \frac{1}{a_{ii}}[b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}] = \frac{1}{a_{ii}}[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}] \quad (1 \leq i \leq n), \tag{4.10}$$

ou ainda

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{pmatrix} = \begin{pmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{pmatrix} \left[\begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & 0 & a_{23} & \cdots & a_{2n} \\ & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \right]$$

O método (4.10) é o método de Jacobi.

Como o método de Jacobi produz atualizações das entradas de x independentemente e simultaneamente, é também conhecido como o método das substituições sucessivas. Maiores detalhes podem ser obtidos em ([3], [4], [17]).

Agora apresentaremos o método sob a forma matricial. Para isto, considere a seguinte decomposição para a matriz A, associada ao sistema (4.1):

$$A = D + L + U \tag{4.11}$$

em que

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix},$$

$$L = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \cdots & \cdots & \ddots & \cdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

em que D é uma matriz diagonal $n \times n$, L e U são matrizes $n \times n$ triangular inferior e superior, respectivamente e b é um vetor em $\mathbb{R}^n$. Esta decomposição é estudada, por exemplo, em ([1], [4], [15], [17]).

Então (4.10) pode ser reescrita da seguinte forma

$$x^{(k+1)} = -D^{-1}(L + U)x^{(k)} + D^{-1}b. \tag{4.12}$$

Daqui em diante chamaremos a matriz $-D^{-1}(L+U)$ de matriz de Jacobi e a denotaremos por M_J. Portanto,

$$M_J = -D^{-1}(L+U). \tag{4.13}$$

O algoritmo numérico para o método de Jacobi é o seguinte:

Algoritmo 4.1 *Método de Jacobi*

Dado $x^{(0)} \in \mathbb{R}^n$

Supondo $a_{ii} \neq 0$

Para $k = 0, 1, 2, \cdots$

Para $i = 1, 2, \cdots, n$

$s = 0$

Para $j = 1, 2, \cdots, i-1, i+1, \cdots, n$

$s = s + a_{ij}x^{(k)}$

Fim

$x_i^{(k+1)} = (b_i - s)/a_{ii}$

Fim

Verifique convergência, continue se necessário

Fim

Pelo teorema (4.1), a seqüência $\{x^k\}$ gerada pelo algoritmo (4.1), determinado pelo método de Jacobi, convergirá para uma solução do sistema linear $Ax = b$ dado pela relação (4.1) se $\rho(M_J) < 1$ onde M_J é a matriz de Jacobi definida na relação (4.13).

Na seqüência citaremos algumas condições suficientes para convergência do método de Jacobi.

Teorema 4.2 *O método de Jacobi é convergente quando A é uma matriz do tipo das seguintes:*

1. *A é estritamente diagonal dominante.*
2. *A é irredutível diagonal dominante.*

Prova: Ver [17] ∎

Exemplo 4.3 *Suponha que queiramos resolver o sistema linear*

$$\begin{pmatrix} 5 & 1 & -1 & 0 \\ -1 & 6 & -1 & 1 \\ -1 & 1 & 7 & 0 \\ 0 & 1 & -1 & 8 \end{pmatrix} x = \begin{pmatrix} 5 \\ -5 \\ -9 \\ 8 \end{pmatrix}, \text{ cuja solução exata é } (1, -1, -1, 1).$$

Como a matriz de coeficientes é diagonal dominante, o método de Jacobi converge (Parte 1 do Teorema 4.2). Sabemos que $\rho(M_J) = 0.1944 < 1$, calculado para este exemplo. Então aplicamos o algoritmo anterior para este exemplo, com vetor inicial $x^{(0)} = 0$. Algumas iterações resultantes são:

$$x^{(3)} = \begin{pmatrix} 1.0048 \\ -1.0096 \\ -1.0061 \\ 1.0030 \end{pmatrix}, \quad x^{(5)} = \begin{pmatrix} 1.0006 \\ -0.9996 \\ -0.9998 \\ 1.0003 \end{pmatrix}, \quad x^{(7)} = \begin{pmatrix} 1.0000 \\ -1.0000 \\ -1.0000 \\ 1.0000 \end{pmatrix}.$$

4.4 O Método de Gauss-Seidel

Gauss apresentou seu método iterativo para resolver sistemas de equações provenientes do Método dos Quadrados Mínimos. Seidel foi aluno de Jacobi e publicou a versão hoje usada, em 1874.

O método de Gauss-Seidel pode ser visto como uma modificação do Método de Jacobi. Nele, as iterações serão calculadas aproveitando-se os cálculos já atualizados, das outras componentes, para atualizar a componente que está sendo calculada. Veja ([3], [17]).

Formalmente, nós temos novos valores das entradas x_j ($j = 1, 2, \cdots, i-1$) de x quando nós calculamos x_i no passo $k+1$.

Em geral, tem-se que, $x_1^{(k+1)}, x_2^{(k+1)}, \cdots, x_{i-1}^{(k+1)}$ é melhor que $x_1^{(k)}, x_2^{(k)}, \cdots, x_{i-1}^{(k)}$.

Em seguida, para melhorar a taxa de convergência do método de Jacobi, nós substituiremos $x_1^{(k)}, x_2^{(k)}, \cdots, x_{i-1}^{(k)}$ por $x_1^{(k+1)}, x_2^{(k+1)}, \cdots, x_{i-1}^{(k+1)}$ em (4.10). Então nós obtemos um novo método iterativo, chamado método de Gauss-Seidel, cujo argumento principal é mostrado a seguir

$$x_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)})/a_{ii} \quad (1 \leq i \leq n), \tag{4.14}$$

que na forma estendida é

$$(G) \begin{cases} x_1^{(k+1)} = \dfrac{1}{a_{11}}[b_1 - a_{12}x_2^{(k)} - \cdots - a_{1n}x_n^{(k)}] \\ x_2^{(k+1)} = \dfrac{1}{a_{22}}[b_2 - a_{21}x_1^{(k+1)} - \cdots - a_{2n}x_n^{(k)}] \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_l^{(k+1)} = \dfrac{1}{a_{ll}}[b_l - a_{l1}x_l^{(k+1)} - \cdots - a_{l,l-1}x_{l-1}^{(k+1)} - a_{l,l+1}x_{l+1}^{(k)} - a_{l,l+1}x_{l+1}^{(k)}] \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_n^{(k+1)} = \dfrac{1}{a_{nn}}[b_n - a_{n1}x_{21}^{(k+1)} - \cdots - a_{n,n-1}x_{n-1}^{(k+1)}] \end{cases}$$

Na forma matricial, (4.14) é

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \cdots & \cdots\cdot & \ddots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} x^{(k+1)} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} - \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & \cdots & \cdots & a_{2n} \\ \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} x^{(k)}$$

ou separando a matriz à esquerda desta última igualdade

$$\left[\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \cdots & \cdots & \ddots & \cdots \\ 0 & \cdots & \cdots & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \cdots & \cdots & \ddots & \cdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{pmatrix}\right] x^{(k+1)} = - \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & \cdots & \cdots & a_{2n} \\ \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} x^{(k)} + \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n, \end{pmatrix}.$$

ou seja,

$$(D + L)x^{(k+1)} = -Ux^{(k)} + b \tag{4.15}$$

ou

$$x^{(k+1)} = -(D + L)^{-1}Ux^{(k)} + (D + L)^{-1}b. \tag{4.16}$$

A matriz $-(D + L)^{-1}U$ será chamada, daqui em diante, de M_{gs} ou matriz de Gauss-Seidel e denotada

$$M_{gs} = -(D + L)^{-1}U. \tag{4.17}$$

A sistemática do Método de Gauss-Seidel é apresentada no algoritmo seguinte.

Algoritmo 4.2 *Método de Gauss-Seidel*

Supondo $a_{ii} \neq 0, \quad i = 1, 2, \cdots, n$

Dado $x^0 \in \mathbb{R}^n$

Para $k = 0, 1, 2, \cdots$

  Para $i = 1, 2, \cdots, n$

    $s = 0$

      Para $j = 1, 2, \cdots, i - 1$

        $s = s + a_{ij}x_j^{(k+1)}$

      Fim

      Para $j = i + 1, \cdots, n$

        $s = s + a_{ij}x_j^{(k)}$

      Fim

$x_i^{(k+1)} = (b_i - s)/a_{ii}$

Fim

Cheque convergência, continue se necessário

Fim

Pelo teorema (4.1), a seqüência $\{x^{(k)}\}$ gerada pelo algoritmo (4.2) convergirá para uma solução do sistema linear $Ax = b$ se a matriz de Gauss-Seidel M_{gs}, dada na relação (4.17), satisfizer $\rho(M_{gs}) < 1$. O método de Gauss-Seidel em geral é mais eficiente que o método de Jacobi, no sentido de que converge mais rápido. Intuitivamente isto pode ser percebido pela própria "construção"destes métodos. Veremos mais tarde, através de resultados teóricos e também de testes numéricos, que isto de fato se verifica.

A seguir apresentaremos um outro método iterativo que acelera a convergência do método de Gauss-Seidel, que é conhecido na literatura, como método de aceleração ou método SOR.

4.5 O Método SOR

O Método da Sobrerelaxação Sucessiva, que é conhecido na literatura como SOR (sucessive Over relaxation method), consiste em uma generalização do método de Gauss-Seidel. Veremos, mais adiante, que para um certo parâmetro ω, o método de Gauss-Seidel se torna um caso particular do método SOR.

Iniciando com um vetor arbitrário $x^{(0)} \in \mathbb{R}^n$, o método SOR gera novos vetores que aproximam a solução do sistema $Ax = b$ da seguinte forma:

Combina a solução obtida pelo método de Gauss-Seidel com a solução da iterada atual.

A combinação é feita utilizando-se um peso que, escolhido adequadamente, acelera a convergência da seqüência gerada pelo método.

Desta forma, o método SOR para resolver o sistema linear $Ax = b$, dado na relação (4.1) é definido por

$$(S) \qquad \begin{cases} x_i^{(k+1)} &= \dfrac{1}{a_{ii}}[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}] \\ \\ x_i^{(k+1)} &= \omega x_i^{(k+1)} + (1-\omega)x_i^{(k)} \end{cases}$$

No Método SOR, ω é chamado parâmetro de sobre-relaxação. Pode-se observar que quando $\omega = 1$ temos o método de Gauss-Seidel. A escolha $1 < \omega < 2$ caracteriza os métodos de sobre-relaxação, ao passo que os métodos de sub-relaxação são obtidos por valores $0 < \omega < 1$. A prática mostra que melhores resultados são obtidos na sobre-relaxação. Dentre os resultados de convergência dos métodos SOR destacamos que o método só converge se $0 < \omega < 2$.

Reescrevendo (S) em uma única equação obtém-se:

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}}[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}] + (1-\omega)x_i^{(k)}, \quad i = 1, 2, \cdots . n. \tag{4.18}$$

Desta última expressão, juntamente com a expressão matricial para o método de Gauss-Seidel (4.16), obtém-se a forma matricial para o método SOR.

$$(L + \omega D)x^{(k+1)} = +[(1-\omega)D - \omega U]x^{(k)} + \omega b \tag{4.19}$$

ou

$$x^{(k+1)} = (L + \omega D)^{-1}[(1-\omega)D - \omega U]x^{(k)} + \omega(L + \omega D)^{-1}b. \tag{4.20}$$

Como fizemos antes com os métodos de Jacobi e Gauss-Seidel denotaremos a matriz do método SOR por M_{SOR} e esta é dada por

$$M_{SOR} = (L + \omega D)^{-1}((1-\omega)D - \omega U) \tag{4.21}$$

De acordo com o teorema (4.1) o método SOR converge para uma solução do sistema linear $Ax = b$ sempre que $\rho(M_{SOR}) < 1$ onde $\rho(M_{SOR})$ é o raio espectral da matriz M_{SOR} dada na relação (4.21).

A sistemática iterativa do método SOR é descrita pelo procedimento algorítmico a seguir.

Algoritmo 4.3 *Método SOR*

Supondo $a_{ii} \neq 0$

Dado $x^{(0)} \in \mathbb{R}^n$ *e* $\omega \in \mathbb{R}$

Para $k = 0, 1, 2, \cdots$

Para $i = 1, 2, \cdots, n$

$s = 0$

Para $j = 1, 2, \cdots, i-1$

$s = s + a_{ij} x_j^{(k+1)}$

Fim

Para $j = i+1, \cdots, n$

$s = s + a_{ij} x_j^{(k)}$

Fim

$s = (b_i - s)/a_{ii}$

$x_i^{(k+1)} = x_i^{(k)} + \omega(s - x_i^{(k)})$

Fim

Cheque convergência, continue se necessário

Fim

A seqüência $\{x^{(k)}\}$ gerada pelo algoritmo 4.3 aproximará uma solução do sistema linear $Ax = b$ se o raio espectral da matriz M_{SOR} dada na relação (4.21) for estritamente menor que a unidade, como já mencionamos anteriormente.

Como vimos, nos métodos estudados neste capítulo, as matrizes que dão origem a estes métodos são importantes para decidir se o método é convergente para uma solução do sistema considerado.

A seguir vamos descrever outros resultados, conhecidos na literatura, quando tais matrizes ou a matriz dos coeficientes do sistema que se procura resolver, tem alguma estrutura especial ou alguma propriedade conhecida.

Para os resultados a seguir considere o sistema linear $Ax = b$, dado em (4.1).

Teorema 4.3 *Se a matriz A do sistema linear $Ax = b$ é positiva definida, o método de Gauss-Seidel é sempre convergente e o método SOR converge sempre que $0 < \omega < 2$.*

Prova: Ver [6]. ∎

Nota: A primeira parte do teorema não se aplica ao método de Jacobi, o que pode ser verificado através do seguinte exemplo:

Exemplo 4.4 $A = \begin{pmatrix} 1 & \alpha & \alpha \\ \alpha & 1 & \alpha \\ \alpha & \alpha & 1 \end{pmatrix}$, $\forall \alpha \in I\!R$.

É fácil checar que a matriz A é positiva definida se $\alpha \in (-0.5, 1)$. Por outro lado, pelo teorema (4.1), o raio espectral da matriz de Jacobi M_j dada na relação (4.7) é menor que a unidade se $\alpha \in (-0.5, 0.5)$. Portanto, para $\alpha \in (-0.5, 1)$ a matriz A é positiva definida e o método de Jacobi não converge.

Agora, se a matriz A dos coeficientes do sistema $Ax = b$ é tridiagonal, então tem-se o seguinte resultado.

Teorema 4.4 *Se os métodos de Jacobi e Gauss-Seidel são aplicados a um sistema linear $Ax = b$ em que A é tridiagonal então ou ambos convergem ou ambos divergem. Se ambos convergirem, Gauss-Seidel convergirá mais rápido que Jacobi.*

Prova: Ver [13]. ∎

Um resultado que se aplica aos três métodos é quando a matriz dos coeficientes do sistema $Ax = b$ é tridiagonalmente dominante.

Teorema 4.5 *Se a matriz A dos coeficientes do sistema $Ax = b$ é tridiagonalmente dominante então os métodos de Jacobi, Gauss-Seidel e SOR são sempre convergentes.*

Prova: Ver [6]. ∎

Para finalizar os resultados de convergência, apresentaremos um teorema conhecido na literatura, para determinar o parâmetro ω ótimo para o método SOR, quando a matriz A do sistema é positiva definida e tridiagonal.

Teorema 4.6 *Se a matriz A dos coeficientes do sistema $Ax = b$ é positiva definida e tridiagonal então $\rho(M_{gs}) = (\rho(M_j))^2 < 1$ e a escolha ótima ω para o método SOR é*

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(M_j)^2}}$$

onde $\rho(M_{gs})$ *e* $\rho(M_j)$ *são, respectivamente, os raios espectrais da matriz de Gauss-Seidel e da matriz de Jacobi.*

Prova: Ver ([4], [13]). ∎

4.6 Métodos Híbridos

Nesta seção descreveremos a principal contribuição deste trabalho. Chamaremos de métodos híbridos, os três estudados nas seções precedentes combinados com uma iteração do método de eliminação de Gauss, descrito na subseção 3.3.2.

A metodologia para a determinação destes métodos híbridos será a seguinte:

Faremos uma iteração do método de eliminação de Gauss, descrito no capítulo 2, e em seguida descreveremos novos métodos baseados nos métodos de Jacobi, Gauss-Seidel e SOR. Desta forma, a partir do sistema linear

$$Ax = b \text{ em que } A \in \mathbb{R}^{n\times n} \text{ e } x, b \in \mathbb{R}^n,$$

escrito na forma estendida como

$$\text{(SE)} \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

e supondo que o elemento a_{11} é não nulo (caso este seja nulo pode-se procurar, na primeira coluna, se algum dos elementos a_{i1} é não nulo, $i = 2, \cdots, n$, e então permutar as linhas), obtemos o sistema

$$\text{(SE1)} \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ (a_{22} - m_{21}a_{12})x_2 + \cdots + (a_{2n} - m_{21}a_{1n})x_n = b_2 - m_{21}b_1 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ (a_{n2} - m_{n1}a_{12})x_2 + \ldots\ldots + (a_{nn} - m_{n1}a_{1n})x_n = b_n - m_{n1}b_1 \end{cases}$$

em que $m_{i1} = \dfrac{a_{i1}}{a_{11}}$, $i = 2, 3, \cdots, n$.

Observe que após a primeira iteração do método de eliminação de Gauss, o sistema (SE) é transformado no sistema (SE1) em que x_1 desaparece da segunda até a n-ésima equação, na primeira coluna.

Como os sistemas (SE) e (SE1) são equivalentes, deduziremos novos métodos iterativos a partir do sistema linear (SE1). Os resultados de convergência, para estes métodos, estarão relacionados com a matriz dos coeficientes, H, associada ao sistema linear (SE1), ou seja,

$$H = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} - m_{21}a_{12} & \cdots & a_{2n} - m_{21}a_{1n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & a_{n2} - m_{n1}a_{12} & \cdots & a_{nn} - m_{n1}a_{1n} \end{pmatrix}. \qquad (4.22)$$

4.6.1 Método de Jacobi Híbrido

Nesta seção, descrevemos o método de Jacobi híbrido aplicado ao sistema linear (SE1), resultados de convergência e o algoritmo relacionado a este.

Assim, o método de Jacobi para o sistema (SE1) é dado por

$$\begin{cases} x_1^{(k+1)} = [b_1 - a_{12}x_2^{(k)} - \cdots - a_{1n}x_n^{(k)}]/a_{11} \\ x_2^{(k+1)} = [(b_2 - m_{21}b_1) - (a_{23} - m_{21}a_{13})x_3^{(k)} - \cdots - (a_{2n} - m_{21}a_{1n})x_n^{(k)}]/(a_{22} - m_{21}a_{12}) \\ \cdots\cdots\cdots\cdots \\ x_n^{(k+1)} = [(b_n - m_{n1}b_1) - (a_{n2} - m_{n1}a_{12})x_2^{(k)} - \ldots - (a_{n,n-1} - m_{n1}a_{1,n-1})x_n^{(k)}]/(a_{nn} - m_{n1}a_{1n}) \end{cases}$$

cuja forma matricial para este é dada por

$$x^{(k+1)} = -\bar{D}^{-1}(\bar{L} + \bar{U})x^{(k)} + \bar{D}^{-1}\bar{b} \qquad (4.23)$$

em que

$$\bar{D}^{-1} = \begin{pmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22} - m_{21}a_{12}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn} - m_{n1}a_{1n}} \end{pmatrix},$$

$$\bar{L} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ \ddots & & & & \\ 0 & 0 & 0 & \cdots & 0 \\ & & \ddots & & \\ 0 & a_{32} - m_{31}a_{13} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & & \cdots \\ 0 & a_{n2} - m_{n1}a_{12} & a_{n3} - m_{n1}a_{13} & \cdots & 0 \end{pmatrix},$$

$$\bar{U} = \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} - m_{21}a_{13} & \cdots & a_{2n} - m_{21}a_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & a_{n-1,n} - m_{n-1,1}a_{1n} \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

e

$$\bar{b} = \begin{pmatrix} b_1 \\ b_2 - m_{21}b_1 \\ \vdots \\ b_n - m_{n1}b_1 \end{pmatrix}.$$

Vamos denotar por

$$M_{\bar{J}} = -\bar{D}^{-1}(\bar{L} + \bar{U}), \tag{4.24}$$

a matriz de Jacobi para esta nova configuração.

Desta forma, os resultados de convergência para o método clássico de Jacobi, estudado nas seções precedentes, se aplicam também para o método de Jacobi híbrido, proposto em (4.23).

Assim, o método de Jacobi híbrido converge para uma solução do sitema linear (SE1), quando:

- $\rho(M_{\bar{J}}) < 1$, onde $\rho(M_{\bar{J}})$ é o raio espectral da matriz de Jacobi $M_{\bar{J}}$, de acordo com o teorema(4.1).
- A matriz H dos coeficientes do sistema linear (SE1) é tridiagonalmente dominante. Veja teorema (4.5).
- A matriz H dos coeficientes do sistema linear (SE1) é positiva definida e tridiagonal. Veja o teorema (4.3).

Algoritmo 4.4 *Método de Jacobi Híbrido*

Dado $x^{(0)} \in \mathbb{R}^n$

Supondo $a_{ii} \neq 0$

Para $i = 2, \ldots, n$

$m_{i1} = \dfrac{a_{i1}}{a_{11}}$

Para $j = 2, \ldots, n$

$a_{ij} = a_{ij} - m_{i1} a_{1j}$

Fim

$b_i - m_{i1} b_1$

Fim

Para $k = 0, 1, 2, \ldots$

$s1 = 0$

Para $i = 2, 3, \ldots, n$

$s = 0$

Para $j = 2, \ldots, (i-1), (i+1), \ldots, n$

$s = s + a_{ij} x_j^{(k)}$

Fim

$x_i^{(k+1)} = (b_i - s)/a_{ii}$

$s1 = s1 + a_{1i} x_i^{(k)}$

Fim

$x_1^{k+1} = (b_1 - s1)/a_{11}$

Fim

4.6.2 Método de Gauss-Seidel Híbrido

Através dos sistemas (SE) e (SE1) dados na seção 4.6, definimos o método Gauss-Seidel híbrido combinado com uma iteração do método de eliminação Gauss.

Assim, o método Gauss-Seidel híbrido para o sistema (SE1) da seção 4.6 é definido por

$$\begin{cases} x_1^{(k+1)} = [b_1 - a_{12}x_2^{(k)} - \cdots - (a_{2n} - m_{21}a_{1n})x_n^{(k)}]/(a_{22} - m_{21}a_{12}) \\ x_2^{(k+1)} = [(b_2 - m_{21}b_1) - (a_{23} - m_{21}a_{13})x_3^{(k)} - \cdots - (a_{2n} - m_{21}a_{1n})x_n^{(k)}]/(a_{22} - m_{21}a_{12}) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_n^{(k+1)} = [(b_n - m_{n1}b_1) - (a_{n2} - m_{n1}a_{12})x_2^{(k+1)} - \cdots - (a_{n,n-1} - m_{n1}a_{n,n-1})x_{n-1}^{(k+1)}]/(a_{nn} - m_{n1}a_{1n}) \end{cases}$$

cuja forma matricial é dado por

$$x^{(k+1)} = -(\bar{D} + \bar{L})^{-1}x^{(k)} + (\bar{D} + \bar{L})^{-1}\bar{b} \tag{4.25}$$

em que

$$\bar{D} = \begin{pmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} - m_{21}a_{12} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & a_{nn} - m_{n1}a_{1n} \end{pmatrix},$$

$$\bar{L} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & a_{32} - m_{31}a_{12} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & a_{n2} - m_{n2}a_{12} & a_{n3} - m_{n1}a_{13} & 0 \end{pmatrix},$$

$$\bar{U} = \begin{pmatrix} 0 & a_{12} & a_{13} & a_{1n} \\ 0 & 0 & a_{23} - m_{21}a_{13} & a_{2n} - m_{21}a_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & a_{n-1,n} - m_{n-1,1}a_{1n} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

e

$$\bar{b} = \begin{pmatrix} b_1 \\ b_2 - m_{21}b_1 \\ \vdots \\ b_n - m_{n1}b_1 \end{pmatrix}.$$

Denotemos por

$$M_{\bar{gs}} = -(\bar{D} + \bar{L})^{-1}\bar{U} \tag{4.26}$$

a matriz de Gauss-Seidel para esta nova configuração.

Dos resultados de convergência do método de Gauss-Seidel clássico, tem-se que o método híbrido (4.25) convergirá quando

- $\rho(M_{\bar{gs}}) < 1$ onde $M_{\bar{gs}}$ é a matriz de Gauss-Seidel (4.26). Conforme teorema (4.1).
- A matriz H dos coeficientes do sistema (SE1) é positiva definida. Teorema (4.3).
- A matriz H dos coeficientes do sistema (SE1) é tridiagonalmente dominante. Teorema (4.5).
- A matriz H dos coeficientes do sistema (SE1) é positiva e tridiagonal. Teorema (4.3).

Algoritmo 4.5 *Método de Gauss-Seidel Híbrido*

Dado $x^{(0)} \in \mathbb{R}^n$

Supondo $a_{ii} \neq 0$

Para $i = 2, \ldots, n$

$m_{i1} = \dfrac{a_{i1}}{a_{11}}$

Para $j = 2, \ldots, n$

$a_{ij} = a_{ij} - m_{i1}a_{1j}$

Fim

$b_i - m_{i1}b_1$

Fim

Para $k = 0, 1, 2, \ldots$

$s1 = 0$

Para $i = 2, 3, \ldots, n$

$s = 0$

Para $j = 2, \ldots, (i-1)$

$s = s + a_{ij} x_j^{(k)}$

Fim

Para $j = i+1, \ldots, n$

$s = s + a_{ij} x_j^{(k)}$

Fim

$x_i^{(k+1)} = (b_i - s)/a_{ii}$

$s1 = s1 + a_{ij} x_1^{(k)}$

Fim

$x_1^{k+1} = (b_1 - s1)/a_{11}$

Fim

4.6.3 O Método SOR Híbrido

Considere os sistemas lineares (SE) e (SE1) dados na seção 4.6. Definiremos, nesta seção, o método híbrido que combina o método SOR, dado na seção 4.6, com a primeira iteração do método de Eliminação Gaussiana, dado da seção 3.3 do capítulo 3.

Desta forma, definimos o método SOR híbrido como

$$(\text{SH}) \begin{cases} x_i^{(k+1)} = \dfrac{[(b_i - m_{i1}b_1) - \sum_{j=1}^{i-1}(a_{i2} - m_{i1}a_{12})x_j^{(k+1)} - \sum_{j=i+1}^{n}(a_{i2} - m_{i1}a_{12})x_j^{(k)}]}{(a_{ii} - m_{i1}a_{1i})} \\ \\ x_i^{(k+1)} = \omega x_i^{(k+1)} - (1-\omega)x_i^{(k)}, \quad i = 2, ..., n \text{ e } \omega \in I\!R \end{cases}$$

Para $\omega = 1$ (SH) é o método de Gauss-Seidel híbrido dado na seção 4.6.2. Na forma matricial o método SOR híbrido tem a forma

$$x^{(k+1)} = (\bar{L} + \omega\bar{D})^{-1}((1-\omega)\bar{D} - \omega\bar{U})x^{(k)} + \omega(\bar{L} + \omega\bar{D})^{-1}\bar{b} \qquad (4.27)$$

onde $\bar{D}$, $\bar{L}$ e $\bar{U}$ e $\bar{b}$ são as mesmas matrizes obtidas anteriormente pelo método de Gauss-Seidel híbrido dado na seção 4.6.2.

Denotaremos por $M_{S\bar{O}R}$ a matriz do método SOR híbrido e esta terá a forma

$$M_{S\bar{O}R} = (\bar{L} + \omega\bar{D})^{-1}((1-\omega)\bar{D} - \omega U) \tag{4.28}$$

Segundo os resultados de convergência apresentados anteriormente, neste capítulo, o método SOR híbrido convergirá quando:

- $\rho(M_{S\bar{O}R}) < 1$ onde $\rho(M_{S\bar{O}R})$ é o raio espectral da matriz $M_{S\bar{O}R}$ dada na relação (4.28).Teorema(4.1).
- A matriz H dos coeficientes do sistema linear (SE1) é tridiagonalmente dominante, conforme teorema (4.5).

Algoritmo 4.6 *Método SOR Híbrido*

Dado $x^{(0)} \in \mathbb{R}^n$ *e supondo* $a_{ii} \neq 0$

Para $i = 2, \ldots, n$

$m_{i1} = \dfrac{a_{i1}}{a_{11}}$

Para $j = 2, \ldots, n$

$a_{ij} = a_{ij} - m_{i1} a_{1j}$

Fim

$b_i - m_{i1} b_1$

Fim

Para $k = 0, 1, 2, \ldots$

$s1 = 0$

Para $i = 2, 3, \ldots, n$

$s = 0$

Para $j = 2, \ldots, (i-1)$

$s = s + a_{ij} x_j^{(k)}$

Fim

Para $j = i+1, \ldots, n$

$s = s + a_{ij} x_j^{(k)}$

Fim

$s = (b_i - s)/a_{ii}$

$x_i^{(k+1)} = x_i^{(k)} + \omega(s - x_i^{(k)})$

Fim

Cheque convergência, continue se necessário

Fim

No capítulo seguinte serão apresentadas implementações dos métodos iterativos descritos no trabalho. Além disso, serão feitos vários testes numéricos com o objetivo de comparar o desempenho dos algoritmos propostos, principalmente com respeito ao número de iterações, tempo de processamento e outros fatores importantes relacionados à convergência.

Capítulo 5

Resultados Numéricos

5.1 Introdução

Neste capítulo apresentaremos resultados numéricos referentes aos métodos iterativos desenvolvidos nos capítulos precedentes. Estes métodos foram implementados em Matlab versão 6.1.0.450 (R12.1) para o sistema operacinal linux, rodando em um PC. Os algoritmos implementados são: algoritmo de Jacobi 4.1, algoritmo de Gauss-Seidel 4.2, algoritmo SOR 4.3, algoritmo Jacobi híbrido 4.4, algoritmo Gauss-Seidel híbrido 4.5 e algoritmo SOR híbrido 4.6.

Os resultados dos testes dos algoritmos programados serão apresentados em forma de tabelas e, o principal objetivo dos testes numéricos é avaliar a performance dos algoritmos propostos. Estaremos sempre comparando o algoritmo tradicional com o respectivo algoritmo híbrido proposto.

Para testar os algoritmos utilizaremos um conjunto de matrizes de testes descritas em The Matrix Computational Toolbox for Matlab, ver refência [8]. As matrizes deste toolbox estão descritas na tabela 5.1 a seguir. Nesta tabela os números de 1 a 52 identificam as matrizes de testes. Desta forma, o número 1 se refere à matriz cauchy, o número 2 se refere à matrix chebspec e assim sucessivamente até o número 52 que se refere à matriz vand. As matrizes de 1 a 46 são descritas, também, no Matlab.

Por questões didáticas, em todas as tabelas que serão descritos os resultados dos algoritmos implementados, as matrizes serão identificadas por seus números, que são dados na tabela 5.1 a seguir:

1	cauchy	12	frank	23	lesp	34	randsvd	45	rand
2	chebspec	13	gearmat	24	lotkin	35	redheff	46	randn
3	chebvand	14	grcar	25	minij	36	riemann	47	augment
4	chow	15	invhess	26	moler	37	ris	48	gfpp
5	circul	16	invol	27	orthog	38	smoke	49	magic
6	clement	17	ipjfact	28	parter	39	toeppd	50	makejcf
7	condex	18	jordbloc	29	pei	40	triw	51	rschur
8	cycol	19	kahan	30	prolate	41	hilb	52	vand
9	dramadah	20	kms	31	randcolu	42	invhilb		
10	fiedler	21	krylov	32	randcorr	43	magic		
11	forsythe	22	lehmer	33	rando	44	pascal		

Tabela 5.1: Matrizes de testes para os algoritmos

Destas 52 matrizes, utilizamos 32 para os testes, pois algumas são complexas e outras são geradas aleatoriamente pelo Matlab, o que significa que a cada vez que são geradas elas são diferentes, fugindo à estratégia de metodologia dos testes.

Outra característica destas matrizes é que o usuário pode escolher a dimensão. Tanto em Matlab, (ver gallery, utilizando o comando help do Matlab) quanto em [8] é possível fornecer a dimensão da matriz que será utilizada no teste. Por isso, primeiramente faremos uma rodada de testes com as 32 matrizes de dimensão 3, para todos os 6 algoritmos descritos no início desta seção. Da mesma forma faremos uma segunda rodada de testes com as 32 matrizes de dimensão 40.

Estamos nos referindo às matrizes de teste, mas na verdade estamos resolvendo sistemas lineares na forma $Ax = b$, em que A é a matriz dos coeficientes descritas na tabela 5.1, x é o vetor das incógnitas e b é fixado e vale $b = (6, 2, 4)$ para a primeira rodada de testes e $b = (1, 2, 3, \ldots, 40)$ para a segunda rodada de testes.

Fixaremos, para todos os testes, um número máximo de 300 iterações. Portanto, nenhum teste excederá a este número. Além disso, se o algoritmo atingiu o número máximo de iterações significa que não alcançou a solução do sistema com precisão pré-estabelecida pelo valor de 1e-6 (10^-6), a menos que tenha atingido a convergência exatamente em 300 iterações.

5.2 Apresentação de Resultados Numéricos para Matrizes Quadradas de Dimensão 3

Nesta seção, apresentaremos resultados numéricos para a primeira rodada de testes, em que as matrizes utilizadas são todas quadradas de dimensão 3.

A seguir será apresentada a tabela que exibe os resultados dos testes do método de Jacobi 4.1 e Jacobi híbrido 4.4. Os métodos são identificados pela ordem em que aparecem. Assim, na primeira coluna aparecem os números das matrizes, sempre repetidos em dupla. O primeiro é referente ao método de Jacobi e o segundo é referente ao método de Jacobi híbrido, o mesmo ocorrendo com as demais tabelas que serão apresentadas. As colunas da tabela são identificadas na primeira linha pelos seguintes nomes:

- **matriz** - número das matrizes de testes descritas anteriormente na tabela 5.1.
- **iterações** - número de iterações executadas pelos programas dos respectivos métodos.
- **erro** - erro relativo dos dois últimos pontos gerados na execução dos programas. O erro relativo é utilizado como um critério de parada, juntamente com o número máximo de iterações.
- **cputime** - exibe o tempo gasto pelo programa (em segundos), para executar o laço principal (processo iterativo).
- **pos def** - identifica se a matriz é ou não positiva definida. Se o número é zero a matriz é positiva definida, caso contrário não é.
- **condição** - exibe o número de condição da matriz.

Tabela 5.2: Resultados numéricos

matriz	iterações	erro	cputime	pos def	condição
1	300	2.8599	0.2	0	1353.3
1	300	5.2185e-05	0.16	0	1353.3
2	18	5.311e-07	0.01	2	1.4296e+17
2	300	0.0040895	0.16	2	1.4296e+17
3	300	2.0393	0.21	2	12.488
3	40	9.6522e-07	0.03	2	12.488
5	300	6	0.21	2	3.4641
5	25	6.7156e-07	0.01	2	3.4641
7	2	6.1698e-15	0	0	1
7	2	1.8916e-15	0.01	0	1
12	300	2.2017	0.2	0	27.58
12	39	5.6138e-07	0.02	0	27.58

continua ...

Tabela 5.2: Resultados numéricos (continuação)

matriz	iterações	erro	cputime	pos def	condição
14	300	1.6542	0.2	2	1.4142
14	300	1.7638	0.16	2	1.4142
15	300	1.2723	0.2	3	3.0765
15	21	7.5736e-07	0.02	3	3.0765
16	300	2.7229	0.2	1	2396.6
16	300	1.3044e-05	0.16	1	2396.6
17	300	2.6134	0.2	0	4679.6
17	247	9.363e-07	0.13	0	4679.6
18	4	0	0.01	2	4.0489
18	4	0	0	2	4.0489
19	4	0	0.01	0	1.9198
19	4	0	0	0	1.9198
20	77	9.6618e-07	0.05	0	4.5292
20	17	9.4061e-07	0.01	0	4.5292
22	300	1.8969	0.2	0	6.6633
22	27	9.9017e-07	0.02	0	6.6633
23	11	7.1316e-07	0.01	1	2.3364
23	9	6.7316e-07	0.01	1	2.3364
24	300	3.3331	0.2	2	482.92
24	300	3.3276e-06	0.16	2	482.92
25	300	2.4051	0.2	0	16.394
25	42	7.5395e-07	0.02	0	16.394
26	126	9.0938e-07	0.08	0	29.284
26	38	8.9463e-07	0.02	0	29.284
27	300	NaN	0.21	2	1
27	300	NaN	0.17	2	1
28	104	9.2282e-07	0.07	3	1.9552
28	23	5.915e-07	0.02	3	1.9552
29	20	8.133e-07	0.01	0	4
29	14	8.551e-07	0.01	0	4
30	116	9.4122e-07	0.08	0	19.063
30	64	9.5374e-07	0.03	0	19.063
36	25	7.6701e-07	0.01	0	8.1936
36	4	0	0	0	8.1936
37	300	4.7675	0.2	3	1.9552
37	300	1.0206	0.16	3	1.9552
40	4	0	0	2	5.4115
40	4	0	0	2	5.4115
41	300	2.7229	0.21	0	524.06
41	300	2.0872e-06	0.16	0	524.06
42	300	2.7229	0.2	0	524.06
42	300	3.1245e-05	0.16	0	524.06
43	300	3.9464	0.21	3	4.3301

continua ...

Tabela 5.2: Resultados numéricos (continuação)

matriz	iterações	erro	cputime	pos def	condição
43	300	2.4946	0.16	3	4.3301
44	300	2.338	0.2	0	61.984
44	119	8.8226e-07	0.06	0	61.984
48	300	1.8256	0.2	3	1.4142
48	300	1.1574	0.15	3	1.4142
51	300	10.05	0.2	1	5.206
51	4	0	0	1	5.206
52	40	9.9215e-07	0.03	2	15.1
52	40	9.9215e-07	0.03	2	15.1

Resultados dos métodos de Jacobi e Jacobi híbrido

Para um melhor detalhadmento da tabela 5.2, observe que o número 1 da linha 2 refere-se à matriz de cauchy (ver tabela 5.1). O número 300, refere-se ao número de iterações realizadas pelo método de Jacobi (atingiu o número máximo de iterações sem alcançar convergência), os próximos números são, respectivamente, o erro cometido, o tempo de cpu (para o laço principal), a positividade da matriz e finalmente o número de condição da matriz, para o método de Jacobi. O mesmo ocorrendo com a linha 3 que é referente ao método de Jacobi híbrido.

Alguns dados serão comuns à maioria das tabelas apresentadas. Por exemplo, na coluna do erro aparece a palavra NaN que é definido pelo software Matlab como: Not a Number, isso significa que NaN é o infinito ou alguma indeterminação da forma $0.0/0.0$ e $\infty - \infty$ do Matlab. Pelo acompanhamento que fizemos o erro relativo crescia indefinidamente. Portanto, nessa situação, o respectivo método não convergiu e não convergirá se for aumentado o número de iterações. Em outras situações estes erros são da ordem de 1e-1, 1e-2, 1e-3 e assim por diante, significando que o método poderia convergir se fosse aumentado o número iterações.

Para uma análise comparativa dos resultados dos testes dos métodos de Jacobi e Jacobi híbrido, da tabela 5.2, destacamos na tabela 5.3 dada a seguir, o número de vezes que cada um atingiu convergência, ou seja, alcançou a solução com precisão de 1e-6 em menos de 300 iterações, além da soma do tempo de cpu. Assim, a coluna **menos-iterações** fornece o número de vezes que o método atingiu convergência e a coluna **tempo-cpu** é a soma do tempo de processamento (do laço principal) da coluna cputime da tabela 5.2.

Tabela 5.3: Comparações

	menos-iterações	tempo-cpu
jacobi	13	4.22
jacobi-híb	21	2.21

Comparação entre os métodos de Jacobi e Jacobi híbrido - dimensão 3

Observe como Jacobi híbrido foi muito superior. Dos 32 problemas testados o método conseguiu resolver 21, e o tempo de cpu foi praticamente a metade do método de Jacobi clássico. Na totalidade dos sistemas que foram resolvidos pelos dois métodos Jacobi híbrido foi mais rápido em número de iterações, assim como em tempo de processamento. A maior diferença encontra-se na matriz 51 em que Jacobi híbrido resolveu o sistema em 4 iterações e Jacobi clássico não conseguiu resolver no número máximo de iterações.

Outro detalhe a ser observado na tabela 5.2 é com relação ao erro relativo. Olhando para os problemas que os métodos não alcançaram solução (300 iterações), verifica-se que o método de Jacobi híbrido, na maioria das matrizes testadas, chega a um erro relativo menor. Pela característica do erro é muito provável que o método de Jacobi híbrido chegará mais rápido à solução, se for aumentado o número de iterações.

O melhor desempenho do método de Jacobi híbrido em relação ao método de Jacobi clássico, também será percebido nos demais métodos iterativos desenvolvidos, como será visto nas próximas tabelas apresentadas.

A seguir é apresentada a tabela de resultados dos métodos de Gauss-Seidel e Gauss-Seidel híbrido, para matrizes quadradas de dimensão 3. A explicação da tabela é a mesma dos métodos de Jacobi e Jacobi híbrido apresentados na tabela 5.2.

Tabela 5.4: Resultados numéricos

matriz	iterações	erro	cputime	pos def	condição
1	300	0.0012626	0.22	0	1353.3
1	261	9.8988e-07	0.14	0	1353.3
2	300	NaN	0.22	2	1.4296e+17
2	300	0.0033478	0.17	2	1.4296e+17
3	300	3.7321	0.21	2	12.488
3	21	9.698e-07	0.01	2	12.488
5	300	9	0.22	2	3.4641
5	12	8.6189e-07	0.01	2	3.4641
7	2	1.8578e-15	0	0	1

continua ...

Tabela 5.4: Resultados numéricos (continuação)

matriz	iterações	erro	cputime	pos def	condição
7	2	8.3081e-16	0.01	0	1
12	67	9.396e-07	0.05	0	27.58
12	21	5.0991e-07	0.01	0	27.58
14	300	4	0.22	2	1.4142
14	300	1.7085	0.17	2	1.4142
15	35	8.3891e-07	0.02	3	3.0765
15	10	9.0866e-07	0	3	3.0765
16	300	5.615e-05	0.22	1	2396.6
16	171	9.7352e-07	0.1	1	2396.6
17	283	9.7894e-07	0.2	0	4679.6
17	84	9.5496e-07	0.05	0	4679.6
18	4	0	0.01	2	4.0489
18	3	0	0	2	4.0489
19	4	0	0	0	1.9198
19	3	0	0	0	1.9198
20	13	6.1573e-07	0.01	0	4.5292
20	10	4.1724e-07	0	0	4.5292
22	18	8.0968e-07	0.01	0	6.6633
22	15	5.8195e-07	0.01	0	6.6633
23	7	6.2425e-07	0.01	1	2.3364
23	6	3.0181e-08	0.01	1	2.3364
24	300	0.1843	0.22	2	482.92
24	173	9.9377e-07	0.1	2	482.92
25	30	9.6126e-07	0.02	0	16.394
25	18	7.8972e-07	0.01	0	16.394
26	67	9.8454e-07	0.05	0	29.284
26	21	5.1268e-07	0.01	0	29.284
27	300	NaN	0.22	2	1
27	300	NaN	0.18	2	1
28	17	3.5293e-07	0.01	3	1.9552
28	10	7.1021e-07	0	3	1.9552
29	14	7.5346e-07	0.02	0	4
29	2	0	0	0	4
30	59	9.9488e-07	0.04	0	19.063
30	35	7.3578e-07	0.02	0	19.063
36	20	7.4036e-07	0.01	0	8.1936
36	2	0	0	0	8.1936
37	300	NaN	0.22	3	1.9552
37	300	4	0.17	3	1.9552
40	4	0	0	2	5.4115
40	3	0	0	2	5.4115
41	300	5.8987e-05	0.22	0	524.06
41	173	9.6394e-07	0.1	0	524.06

continua ...

Tabela 5.4: Resultados numéricos (continuação)

matriz	iterações	erro	cputime	pos def	condição
42	300	3.7957e-05	0.22	0	524.06
42	164	9.9986e-07	0.1	0	524.06
43	300	4.7895	0.22	3	4.3301
43	300	9.7297	0.17	3	4.3301
44	94	9.4054e-07	0.07	0	61.984
44	54	9.6215e-07	0.03	0	61.984
48	300	4	0.22	3	1.4142
48	300	1.8074	0.17	3	1.4142
51	300	NaN	0.22	1	5.206
51	3	0	0	1	5.206
52	22	6.7768e-07	0.02	2	15.1
52	21	9.493e-07	0.01	2	15.1

Resultados dos métodos de Gauss-Seidel e Gauss-Seidel híbrido

As observações feitas para a tabela 5.2 são aplicáveis também para tabela 5.4 e a seguir apresentaremos uma tabela comparando os dois métodos. Essa tabela é idêntica à tabela 5.3 só que agora os valores são referentes aos métodos de Gauss-Seidel e Gauss-Seidel híbrido.

Tabela 5.5: Comparações

	menos-iterações	tempo-cpu
Gauss-Seidel	18	3.62
Gauss-Seidel-híb	26	1.76

Comparação entre os métodos Gauss-Seidel e Gauss-Seidel híbrido - dimensão 3

Observe, como no método de Jacobi híbrido, que o método de Gauss-Seidel híbrido é muito superior, tanto em relação ao número de iterações quanto em relação ao tempo de processamento.

A próxima tabela é referente aos resultados dos métodos SOR e SOR híbrido e valem as mesmas explicações das tabelas 5.2 do método de Jacobi e Jacobi híbrido e da tabela 5.4 do método de Gauss-Seidel e Gauss-Seidel híbrido.

Tabela 5.6: Resultados numéricos

matriz	iterações	erro	cputime	pos def	condição
1	300	8.1553e-05	0.22	0	1353.3
1	89	9.9717e-07	0.06	0	1353.3
2	300	NaN	0.23	2	1.4296e+17
2	300	0.0033472	0.19	2	1.4296e+17
3	300	7.5734	0.23	2	12.488
3	22	7.9428e-07	0.01	2	12.488
5	300	NaN	0.23	2	3.4641
5	21	7.751e-07	0.01	2	3.4641
7	22	5.6546e-07	0.02	0	1
7	22	5.6546e-07	0.02	0	1
12	74	8.8167e-07	0.06	0	27.58
12	20	9.1349e-07	0.01	0	27.58
14	300	10.045	0.23	2	1.4142
14	300	4.1712	0.18	2	1.4142
15	300	3	0.22	3	3.0765
15	300	2.3815	0.18	3	3.0765
16	166	9.424e-07	0.12	1	2396.6
16	54	8.537e-07	0.03	1	2396.6
17	77	9.9777e-07	0.06	0	4679.6
17	25	8.612e-07	0.02	0	4679.6
18	32	9.5738e-07	0.03	2	4.0489
18	31	9.5711e-07	0.02	2	4.0489
19	29	7.543e-07	0.02	0	1.9198
19	28	7.5112e-07	0.01	0	1.9198
20	24	4.9739e-07	0.02	0	4.5292
20	21	7.3003e-07	0.02	0	4.5292
22	25	6.6501e-07	0.01	0	6.6633
22	21	6.8046e-07	0.01	0	6.6633
23	26	3.9141e-07	0.02	1	2.3364
23	23	8.1613e-07	0.02	1	2.3364
24	56	6.1274e-07	0.04	2	482.92
24	56	9.2408e-07	0.03	2	482.92
25	35	7.6709e-07	0.03	0	16.394
25	21	6.2161e-07	0.02	0	16.394
26	19	4.7621e-07	0.02	0	29.284
26	21	9.1316e-07	0.02	0	29.284
27	300	NaN	0.23	2	1
27	300	NaN	0.19	2	1
28	300	2.8376	0.22	3	1.9552
28	300	2.5931	0.18	3	1.9552
29	28	7.4756e-07	0.02	0	4
29	21	7.3124e-07	0.02	0	4
30	21	2.2206e-07	0.01	0	19.063

continua ...

Tabela 5.6: Resultados numéricos (continuação)

matriz	iterações	erro	cputime	pos def	condição
30	21	9.9265e-07	0.02	0	19.063
36	300	2.494	0.23	0	8.1936
36	30	7.53e-07	0.02	0	8.1936
37	300	NaN	0.23	3	1.9552
37	300	8.7176	0.18	3	1.9552
40	31	8.999e-07	0.03	2	5.4115
40	30	8.9892e-07	0.02	2	5.4115
41	171	9.6677e-07	0.13	0	524.06
41	56	8.5967e-07	0.04	0	524.06
42	162	9.4595e-07	0.12	0	524.06
42	52	9.1158e-07	0.03	0	524.06
43	300	9.0711	0.22	3	4.3301
43	300	NaN	0.18	3	4.3301
44	27	3.2351e-07	0.02	0	61.984
44	20	7.8083e-07	0.02	0	61.984
48	300	10.045	0.23	3	1.4142
48	300	4.1712	0.19	3	1.4142
51	300	NaN	0.23	1	5.206
51	32	8.7945e-07	0.02	1	5.206
52	23	3.218e-07	0.02	2	15.1
52	22	7.8774e-07	0.01	2	15.1

Resultados dos métodos SOR e SOR híbrido

Na tabela a seguir comparamos o desempenho entre o método SOR e SOR híbrido com relação ao número de iterações e tempo de processamento.

Tabela 5.7: Comparações

	menos-iterações	tempo-cpu
SOR	19	3.75
SOR-híb	24	1.98

Comparação entre os métodos SOR e SOR híbrido - dimensão 3

Observe como o método SOR híbrido é superior ao método SOR clássico. Cabe destacar que estes métodos são fortemente dependentes do parâmetro de relaxão ω (ver algoritmo 4.6). Para os testes apresentados este parâmetro foi fixado em $\omega = 1.5$ e pelos teoremas apresentados no capítulo 4 estes métodos são convergentes somente para valores particulares do parâmetro ω. Pode-se fazer um estudo adicional procurando-se otimizar este parâmetro. Veja, por exemplo, teorema 4.6.

5.3 Apresentação de Resultados Numéricos para Matrizes Quadradas de Dimensão 40

Nesta seção repetiremos os testes da seção 5.2 só que agora para matrizes quadradas de dimensão 40.

A tabela a seguir é referente ao método de Jacobi e Jacobi híbrido.

Tabela 5.8: Resultados numéricos

matriz	iterações	erro	cputime	pos def	condição
1	300	NaN	21.04	14	7.7903e+18
1	300	NaN	20.66	14	7.7903e+18
2	300	NaN	21.11	2	4.7851e+16
2	300	NaN	20.62	2	4.7851e+16
3	300	NaN	21.15	2	2.6218e+17
3	300	NaN	20.66	2	2.6218e+17
5	300	NaN	21.2	2	41
5	300	NaN	20.68	2	41
7	300	0.00045071	21.08	0	101
7	300	0.00045071	20.5	0	101
12	300	3.8872	21.02	0	5.9094e+17
12	300	3.8836	20.64	0	5.9094e+17
14	300	2.2032	20.99	2	3.4609
14	300	2.4259	20.61	2	3.4609
15	300	NaN	21.13	3	310.27
15	300	NaN	20.67	3	310.27
16	300	NaN	21.18	1	6.2647e+28
16	300	NaN	20.59	1	6.2647e+28
17	300	NaN	21.13	21	6.8616e+96
17	300	NaN	20.71	21	6.8616e+96
18	41	0	2.85	2	51.531
18	41	0	2.83	2	51.531
19	26	7.2203e-07	1.83	4	7.6459e+06
19	26	7.2203e-07	1.8	4	7.6459e+06
20	300	2.9683	21.03	0	8.8929
20	300	2.9661	20.49	0	8.8929
22	300	NaN	21.11	0	1570.8
22	300	NaN	20.67	0	1570.8
23	21	5.0465e-07	1.47	1	26.619
23	21	5.0202e-07	1.45	1	26.619
24	300	NaN	21.09	2	6.6859e+19
24	300	NaN	20.73	2	6.6859e+19
25	300	NaN	21.09	0	2655.4

continua ...

Tabela 5.8: Resultados numéricos (continuação)

matriz	iterações	erro	cputime	pos def	condição
25	300	NaN	20.59	0	2655.4
26	300	NaN	21.05	0	7.3496e+17
26	300	NaN	20.6	0	7.3496e+17
27	300	NaN	21.1	2	1
27	300	NaN	20.62	2	1
28	300	1.5498	21.05	3	2.9553
28	300	1.5314	20.55	3	2.9553
29	300	NaN	21.1	0	41
29	300	NaN	20.59	0	41
30	300	0.49744	21.02	25	4.7989e+16
30	300	2.2142	20.58	25	4.7989e+16
36	300	0.39989	21.04	4	240.94
36	300	0.19138	20.62	4	240.94
37	300	NaN	21.16	8	2.9553
37	300	NaN	20.63	8	2.9553
40	35	9.1449e-07	2.44	2	8.9989e+12
40	35	9.1449e-07	2.4	2	8.9989e+12
41	300	NaN	21.02	14	1.1635e+19
41	300	NaN	20.55	14	1.1635e+19
42	300	NaN	21.07	14	2.9966e+45
42	300	NaN	20.66	14	2.9966e+45
43	300	NaN	21.1	3	2.9085e+19
43	300	NaN	20.64	3	2.9085e+19
44	300	NaN	21.09	33	8.3448e+28
44	300	NaN	20.55	33	8.3448e+28
48	300	NaN	20.96	40	17.81
48	300	NaN	20.52	40	17.81
51	300	10.05	21.09	1	55.898
51	300	5.0962	20.68	1	55.898
52	300	NaN	21.11	2	6.3519e+18
52	300	NaN	20.56	2	6.3519e+18

Resultados dos métodos Jacobi e Jacobi híbrido

Para matrizes de dimensão mais alta verifica-se que o número de problemas resolvidos, para ambos os métodos, com a precisão fixada em 1e-6 e número máximo de 300 itereções é bem menor, como destacado na tabela 5.9 apresentada a seguir.

Comparação entre os métodos de Jacobi e Jacobi híbrido - dimensão 40

Observe que mesmo para dimensão mais alta o método de Jacobi híbrido apresenta um melhor desempenho. Ambos resolveram 4 problemas, no entanto, o tempo de processamento para Jacobi híbrido foi inferior ao tempo de Jacobi clássico.

Tabela 5.9: Comparações

	menos-iterações	tempo-cpu
jacobi	4	598.90
jacobi-híb	4	585.65

A tabela a seguir é refente aos resultados dos métodos de Gauss-Seidel e Gauss-Seidel híbrido, para matrizes de dimensão 40.

Tabela 5.10: Resultados numéricos

matriz	iterações	erro	cputime	pos def	condição
1	300	0.0032154	19.77	14	7.7903e+18
1	300	0.0035858	19.26	14	7.7903e+18
2	300	NaN	19.87	2	4.7851e+16
2	300	NaN	19.4	2	4.7851e+16
3	300	NaN	19.95	2	2.6218e+17
3	300	NaN	19.37	2	2.6218e+17
5	300	NaN	19.94	2	41
5	300	0.12502	19.33	2	41
7	300	3.7262e-05	19.73	0	101
7	300	3.7262e-05	19.23	0	101
12	300	0.08092	19.81	0	5.9094e+17
12	300	0.10222	19.24	0	5.9094e+17
14	300	NaN	19.78	2	3.4609
14	300	NaN	19.25	2	3.4609
15	300	0.0047108	19.74	3	310.27
15	191	9.9958e-07	12.24	3	310.27
16	300	0.0021962	19.74	1	6.2647e+28
16	300	0.0019606	19.22	1	6.2647e+28
17	300	0.0014681	19.71	21	6.8616e+96
17	300	0.0043082	19.28	21	6.8616e+96
18	41	0	2.7	2	51.531
18	40	0	2.57	2	51.531
19	26	7.2203e-07	1.7	4	7.6459e+06
19	26	2.902e-07	1.67	4	7.6459e+06
20	20	5.8046e-07	1.31	0	8.8929
20	20	6.17e-07	1.29	0	8.8929
22	300	7.6687e-06	19.75	0	1570.8
22	300	7.6251e-06	19.23	0	1570.8
23	21	4.8517e-07	1.39	1	26.619
23	21	4.8205e-07	1.36	1	26.619
24	300	0.019893	19.72	2	6.6859e+19
24	300	0.0020777	19.23	2	6.6859e+19

continua ...

Tabela 5.10: Resultados numéricos (continuação)

matriz	iteraçoes	erro	cputime	pos def	condição
25	300	0.0003321	19.85	0	2655.4
25	300	0.00029061	19.26	0	2655.4
26	300	0.0033272	19.75	0	7.3496e+17
26	300	0.0033306	19.26	0	7.3496e+17
27	300	NaN	19.93	2	1
27	300	NaN	19.33	2	1
28	131	9.6971e-07	8.57	3	2.9553
28	127	8.2274e-07	7.92	3	2.9553
29	300	0.00014441	19.36	0	41
29	196	9.8643e-07	12.11	0	41
30	300	0.0021426	19.17	25	4.7989e+16
30	300	0.0020737	18.6	25	4.7989e+16
36	300	0.53039	19.16	4	240.94
36	300	1.5256	18.65	4	240.94
37	300	NaN	19.24	8	2.9553
37	300	NaN	18.75	8	2.9553
40	35	9.1449e-07	2.22	2	8.9989e+12
40	35	1.7453e-07	2.18	2	8.9989e+12
41	300	0.0028716	19.18	14	1.1635e+19
41	300	0.0024483	18.68	14	1.1635e+19
42	300	0.078642	19.14	14	2.9966e+45
42	300	0.41048	18.6	14	2.9966e+45
43	300	NaN	19.28	3	2.9085e+19
43	300	NaN	18.75	3	2.9085e+19
44	300	0.014047	19.15	33	8.3448e+28
44	300	0.011267	18.66	33	8.3448e+28
48	300	NaN	19.28	40	17.81
48	300	NaN	18.79	40	17.81
51	300	NaN	19.21	1	55.898
51	300	NaN	18.67	1	55.898
52	300	0.010242	19.11	2	6.3519e+18
52	300	0.010243	18.63	2	6.3519e+18

Resultados dos métodos Gauss-Seidel e Gauss-Seidel híbrido

Aqui valem a mesmas observações feitas anteriormente para os métodos de Jacobi e Jacobi híbrido. A diferença entre o número de problemas resolvidos cai em relação às matrizes quadradas de dimensão 3 para os métodos de Gauss-Seidel e Gauss-Seidel híbrido, conforme se verifica na tabela seguinte.

Comparação entre os métodos Gauss-Seidel e Gauss-Seidel híbrido - dimensão 40

Observe que Gauss-Seidel híbrido é superior ao método de Gauss-Seidel. Ele ganha no número de problemas resolvidos, assim como no tempo de processamento,

Tabela 5.11: Comparações

	menos-iterações	tempo-cpu
Gauss-Seidel	6	526.21
Gauss-Seidel-híb	8	498.01

sendo que o ganho é de aproximadamente 5.4% em relação ao método de Gauss-Seidel clássico.

A tabela dada a seguir é referente aos métodos SOR e SOR híbrido, para matrizes de dimensão 40.

Tabela 5.12: Resultados numéricos

matriz	iterações	erro	cputime	pos def	condição
1	300	0.0048849	19.24	14	7.7903e+18
1	300	0.0032282	18.71	14	7.7903e+18
2	300	NaN	19.43	2	4.7851e+16
2	300	NaN	18.91	2	4.7851e+16
3	300	NaN	19.32	2	2.6218e+17
3	300	NaN	18.87	2	2.6218e+17
5	300	NaN	19.38	2	41
5	300	0.19021	18.78	2	41
7	172	9.7116e-07	11.01	0	101
7	172	9.7116e-07	10.79	0	101
12	300	NaN	19.41	0	5.9094e+17
12	300	NaN	18.94	0	5.9094e+17
14	300	NaN	19.38	2	3.4609
14	300	NaN	18.82	2	3.4609
15	300	5.0231	19.34	3	310.27
15	300	4.7425	18.75	3	310.27
16	300	0.0019502	19.34	1	6.2647e+28
16	300	0.0018971	18.75	1	6.2647e+28
17	300	0.0047541	19.34	21	6.8616e+96
17	300	0.0097452	18.75	21	6.8616e+96
18	228	9.2762e-07	14.57	2	51.531
18	227	9.3148e-07	14.19	2	51.531
19	45	9.7415e-07	2.89	4	7.6459e+06
19	45	7.7335e-07	2.81	4	7.6459e+06
20	45	8.2896e-07	2.89	0	8.8929
20	45	8.4837e-07	2.81	0	8.8929
22	300	0.00042078	19.28	0	1570.8
22	300	0.00041858	18.73	0	1570.8
23	159	7.0925e-07	10.19	1	26.619

continua ...

Tabela 5.12: Resultados numéricos (continuação)

matriz	iterações	erro	cputime	pos def	condição
23	160	6.5646e-07	10	1	26.619
24	300	0.01061	19.28	2	6.6859e+19
24	300	0.0024073	18.72	2	6.6859e+19
25	300	0.0095483	19.25	0	2655.4
25	300	0.0087449	18.75	0	2655.4
26	300	0.0033383	19.33	0	7.3496e+17
26	300	0.0033379	18.81	0	7.3496e+17
27	300	NaN	19.47	2	1
27	300	NaN	18.84	2	1
28	300	3.4156	19.27	3	2.9553
28	300	3.4305	18.79	3	2.9553
29	300	0.012123	19.33	0	41
29	300	0.00033274	18.73	0	41
30	300	0.0016736	19.28	25	4.7989e+16
30	300	0.0016161	18.83	25	4.7989e+16
36	300	3.6695	19.23	4	240.94
36	300	8.3836	18.74	4	240.94
37	300	NaN	19.44	8	2.9553
37	300	NaN	18.9	8	2.9553
40	163	7.353e-07	10.41	2	8.9989e+12
40	162	7.438e-07	10.09	2	8.9989e+12
41	300	0.0027213	19.29	14	1.1635e+19
41	300	0.0023345	18.79	14	1.1635e+19
42	300	0.17944	19.24	14	2.9966e+45
42	300	0.20051	18.8	14	2.9966e+45
43	300	NaN	19.39	3	2.9085e+19
43	300	NaN	18.89	3	2.9085e+19
44	300	0.0047739	19.33	33	8.3448e+28
44	300	0.0036505	18.76	33	8.3448e+28
48	300	NaN	19.5	40	17.81
48	300	NaN	18.92	40	17.81
51	300	NaN	19.46	1	55.898
51	300	NaN	18.86	1	55.898
52	300	0.037361	19.32	2	6.3519e+18
52	300	0.037385	18.78	2	6.3519e+18

Resultados dos métodos SOR e SOR híbrido

Observe que os resultados são muito parecidos para os dois métodos, conforme destaca a tabela a seguir.

Comparação entre os métodos SOR e SOR híbrido - dimensão 40

Neste caso o ganho foi menos significativo em relação ao método de Gauss-Seidel híbrido, pois o número de problemas resolvidos pelos dois métodos SOR e SOR

Tabela 5.13: Comparações

	menos-iterações	tempo-cpu
SOR	6	554.83
SOR-híb	6	539.61

híbrido foi o mesmo. Quanto ao tempo de processamento, o método SOR híbrido foi cerca de 2.74% mais rápido que o método SOR clássico.

É conveniente destacar que os métodos SOR são dependentes de uma boa escolha do parâmetro de relaxação ω. Existem técnicas para determinar o parâmetro ω ótimo para casos de matrizes que tenham alguma estrutura, como é o caso de matrizes positivas definidas tridiagonais, conforme o teorema 4.6.

Para finalizar a parte numérica, apresentaremos uma última tabela com os três métodos híbridos propostos e analizaremos qual deles teve um melhor desempenho, com os mesmos parâmetros para todos, e $\omega = 1.5$ para o método SOR híbrido.

Comparação entre os três métodos híbridos para matrizes de dimensão 3

Tabela 5.14: Comparações

	menos-iterações	tempo-cpu
jacobi-híb	21	2.21
Gauss-Seidel-híb	26	1.76
SOR-híb	24	1.98

Comparação entre os métodos híbridos - dimensão 3

O melhor desempenho entre os três métodos foi alcançado pelo método de Gauss-Seidel híbrido, tanto em número de iterações como em tempo de cpu. O fato de Gauss-Seidel híbrido ser mais rápido que o método SOR híbrido, se deve, como justificado antes, ao fato de não termos utilizado nenhuma técnica para escolher o parâmetro de relaxação que simplesmente foi fixado em $\omega = 1.5$. Quanto a ser melhor que o método de Jacobi já era de se esperar, pela própria construção dos dois métodos.

Na próxima tabela repetiremos a comparação da tabela anterior, só que agora para matrizes de dimensão 40.

Comparação entre os métodos híbridos - dimensão 40

Novamente o método de Gauss-Seidel híbrido apresenta um melhor desempenho no número de problemas resolvidos, assim como em tempo de processamento.

Tabela 5.15: Comparações

	menos-iterações	tempo-cpu
jacobi-híb	4	585.65
Gauss-Seidel-híb	8	498.01
SOR-híb	6	539.61

5.4 Conclusões sobre os Resultados Numéricos

Através dos testes realizados e exibidos nas tabelas deste capítulo, vimos que os métodos propostos são competitivos. Para matrizes de dimensões baixas o desempenho dos métodos propostos é bem superior aos clássicos (ver tabelas, 5.3, 5.5, 5.7). Para matrizes de dimensões mais altas todos os três métodos, Jacobi híbrido, Gauss-Seidel híbrido e SOR híbrido foram superiores aos clássicos, principalmente em relação ao tempo de processamento.

Quando comparamos os métodos híbridos entre si verificamos que o método de Gauss-Seidel teve um melhor desempenho. Ele consegue resolver um maior número de problemas e também é mais rápido no tempo de processamento. Para o método SOR existe a necessidade de se trabalhar um pouco mais a questão do parâmetro de relaxação, pois este método é fortemente dependente deste parâmetro. Possivelmente alguns testes futuros poderão auxiliar em uma melhor escolha deste parâmetro. Nos testes que aqui realizamos simplesmente fixamos este parâmetro em $\omega = 1.5$.

Para matrizes de dimensão 40, os testes comparativos não foram muito diferentes, no sentido que Gauss-Seidel híbrido resolve mais problemas e e tem um menor tempo de processamento. Com o método SOR híbrido, como nos testes com matrizes de dimensão 3, há a necessidade de uma melhor escolha do parâmetro de relaxação.

Deve ser lembrado que as matrizes de testes utilizadas, na maioria, são muito mal condicionadas. Estas matrizes foram inventadas com o objetivo de testar métodos numéricos através da implementação de seus algoritmos e, portanto, têm a função de dificultar na resolução de sistemas lineares formados com estas matrizes como coeficientes.

Finalmente, concluímos que os métodos híbridos combinando uma iteração do método de eliminação de Gauss com os métodos iterativos de Jacobi, Gauss-Seidel e SOR são relevantes e podem servir como uma alternativa aos métodos iterativos clássicos.

Conclusão

A combinação do método de eliminação de Gauss com os métodos iterativos mostrou-se bastante promissora. Constatamos através dos testes computacionais que os algoritmos propostos podem ser uma alternativa aos métodos iterativos tradicionais, no sentido que são competitivos, principalmente no que se refere ao tempo de processamento. Isto já era esperado de antemão devido ao fato de os métodos híbridos serem aplicados em um sistema cuja matriz de coefientes tem zeros na primeira coluna, a partir da segunda linha. Estes métodos utilizam-se desta estrutura para economizar tempo, o que de fato foi verificado através dos testes que foram apresentados no capítulo 5.

Na comparação entre os métodos híbridos constatamos que o Método de Gauss-Seidel foi o que teve um melhor desempenho, porém o método SOR deveria ter um melhor desempenho caso o parâmetro de relaxação fosse otimizado, o qual está atrelado a cada matriz utilizada nos testes.

Deixamos como indicações para estudos futuros, algumas idéias que foram levantadas quando da finalização desse trabalho. Primeiramente, como se comportariam os métodos híbridos caso aumentássemos o número de iterações do método de eliminação gaussiana, antes de gerar o processo iterativo correspondente. Uma segunda tarefa para o futuro é fazer um estudo teórico relacionados à matrizes esparsas que são os casos em que os métodos iterativos são mais recomendados.

Concluímos que a metodologia proposta no trabalho, combinando uma iteração de método direto com métodos iterativos produziram bons resultados numéricos, pois quando comparados com os métodos tradicionais estes mostraram-se mais eficazes e portanto poder servir como uma alternativa aos métodos clássicos.

Bibliografia

[1] Albrecht, P. **Análise numérica: um curso moderno**, Rio de Janeiro: Livros Técnicos e Científicos, 1973

[2] Boyce, W. E.& DiPrima, R.C **Equações Diferenciais Elementares e Problemas de Valores de Contorno**, 6 ed., Rio de Janeiro: Editora LTC, 1999. San Diego, California: Academic Press, 1973.

[3] Cunha, C. **Métodos Numéricos.**, 2 ed.. Campinas, São Paulo: Editora da Unicamp, 2000.

[4] Datta, B. N. **Numerical Linear algebra and applications**. Brooks/Cole Publishing Company, 1994

[5] Lima, E. L. **Curso de Análise**, Rio de Janeiro, Instituto de Matemática Pura e Aplicada, CNPq, 1976

[6] Forsythe, G. E. & Moler, C. B. **Computer Solutions of Liner Algebraic Systems**. Englewood Cliffs, N.J.:Prentice Hall. 1967

[7] Golub, Gene & van Loan, Charles . **Matrix Computations**, 2. ed. Baltimore: The Johns Hopkins University Press,1989.

[8] Higham, N. J. **The Matrix Computational Toolbox**. http://www.ma.man.ac.uk/ higham/mctoolbox

[9] Kreyszig, E.. **Introductory Functional Analysis with Applicarions**, New York, John Wiley, 1989.

[10] Lay, David C.. **Álgebra Linear e suas Aplicações..** Rio de Janeiro:Editora LTC, 1999.

[11] Leon, Steven J.. **Álgebra Linear com Aplicações.** Rio de Janeiro: Editora LTC, 1999.

[12] Lipschutz, S.. **Álgebra Linear**. São Paulo: Editora McGraw-Hill do BrasilLtda, 1972.

[13] Noble, Ben & Daniel, J. W.**Álgebra Linear Aplicada**. 2. ed.. Rio de Janeiro: Editora Prentice-Hall do Brasil Ltda,1986.

[14] Ruggiero, Márcia A.G. & Lopes, Vera L. R. **Cálculo Numéricos - Aspectos Teóricos e Computacionais**,2 ed., São Paulo: MAKRON *Books*, 1996.

[15] Stewart, G. W.. **Introduction to matrix computations** San Diego, California: Academic Press, 1973.

[16] Varga, R. S.. **Matrix Iterative Analysis**, Englewood Cliffs. N.J.: Prentice Hall, 1962.

[17] Yuan, J. Yun. & Chang, Q. S..**Applied Iterative Analysis.** Institute of Applied Mathematics - Chinese Academy of Sciences, Beijing, China, 2000.