Universidade Federal de Santa Catarina
Curso de Pós-Graduação em Matemática e
Computação Científica

Métodos Numéricos Aplicados à Resolução das Equações da Rede Elétrica

Juliano de Bem Francisco
Orientador: Prof. Dr. Mario César Zambaldi

Florianópolis
Março de 2002

Universidade Federal de Santa Catarina
Curso de Pós-Graduação em Matemática e Computação Científica

Métodos Numéricos Aplicados à Resolução das Equações da Rede Elétrica

Dissertação apresentada ao Curso de Pós-Graduação em Matemática e Computação Científica, do Centro de Ciências Físicas e Matemáticas da Universidade Federal de Santa Catarina, para a obtenção do grau de Mestre em Matemática, com Área de Concentração em Matemática Aplicada.

Juliano de Bem Francisco
Florianópolis
Março de 2002

Métodos Numéricos Aplicados à Resolução das Equações da Rede Elétrica

por

Juliano de Bem Francisco

Esta Dissertação foi julgada para a obtenção do Título de "Mestre", Área de Concentração em Matemática Aplicada, e aprovada em sua forma final pelo Curso de Pós-Graduação em Matemática e Computação Científica.

Celso Melchíades Dória
Coordenador

Comissão Examinadora

Prof. Dr. Mario César Zambaldi (MTM-UFSC-Orientador)

Prof. Dr. Eng. Luciano Vitoria Barboza (UCPel - CEFET/RS)

Prof. Dr. Fermín Sinforiano Viloche Bazán (MTM-UFSC)

Prof. Dr. Clóvis Caesar Gonzaga (MTM-UFSC)

Florianópolis, março de 2002.

A minha família,
a minha avó
e a Deus.

Agradecimentos

Agradeço primeiramente, as pessoas as quais dedico este trabalho, responsáveis por tudo que sou hoje, minha mãe Ivonete e meu pai Agileu, dizendo as palavras certas nas ocasiões que mais precisei. A minha avó Lorena, que sempre estará zelando por mim. Meu irmão Agileu estando sempre do meu lado nos momentos mais difíceis e a Carina, por dedicar seu amor e sua atenção a mim.

Gostaria de agradecer ao amigo Mário Cesar Zambaldi, que além de me ensinar e orientar nesta dissertação, tornou-se um guia, sempre disposto a ajudar, ao meu amigo Luciano, pela disponibilização de seu tempo para tirar algumas dúvidas que apareceram, ao professor Pinho, por sempre acreditar em mim e a todas as pessoas que, direta ou indiretamente contribuíram para realização desta dissertação.

Não poderia deixar de mencionar os meus colegas extra universidade, que me apoiaram e me distraíram, não deixando que minha vida girasse em torno de livros.

Agradeço a coordenação de aperfeiçoamento de pessoal de nível superior - CAPES - por ter financiado este projeto pelo período de um ano, o suficiente para que eu conseguisse concluí-lo.

Para finalizar, agradeço a Deus e a seu filho Jesus Cristo por tudo. Amém.

Sumário

Lista de Figuras

Resumo

Neste trabalho, o problema de encontrar a solução para as equações da rede elétrica é abordado. O problema de fluxo de carga em redes de potência e o problema de restauração de solução, assim como a metodologia específica para ambos, são os temas centrais. Esta metodologia está baseada em métodos numéricos para problemas não lineares esparsos, onde os métodos iterativos em subespaço de Krylov tem um papel importante.

Abstract

The problem of finding the solution to the electric network equations is considered in this work. The power flow problem and the restoring solution problem as well as approaches to deal with them are the main subjetcs. This methodology is based on numerical methods for sparse nonlinear problems where Krylov subspace iterative algorithms play an important role.

Introdução

Devido ao aumento contínuo da demanda de energia elétrica nas últimas décadas, planejar sistemas de potência mais eficientes é de fundamental importância, representando um forte impacto econômico e social. Encontrar uma solução ótima para problemas desses sistemas é sinônimo de economia, eficiência e integridade dos componentes elétricos.

A formulação matemática do problema físico dos sistemas de potência origina um modelo muito interessante e que requer uma metodologia bem elaborada para a sua resolução. Neste contexto, resolver as equações oriundas desses sistemas de maneira ótima é indispensável.

O problema de fluxo de carga é representado por um sistema de equações não lineares, em geral com muitas variáveis, contendo as equações da rede elétrica. Técnicas de otimização aliadas a ferramentas da álgebra linear, mais especificadamente, métodos de Newton Inexatos com modernos métodos iterativos para resolver os sistemas lineares obtidos, constitui uma importante metodologia numérica. Um dos interesses deste trabalho é comparar o desempenho de diversos métodos iterativos em subespaços de Krylov, para a resolução do problema de fluxo de carga via o método de Newton inexato. Na tentativa de acelerar a convergência, foram usadas técnicas de precondicionamento baseadas na eliminação gaussiana, obtendo-se uma melhora significativa no desempenho.

Com a modelagem de sistemas de potência cada vez mais restritos e com estado de operação próximo aos seus limites, ocorrem casos frequentes nos quais as equações da rede elétrica não apresentam solução. A busca de uma solução viável resulta no problema de restauração da solução, evitando situações de colapso do sistema. Surge, então, um problema de quadrados mínimos não linear com restrições, que precisa ser abordado convenientemente. Várias metodologias têm sido adotadas neste sentido [3, 5, 9, 15]. A metodologia usada neste trabalho para encarar este tipo de problema é a mesma proposta em [5], usando uma outra função de mérito, a saber o Lagrangeano aumentado. Observando a taxa de convergência do

método, cogitou-se um mal condicionamento da matriz do sistema, constatado posteriormente, lançando mão de um sistema linear alternativo, com mais variáveis, mas com um condicionamento mais favorável, principalmente quando problemas reais são abordados.

Este trabalho desenvolve as metodologias citadas anteriormente para a busca da solução ótima para as equações da rede elétrica. A ênfase reside na implementação e avaliação do comportamento numérico destas metodologias, além de um estudo dos algoritmos iterativos lineares, ponto central no contexto das equações não lineares.

O conteúdo está organizado como segue. O primeiro capítulo trata do problema físico em questão, fluxo de carga com e sem solução, explicitando os parâmetros necessários e suas formulações. No segundo capítulo, descreve-se os métodos numéricos para problemas não lineares e todas as suas características direcionadas às formulações. Seguem no capítulo três os resultados numéricos resultantes das implementações computacionais. Dada a relevância dos modernos métodos iterativos como ponto central para a resolução de sistemas lineares esparsos, um estudo dos métodos em subespaços de Krylov é desenvolvido no capítulo quatro. Conclusões e futuras diretrizes de pesquisa na mesma linha do trabalho constituem o último capítulo.

Capítulo 1

O Problema das equações da rede elétrica

Este capítulo destina-se esclarecer o problema físico em questão, mostrando os problemas matemáticos, variáveis, parâmetros e formulações. Faz-se uma dedução do problema de fluxo de carga, tema central deste trabalho, e do problema de restauração da solução, ocorrendo quando o problema de fluxo de carga não tem solução.

1.1 Aspectos gerais - fluxo de carga

O cálculo do fluxo de carga em uma rede de energia elétrica consiste, essencialmente, na determinação do estado da rede, da distribuição dos fluxos e de algumas outras grandezas de interesse. Será abordardo o problema estático, significando que a rede é representada por um conjunto de equações/inequações algébricas.

Os componentes de um sistema de energia elétrica podem ser classificados em dois grupos:

- **barras** - *geradores, cargas, reatores e capacitores*
- **circuitos** - *elementos que interligam as barras. (linhas de transmissão e transformadores)*

As equações básicas do fluxo de carga são obtidas por meio da conservação das potências ativa e reativa em cada barra, isto é, a potência líquida injetada deve ser igual a soma das potências que fluem pelos componentes internos da barra. Isso equivale a impor a primeira lei de Kirchhoff.

O problema do fluxo de carga pode ser formulado por um sistema de equações e inequações algébricas não lineares que correspondem, respectivamente, as leis de Kirchhoff e a um conjunto de restrições operacionais da rede elétrica e de seus componentes. Na formulação mais simples do problema, para cada barra da rede são associadas quatro variáveis, sendo que duas delas entram no problema como dados e duas como incógnitas:

V_k - magnitude da tensão nodal na k-ésima barra

θ_k - ângulo de fase da tensão nodal na k-ésima barra

P_k - injeção líquida de potência ativa na k-ésima barra

Q_k - injeção líquida de potência reativa na k-ésima barra

A tensão completa na barra k é dada por $E_k = V_k e^{j\theta_k}$, sendo $j = \sqrt{-1}$. Dependendo de quais variáveis nodais entram como dados e quais são consideradas como incógnitas, definem-se três tipos de barras:

PQ - são dados P_k e Q_k, e calculados V_k e θ_k

PV - são dados P_k e V_k, e calculados Q_k e θ_k

Folga - são dados V_k e θ_k, e calculados P_k e Q_k

As barras do tipo PQ e PV são utilizadas para representar, respectivamente, barras de carga e barras de geração. A barra de folga tem dupla função: fornecer a referência angular e fechar o balanço de potência do sistema, levando em conta as perdas de transmissão não conhecidas antes de se obter a solução final do problema.

O conjunto de equações do problema do fluxo de carga é formado por duas equações para cada barra, cada uma delas representando o fato de as potências ativa e reativa injetadas em uma barra serem iguais à soma dos fluxos correspondentes que deixam a barra através de linhas de transmissão, transformadores, etc. Essas equações são representadas por:

$$P_k = \sum_{m \in \Omega_k} P_{km}(V_k, V_m, \theta_k, \theta_m)$$

$$Q_k = \sum_{m \in \Omega_k} Q_{km}(V_k, V_m \theta_k, \theta_m)$$

onde $k = 1, \ldots, nb$, sendo nb o número de barras e Ω_k é o conjunto das barras vizinhas à barra k. Duas barras são vizinhas quando existe um circuito interligando-as.

1.2 Modelagem de linhas de transmissão

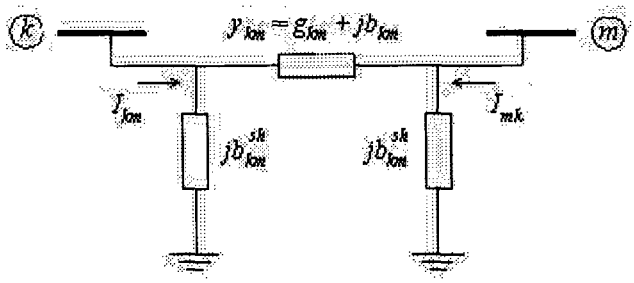

Figura 1.1: Linha de transmissão k-m

Seja I_{km} a corrente em uma linha de transmissão (que liga a barra k à barra m). A injeção líquida de corrente na barra k é obtida aplicando a primeira lei de Kirchhof:

$$I_k + I_k^{sh} = \sum_{m \in \Omega_k} I_{km} + I_k^{sh}, \qquad k = 1, \ldots, nb$$

Generalizando, pode-se escrever as equações das correntes em forma matricial:

$$I = YE,$$

onde I é o vetor das injeções de corrente, cujas componentes são I_k, $k = 1, \ldots, nb$. O vetor E representa as tensões complexas nodais, cujas componentes são $E_k = V_k e^{j\theta_k}$. A matriz $Y = G + jB$ é denominada matriz de admitância, sendo G a matriz de condutância e B a matriz de susceptância. Os elementos das matrizes G e B são obtidos a partir de manipulações algébricas nos parâmetros dos circuitos:

$$g_{km} = \frac{r_{km}}{r_{km}^2 + x_{km}^2} \qquad b_{km} = \frac{-x_{km}}{r_{km}^2 + x_{km}^2},$$

onde r_{km} e x_{km} são a resistência e a reatância série no circuito $k-m$, respectivamente. A impedância série é dada por $z_{km} = r_{km} + jx_{km}$. Em geral, a matriz Y é esparsa, pois, $Y_{km} = 0$ sempre que entre as barras k e m não existir circuito conectando-as.

As equações de potências ativa e reativa são deduzidas aplicando as

leis de Kirchhoff, dadas respectivamente por:

$$P_i^{cal} = G_{ii}V_i^2 + V_i \sum_{k\in\Omega_i} V_k[G_{ik}cos(\theta_i - \theta_k) + B_{ik}sen(\theta_i - \theta_k)] \quad (1.2.1)$$

$$Q_i^{cal} = -B_{ii}V_i^2 + V_i \sum_{k\in\Omega_i} V_k[G_{ik}sen(\theta_i - \theta_k) - B_{ik}cos(\theta_i - \theta_k)], \quad (1.2.2)$$

onde $i = 1, \ldots, nb$; Ω_i é o conjunto de índices das barras vizinhas à barra i excluindo a própria barra i.

1.3 Implementações

Considere inicialmente um problema no qual são dados P_k e Q_k para as barras PQ, P_k e V_k para as barras PV, e V_k e θ_k para a barra $V\theta$ (folga). Pede-se para calcular V_k e θ_k nas barras PQ, θ_k nas barras PV, e P_k e Q_k na barra de folga. Resolvido este problema, será conhecido o estado (V_k, θ_k) para todas as barras da rede $(k = 1, \ldots, nb)$. Sejam NPQ e NPV, respectivamente, o número de barras PQ e PV da rede (será considerada a existência de apenas uma barra de folga). O problema formulado anteriormente pode ser decomposto em dois subsistemas de equações algébricas, conforme indicado seguir:

Subsistema 1: (dimensão: $2NPQ + NPV$)

Neste subproblema são dados P_k e Q_k nas barras PQ e P_k e V_k nas barras PV. Pretende-se calcular V_k e θ_k nas barras PQ, e θ_k nas barras PV. Ou seja, trata-se de um sistema de $(2NPQ + NPV)$ equações algébricas não lineares com o mesmo número de incógnitas, ou seja:

$$P_k^{dado} - P_i^{cal} = 0 \quad para\ as\ barras\ PQ\ e\ PV;$$

$$Q_k^{dado} - Q_i^{cal} = 0 \quad para\ as\ barras\ PQ.$$

Subsistema 2:(dimensão: $NPV + 2$)

Resolvido o Subsistema 1 e, portando, conhecidos (V_k, θ_k) para todas as barras, deseja-se calcular P_k e Q_k na barra de folga, e Q_k nas barras PV. Considerando somente uma barra de folga, tem-se um sistema de $NPV + 2$ equações algébricas não lineares com o mesmo número de incógnitas, no qual todas as incógnitas aparecem de forma explícita, o que torna trivial o processo de resolução. Utiliza-se as equações (1.2.1) e (1.2.2) para a barra de folga e (1.2.2) para as barras PV.

O mesmo não ocorre com o subsistema 1, no qual as incógnitas são implícitas, o que exige um processo iterativo para resolvê-las. Os dois subsistemas correspondem ao *problema de Fluxo de Carga.*

A presente formulação considera limites (máximo e mínimo) na geração de potência reativa nas barras PV. Se durante o processo iterativo um desses limites for violado, Q_k será será fixado no valor extremo correspondente e a barra PV transforma-se em PQ; isto significa que a magnitude da tensão da barra PV não pode ser mantida no valor especificado. Nesse caso, faz-se $Q_k^{dado} = Q_k^{lim}$ e a equação correspondente do Subsistema 2 passa para o Subsistema 1. Eventualmente, numa iteração seguinte, a barra poderá voltar a ser do tipo PV. Chama-se este processo de *controle de reativos.*

As expressões do subsistema 1, podem ser reescritas do seguinte modo:

$$\Delta P_k = P_k^{dado} - P_k^{cal}(V,\theta) = 0 \;\; \textit{para as barras PQ e PV}$$

$$\Delta Q_k = Q_k^{dado} - Q_k^{cal}(V,\theta) = 0 \;\; \textit{para as barras PQ},$$

onde ΔP_k e ΔQ_k são respectivamente os balanços de potências ativa e reativa na barra k.

As funções ΔP_k e ΔQ_k podem ser colocadas na forma vetorial

$$\Delta P = P^{dado} - P^{cal}(V,\theta)$$

$$\Delta Q = Q^{dado} - Q^{cal}(V,\theta),$$

em que $P^{cal}(V,\theta)$ é o vetor das injeções de potência ativa nas barras PQ e PV, e $Q^{cal}(V,\theta)$, o das injeções de potência reativa nas barras PQ.

Considere a função vetorial dada por

$$F(x) = \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix}$$

Por meio dessa função, o subsistema 1 pode ser colocado na forma

$$F(x) = 0. \tag{1.3.1}$$

1.4 O problema sem solução

Pode-se dividir o estado da rede em três níveis de operação: região de operação, região de emergência e região sem solução real.

A região de operação caracteriza-se por apresentar pontos em que as equações estáticas do fluxo de carga possuem solução real, $F(x) = 0$, e não há violações dos limites operacionais. Esses limites podem ser fluxos de potência em circuitos, magnitudes de tensão, geração de potência ativa e reativa, etc. É nessa região que se deseja que os sistemas de energia elétrica operem.

Por outro lado, na região de emergência, as equações estáticas do fluxo de carga apresentam solução real, porém com violações de um ou mais limites operacionais. A princípio, é possível operar nesta região por um intervalo de tempo limitado. O importante é, a partir de um ponto de operação nessa região, utilizar mecanismos que possibilitem a migração do referido ponto para a região de operação. Tanto a região de operação como a de emergência são regiões onde as equações do fluxo de carga possuem solução real, $F(x) = 0$. Por isso, a união dessas duas regiões passará a ser referida como *região com solução do fluxo de carga.*

Existe ainda uma região sem solução, ou seja, aquela onde as equações estáticas do fluxo de carga não apresentam solução real. Qualquer tentativa de operar o sistema nesta região pode causar instabilidade no sistema, e até mesmo o colapso da tensão . Esse fenômeno pode ser observado em duas situações: quando o sistema elétrico torna-se extremamente carregado devido ao aumento de demanda e/ou quando o sistema sofre uma contingência severa.

Este trabalho, além da resolução do problema de fluxo de carga, visa a partir de um ponto na região sem solução e através de métodos de otimização, encontrar um ponto na região de emergência.

Como na região sem solução não existe x tal que $F(x) = 0$, será considerado o problema de quadrados mínimos não lineares:

$$\min \frac{1}{2} F(x)^T F(x) = \min \frac{1}{2} \|F(x)\|_2^2$$

Será considerada a existência de barras de injeção nula, nas quais não pode haver resíduos de potências. Pode-se considerar também o controle de reativos. Portanto, ao transformar um barra PV (digamos a barra k) em barra PQ, exige-se que $\Delta P_k = 0$ e $\Delta Q_k = 0$, ou seja, nesta barra não pode haver resíduos de potências ativa e reativa. Isto significa que as barras PV transformadas em PQ passam a ser

tratadas como barras de injeção nula.

Com essas considerações, o problema anterior fica:

$$\begin{array}{ll} \min & \frac{1}{2}\|F(x)\|_2^2, \\ s.a. & c(x) = 0 \end{array} \tag{1.4.1}$$

sendo que no vetor $c(x) : \mathbb{R}^n \rightarrow \mathbb{R}^r$ estão contidas as equações dos balanços de potências (ΔP, ΔQ) para as barras de injeção nula. O valor de r depende do sistema elétrico em questão, sendo que em problemas reais, r encontra-se entre 10% a 15% do número total de barras.

Capítulo 2

Métodos numéricos para problemas não lineares

Neste capítulo, será abordado algumas técnicas de otimização essenciais os sistemas não lineares: métodos de Newton inexato e problemas quadrados mínimos não lineares. Com a finalidade de complementar a teoria, essencialmente a exibida no método Newton inexato, comenta-se os métodos iterativos em subespaços de Krylov, sendo que uma teoria mais abrangente sobre tais métodos é desenvolvida no capítulo 4. As técnicas de precondicionamento para sistemas lineares, citadas no capítulo 3, são também descritas neste capítulo.

2.1 Sistemas não lineares

Em muitas aplicações, como em problemas de fluxo de carga, precisa-se resolver um sistema de equações não lineares do tipo

$$F(x) = 0, \tag{2.1.1}$$

onde $F : \mathbb{R}^n \to \mathbb{R}^n$, é uma função vetorial, ou seja

$$F(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{bmatrix}.$$

Uma maneira clássica para a resolução desse problema consiste em utilizar o método de Newton.

2.1.1 Método de Newton para sistemas não lineares

O método Newton para sistemas não lineares consiste em obter, através da fórmula de Taylor, uma aproximação linear de F em $x_k + p$ resultando:

$$M(x_k + p) = F(x_k) + J(x_k)p,$$

sendo $J(x) = \frac{\partial F}{\partial x}$ o Jacobiano de F.

A idéia é encontrar um zero desse modelo, ou equivalente, resolver o sistema linear não simétrico

$$J(x_k)p_k^N = -F(x_k), \tag{2.1.2}$$

e então, atualizar o iterado por $x_{k+1} = x_k + p_k^N$.

Teorema 2.1. *Suponha F continuamente diferenciável em uma vizinhança aberta de x^*, onde $F(x^*) = 0$, suponha também, J lipschitz nesta vizinhança. Considere o processo iterativo $x_{k+1} = x_k + p_k^N$, onde p_k^N é dado por (2.1.2). Então x_k converge para x^* com taxa de convergência quadrática para x_0 suficientemente próximo de x^*.*

Demonstração: Veja DENNIS e SCHNABEL [11].

■

Uma importante tarefa no método de Newton é o cálculo da matriz jacobiana, sendo impossível de obtê-la explicitamente em certos problemas. Felizmente, usar técnicas como diferenças finitas ou diferenciação automática [17], ajudam a contornar este problema.

Uma outra desvantagem ocorre quando o número de variáveis do problema é grande e a matriz jacobiana é esparsa, tornando inviável resolver o sistema (2.1.2) exatamente. Uma alternativa é usar métodos iterativos lineares, resolvendo o sistema de Newton com uma certa tolerância, necessitando conhecer apenas como a matriz jacobiana opera um dado vetor, ou seja, o resultado da multiplicação matriz-vetor. Esse tipo de implementação fornece o método chamado Newton inexato, explicado a seguir.

2.2 Newton inexato para sistemas não lineares

A idéia central do método de Newton inexato é lançar mão de um método iterativo linear, como os descritos no capítulo 4, e resolvê-lo até que uma certa precisão seja encontrada, obtendo uma solução aproximada (inexata) com certa tolerância.

Considere o resíduo

$$r_k = J_k p_k^I + F_k, \tag{2.2.1}$$

onde p_k^I é o passo do Newton inexato e J_k, o Jacobiano de F.

O método iterativo linear será interrompido quando

$$\|r_k\|_2 \leq \eta_k \|F_k\|_2, \tag{2.2.2}$$

onde a sequência $\{\eta_k\}$ é chamada de *sequência de termos forçantes.* O seguinte teorema mostra a influência desta sequência na taxa de convergência do método.

Teorema 2.2. *Suponha $F(x^*) = 0$, F contínua em uma vizinhança de x^* e $J(x^*)$ não singular. Considere a iteração $x_{k+1} = x_k + p_k^I$, onde p_k^I satisfaz (2.2.2). Seja x_0 suficientemente próximo de x^*, então:*

1. *Se $\eta_k \leq \eta$ onde $\eta \in [0, 1)$, $\{\|F_k\|_2\}_k \to 0$ linearmente;*
2. *Se $\eta_k \to 0$, $\{\|F_k\|_2\}_k \to 0$ superlinearmente;*
3. *Se $\eta_k \leq K\|F_k\|_2$, para alguma constante K, $\{\|F_k\|_2\}_k \to 0$ quadraticamente.*

Demonstração: Como $J(x^*)$ é não singular, existe $\delta > 0$ e uma constante positiva U tal que, para todo x na bola $B(x^*, \delta) = \{x \mid \|x - x^*\|_2 \leq \delta\}$, $\|J(x)^{-1}\|_2 \leq U$. Como $\{J(x_k)\}_k \to J(x^*)$, existe k_0 tal que $\|J(x_k)^{-1}\|_2 < U$, para todo $k > k_0$. Então, usando (2.2.1) e (2.2.2) tem-se:

$$\|p_k^I\|_2 \leq U(\|r_k\|_2 + \|F(x_k)\|_2) \leq \widetilde{U}\|F(x_k)\|_2.$$

Usando o teorema de Taylor, obtém-se

$$\begin{aligned} F(x_{k+1}) &= F(x_k) + J(x_k)p_k^I + O(\|p_k^I\|_2^2) \\ &= r_k + O(\widetilde{U}^2\|F(x_k)\|_2^2) \\ &= r_k + O(\|F(x_k)\|_2^2) \end{aligned}$$

então por (2.2.2)

$$\|F(x_{k+1})\|_2 \leq \eta_k\|F(x_k)\|_2 + O(\|F(x_k)\|_2^2)$$

de onde os resultados (1), (2) e (3) seguem.

■

O teorema anterior mostra que a sequência $\{\eta_k\}$ interfere significativamente na taxa de convergência do método de Newton inexato. Esses parâmetros são usados para obter uma certa precisão β, requerida nos métodos iterativos para resolver o sistema linear. Assim, para um passo k do Newton inexato, escolhe-se $\beta = \eta_k\|F_k\|_2$ e o sistema (2.1.2) é resolvido até esta precisão ser encontrada.

Vários métodos iterativos podem ser usados para a resolução do sistema (2.1.2). Entre os mais bem sucedidos e atuais estão aqueles baseados em subespaços de Krylov. Por agora, será apresentado, resumidamente, alguns destes métodos para sistemas lineares não simétricos. Para isso, considere o sistema (2.1.2) fazendo $J_k \equiv A$, $-F_k \equiv b$ e $p_k \equiv x$. Assim, tem-se o sistema linear

$$Ax = b.$$

O subespaço de Krylov associado a matrix A é definido por

$$K_m(A, v) = \{v, Av, \ldots, A^{m-1}v\},$$

para algum vetor v.

Os métodos iterativos não estacionários buscam encontrar uma solução aproximada em $K_m(A, r_0) \equiv K_m$, onde $r_0 = b - Ax_0$. A diferença entre os métodos depende de quatro tipos de projeções utilizadas para encontrar uma aproximação x_m, que são:

(A) $b - Ax_m \perp K_m$;

(B) $\|b - Ax_m\|_2$ ser mínimo sobre K_m;

(C) $b - Ax_m$ ser ortogonal a um subespaço $m - dimensional$ adequado;

(D) $\|x^* - x_m\|_2$ ser mínimo sobre $A^T K_m(A^T, r_0)$;

A abordagem (A) resulta o *Método de Ortogonalização Completa* (FOM) [21], a (B) resulta no método *Mínimo Resíduo Generalizado* (GMRES)[21], que é baseado no algoritmo de Arnoldi. Se em (C) for escolhido o subespaço $m - dimensional$ $K_m(A^T, r_0)$, obtém-se os métodos *Bi-Gradiente Conjugado* (Bi-CG)[21] e *Quasi-Mínimo Resíduo* (QMR)[21], baseados no processo de biortogonalização de Lanczos. A abordagem (D), mais recente, fornece os métodos que não usam a transposta da matriz do sistema, como o *Gradiente Conjugado Quadrado* (CGS)[23] e o *Bi-Gradiente Conjugado Estabilizado* (Bi-CGStab)[6], que são variações do Bi-CG.

Em implementações práticas, usa-se o GMRES com reinício, que consiste em fazer um certo número de iterações GMRES, por exemplo m, e reinicia-se o algoritmo com a solução inicial igual a aproximação x_m normalizada. Neste trabalho designa-se este procedimento por GMRES(m). Uma apresentação mais detalhada destes métodos, incluindo algoritmos e propriedades teóricas, é encontrada no capítulo 4.

Em muitos problemas práticos, resolver simplesmente o sistema linear $Ax = b$ através de um método iterativo, pode não resultar em boas propriedades de convergência, sendo necessário transformá-lo em um sistema linear mais favorável. Para isso, conhecer melhor técnicas de precondicionamento é indispensável.

2.3 Técnicas de precondicionamento

O desempenho dos métodos iterativos depende das propriedades espectrais da matriz do sistema [21]. O precondicionamento busca transformar o sistema linear original em um sistema equivalente, no sentido de ter a mesma solução, mas que tenha propriedades espectrais mais favoráveis.

Encontrar um precondicionador para o sistema $Ax = b$, é encontrar um matriz M, o precondicionador, com as propriedades:

- M ser uma boa aproximação para a matriz A;
- O custo da construção de M ser barato;

- O sistema $Mv = w$ ser muito mais fácil de resolver do que o sistema original.

A idéia é que a matriz $M^{-1}A$ tenha boas propriedades no sentido que os métodos iterativos convirjam mais rapidamente.

Existem diferentes maneiras de implementar o precondicionamento. Três delas são como segue:

- **Precondicionamento a esquerda:** Consiste em aplicar o método iterativo para o sistema linear $M^{-1}Ax = M^{-1}b$.

- **Precondicionamento a direita:** Aplicar o método iterativo ao sistema $AM^{-1}y = b$ e a solução x é obtida resolvendo $Mx = y$.

- **Precondicionamento bilateral:** Seja M um precondicionador com a fatoração $M = M_1M_2$. A idéia desta implementação é resolver o sistema $M_1^{-1}AM_2^{-1}z = M_1^{-1}b$ e depois encontrar a solução resolvendo $M_2x = z$. Note que, se a matriz A é simétrica e positiva definida, pode-se fatorar M de forma que $M_2 = M_1^T$ (fatoração de Cholesky) e assim o produto $M_1^{-1}AM_2^{-1}$ ainda permanece simétrico e positivo definido, o que não ocorre nos dois casos anteriores.

Existem várias maneiras de construir um precondicionador (veja [12]). Neste trabalho, será abordada técnicas baseadas na *fatoração LU incompleta* (ILU).

2.3.1 Fatoração LU incompleta (ILU)

Seja $A \in \mathbb{R}^{n\times n}$ uma matriz esparsa e considere P um subconjunto de $\{(i,j) \mid 1 \leq i,j \leq n, i \neq j\}$. A fatoração LU incompleta (ILU), consiste em encontrar matrizes esparsas L, triangular inferior, e U, triangular superior, de modo que $A = LU + R$, onde o padrão de esparsidade de L e U depende diretamente do subconjunto P, o qual será chamado *conjunto padrão zero*. O precondicionador M é então dado por $M = LU$. A obtenção dos fatores L e U é feita de modo que os elementos a_{ij}, tais que $(i,j) \in P$, não são processados na fatoração. Com isso, pode-se estabelecer o algoritmo geral.

Algoritmo 2.1. *Fatoração ILU Geral*

1. *Para* $i = 2, \ldots, n$ *faça*

2. *Para* $k = 1, \ldots, i-1$ *e para* $(i,j) \notin P$ *Faça*

3. $a_{ik} = a_{ik}/a_{kk}$

4. *Para* $j = k+1, \ldots, n$ *e* $(i,j) \notin P$ *Faça*

5. $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Se for escolhido o conjunto $P = \{(i,j) \mid a_{ij} = 0\}$, tem-se a bem conhecida *fatoração LU incompleta de nível 0,* ILU(0), onde os fatores L e U possuem o mesmo padrão de esparsidade das partes triangular inferior e triangular superior da matriz A, respectivamente. Note que o número de elementos não nulos do produto LU é maior que o da matriz A. A figura (2.1) mostra um exemplo da ILU(0) para uma matriz relacionada com diferenças finitas na discretização de EDP's.

Existem problemas em que a fatoração incompleta ILU(0) não produz um bom precondicionador. Para melhorar a precisão da fatoração e, consequentemente, diminuir o número de iterações, será introduzido implementações que diferem da ILU(0), permitindo a inserção de alguns elementos na estrutura original da matriz. Assim, os fatores L e U terão mais elementos não nulos do que as partes triangular inferior e superior da matriz A. O conceito de *níveis de preenchimento* é atribuído a cada elemento processado pela eliminação gaussiana.

Definição 2.1. *Seja A uma matriz esparsa, o nível de preenchimento de um elemento a_{ij} é definido por*

$$niv_{ij} = \begin{cases} 0 & \text{se } a_{ij} \neq 0, \text{ ou } i = j \\ \infty & \text{caso contrário} \end{cases}$$

Como a cada iteração este elemento é modificado na linha 5 *do algoritmo (2.1), niv_{ij} deve ser atualizado por*

$$niv(a_{ij}) = niv_{ij} = min\{niv_{ij}, niv_{ik} + niv_{kj} + 1\}. \tag{2.3.1}$$

Com a definição anterior, pode-se obter o seguinte conjunto:

$$P_l = \{(i,j) \mid niv_{ij} > l\},$$

onde niv_{ij} é o nível de preenchimento depois de todas as atualizações (2.3.1). Com esse conjunto, pode-se implementar a *fatoração LU incompleta de nível p,* ILU(l). Nesse caso, os elementos cujo o nível de preenchimento não excede l são mantidos .

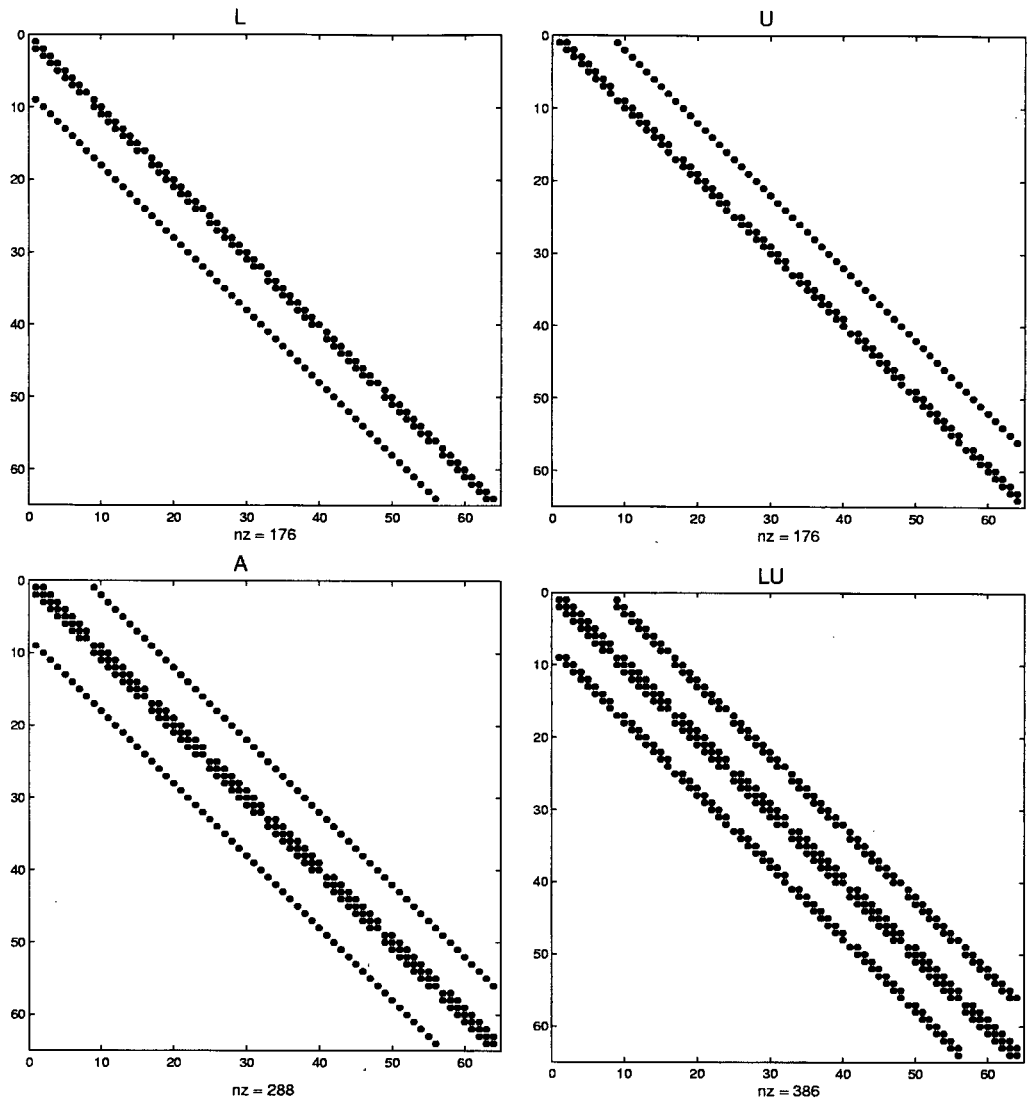

Figura 2.1: Exemplo da ILU(0): matrizes L, U, A e o produto LU, respectivamente

Algoritmo 2.2. *ILU(l)*

1. *Defina* $niv_{ij} = 0$ *onde* $a_{ij} \neq 0$
2. *Para* $i = 2, \ldots, n$ *faça*
3. *Para* $k = 1, \ldots, i-1$ *e* $niv_{ij} \leq l$ *Faça*
4. $a_{ik} = a_{ik}/a_{kk}$
5. *Para* $j = k+1, \ldots, n$ *Faça*
6. $a_{ij} = a_{ij} - a_{ik}a_{kj}$

7. *Atualize* niv_{ij} *usando (2.3.1)*

8. *Anule os elementos da linha i cujo* $niv_{ij} > l$

Quando a matriz A é simétrica e positiva definida, pode-se adaptar a fatoração Cholesky no algoritmo (2.2) e, assim, obter a implementação denominada *Cholesky Incompleta de nível l* .

2.3.2 Fatoração LU incompleta com tolerância (ILUT)

As fatorações incompletas descritas anteriormente são baseadas exclusivamente no padrão de esparsidade da matriz. Procedimentos desta natureza podem não ser suficientes em alguns casos. O processo de fatoração ILUT baseia-se na magnitude dos elementos. A idéia desta fatoração é descartar elementos de pequena magnitude.

Para implementar a fatoração ILUT, é necessário definir um fator de tolerância τ. No processo de fatoração os elementos dos fatores são descartados, exceto os das diagonais, se possuírem magnitude menor do que um certo escalar τ_i, obtido da multiplicação do parâmetro de tolerância τ pela norma da i-ésima linha da matriz em questão.

Algumas implementações, além de ignorar os elementos com magnitudes menores do que τ_i, mantém na linha da matriz apenas os p maiores elementos dos fatores L e U, controlando assim o número de elementos por linha.

2.4 Problemas de quadrados mínimos não lineares (PQMNL)

O problema de quadrados mínimos não lineares consiste em resolver o seguinte problema de otimização:

$$\min_x f(x) \equiv \min_x \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) = \min_x \frac{1}{2}\|r(x)\|_2^2, \tag{2.4.1}$$

onde $r(x) = (r_1(x), \ldots, r_m(x))^T$; cada $r_j : \mathbb{R}^n \to \mathbb{R}$ é designado como resíduo. Assim, a idéia deste tipo de problema é minimizar um resíduo oriundo de algum problema físico. Assume-se para esta seção f duas vezes diferenciável e $m \leq n$.

Definindo a matriz jacobiana de r por $J(x) = [\frac{\partial r_i}{\partial x_j}]_{i=1,\ldots,m,\ j=1,\ldots,n}$, tem-se que

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x)\nabla r_j(x) = J(x)^T r(x), \tag{2.4.2}$$

$$\begin{aligned}\nabla^2 f(x) &= \sum_{j=1}^{m} \nabla r_j(x)\nabla r_j(x)^T + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x) \\ &= J(x)^T J(x) + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x) \qquad (2.4.3) \\ &= J^T(x)J(x) + S(x), \qquad (2.4.4)\end{aligned}$$

onde $S(x) = \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x)$.

O método de Newton aplicado a esse tipo de problema é implementado obtendo-se a direção de descida p_k, ou seja, uma direção tal que $f(x_k + p_k) < f(x_k)$, resolvendo o seguinte sistema linear simétrico:

$$\nabla^2 f_k p_k = -\nabla f_k, \tag{2.4.5}$$

resultante da minimização do seguinte modelo quadrático de f:

$$m(x_k + p) = \frac{1}{2} r(x_k)^T r(x_k) + r(x_k)^T J(x_k)p + \frac{1}{2} p^T [J(x_k)^T J(x_k) + S(x)]p, \tag{2.4.6}$$

sendo $m_k : \mathbb{R}^n \to \mathbb{R}$. Sob certas condições, o método de Newton apresenta convergência quadrática.

Os problemas de quadrados mínimos não lineares podem ser distinguidos em resíduo-nulo, resíduo-pequeno, e resíduo-grande. Esta classificação está relacionada com o valor de f na solução x^*, sendo x^* a solução exata do problema em questão. Um problema onde $f(x^*) = 0$ é chamado resíduo-nulo. A diferença entre resíduo-pequeno e resíduo-grande está relacionada com a magnitude do termo $S(x)$.

Considere o modelo linear de r em torno de x_k :

$$M_k(x) = r(x_k) + J(x_k)(x - x_k), \tag{2.4.7}$$

onde $M_k : \mathbb{R}^n \to \mathbb{R}^m$, $m > n$. Quando não se pode obter uma raiz para este modelo,

a idéia é resolver o problema de quadrados mínimos linear:

$$\min_x \frac{1}{2}\|M_k(x)\|_2^2. \tag{2.4.8}$$

Será assumido que J tem posto completo. Neste caso, o argumento que minimiza o problema acima é dado por

$$x_{k+1} = x_k - [J(x_k)^T J(x_k)]^{-1} J(x_k)^T r(x_k). \tag{2.4.9}$$

Na prática, calcula-se x_{k+1} resolvendo o problema (2.4.8) através de uma decomposição QR de J_k. O processo iterativo (2.4.9) representa o método de Gauss-Newton que, em lugar de resolver o modelo quadrático (2.4.6), resolve o modelo linear (2.4.7). Outra maneira de interpretar esse método é eliminar da informação de segunda ordem, o termo $S(x)$ do modelo quadrático de que deriva do método de Newton.

O sucesso do método Gauss-Newton está relacionado com o quanto o termo omitido $S(x_k)$ é importante. Este resultado é observado no teorema (2.3). Ele mostra que se $S(x_k) = 0$, o método localmente tem taxa de convergência quadrática, ocorrendo quando $r(x)$ é linear (resíduo-nulo). Entretanto, quando $S(x^*)$ é muito representativo, o método de Gauss-Newton pode não convergir.

Teorema 2.3. *Sejam $r : \mathbb{R}^n \to \mathbb{R}^m$, e f como em (2.4.1), duas vezes continuamente diferenciáveis em um conjunto aberto $D \subset \mathbb{R}^n$. Assuma J lipschitz em D com constante de lipschitz γ, $\|J(x)\|_2 \leq \alpha \ \forall x \in D$, e que exista $x^* \in D$ e $\lambda, \sigma \geq 0$, tal que $J(x^*)r(x^*) = 0$, λ o menor autovalor de $J(x^*)^T J(x^*)$, e*

$$\|(J(x) - J(x^*))^T r(x^*)\|_2 \leq \sigma \|x - x^*\|_2$$

$\forall x \in D$. Se $\sigma < \lambda$, então para qualquer $c \in (1, \lambda/\sigma)$, existe $\epsilon > 0$ tal que $\forall x_0 \in B(x^*, \epsilon)$, a sequência gerada pelo método Gauss-Newton, converge para x^*, e

$$\|x_{k+1} - x^*\|_2 \leq \frac{c\sigma}{\lambda}\|x_k - x^*\|_2 + \frac{c\alpha\gamma}{2\lambda}\|x_k - x^*\|_2^2$$

Demonstração: Veja DENNIS e SCHNABEL[11].

■

Esse teorema mostra como a porção $S(x)$ está relacionada com a convergência do método Gauss-Newton, sendo verificada a taxa de convergência quadrática, quando $r(x^*) = 0$. Pode-se agora definir de problema resíduo-pequeno, quando $\|S(x^*)\|_2$ é pequena, e resíduo-grande caso contrário. Para certos problemas a estratégia de busca linear é indispensável para a convergência global de um método local.

2.4.1 Busca linear

Dada uma direção de descida p_k e um iterado x_k, seja $\phi(\alpha) = g(x_k + \alpha p_k)$, sendo $g : \mathbb{R}^n \to \mathbb{R}$. O parâmetro da busca linear é então definido por

$$\alpha_k = \text{argmin } \phi(\alpha), \text{ para } \alpha \geq 0. \tag{2.4.10}$$

Esse parâmetro é usado para obter um decaimento suficiente do valor da função objetivo. A atualização da solução é obtida fazendo-se $x_{k+1} = x_k + \alpha_k p_k$.

Pode-se resolver o problema (2.4.10), interpolando ϕ quadrática ou cubicamente (se a quadrática não for uma boa aproximação) em torno de zero, e então, minimizar esta aproximação, obtendo o parâmetro α_k. Para maiores detalhes veja [17]. Algumas implementações impõem duas condições sobre o parâmetro da busca linear, de modo que o decaimento da função objetivo seja suficiente e passos pequenos sejam evitados. A primeira é conhecida como *condição de Armijo*, a outra como *condição de curvatura* e são, respectivamente,

$$g(x_k + \alpha_k p_k) \leq g(x_k) + c_1 \alpha_k p_k^T \nabla g(x_k) \tag{2.4.11}$$

$$\nabla g(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla g(x_k)^T p_k, \tag{2.4.12}$$

onde $c_1 \in (0, 1)$ e $c_2 \in (c_1, 1)$. Na prática, c_1 deve ser pequeno, geralmente $c_1 = 10^{-4}$ e c_2 deve ser próximo de 1, um valor típico é $c_2 = 0.9$. As condições (2.4.11) e (2.4.12) quando juntas são chamadas *condições de Wolfe.*

A escolha da função g depende do problema abordado. Para problemas de quadrados mínimos não lineares, por exemplo, usa-se a própria função objetivo $f(x)$. Para sistemas não lineares, uma alternativa, é usar $g(x) = \frac{1}{2}\|F(x)\|_2^2$.

2.4.2 O PQMNL com restrições de igualdade

Para certos problemas, como por exemplo, o da restauração das equações da rede elétrica, é necessário introduzir restrições de igualdade no problema (2.4.1), originando o seguinte problema de otimização restrita:

$$\begin{array}{ll} \min & f(x), \quad x \in \mathbb{R}^n \\ s.a. & c(x) = 0 \quad \text{onde } c(x) = (c_1(x), \ldots, c_k(x))^T. \end{array} \tag{2.4.13}$$

Assim, a função lagrangeana é dada por

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x), \quad \lambda \in \mathbb{R}^k, \tag{2.4.14}$$

onde λ é o vetor dos multiplicadores de Lagrange.

A Hessiana do Lagrangeano é dada por

$$\nabla^2 \mathcal{L}(x, \lambda) = \begin{bmatrix} \nabla_{xx} \mathcal{L}(x, \lambda) & C(x)^T \\ C(x) & 0 \end{bmatrix}, \tag{2.4.15}$$

onde $C(x)$ é o Jacobiano de $c(x)$.

Dada uma direção de descida p_k a partir de x_k, a atualização da nova solução aproximada e dos multiplicadores de Lagrange para o problema é dada por

$$x_{k+1} = x_k + p_k^x \tag{2.4.16}$$

$$\lambda_{k+1} = \lambda_k + p_k^\lambda. \tag{2.4.17}$$

Técnicas para encontrar a direção

$$p_k = \begin{bmatrix} p_k^x \\ p_k^\lambda \end{bmatrix}$$

podem ser encontradas em [17].

Como para o problema irrestrito, a busca linear pode ser necessária. Neste caso, é preciso escolher uma função conveniente para realizar a busca. Esta função, denominada *função de mérito,* deve, obviamente, considerar as restrições.

Um passo p é aceito se ele reduzir suficientemente a função de mérito

ϕ definida por

$$\phi(x,\mu) = f(x) + \frac{1}{\mu}h(c(x)), \tag{2.4.18}$$

onde $h : \mathbb{R}^n \to \mathbb{R}$ satisfaz $h(y) \geq 0$, $\forall y \in \mathbb{R}^n$ e $h(0) = 0$. O parâmetro μ é denominado parâmetro de penalidade, tendendo a zero à medida que a solução computada aproxima-se da solução exata.

Entre as funções de mérito mais conhecidas estão a ℓ_1, que toma $h(x) = \|x\|_1$, a que toma $h(x) = \|x\|_2$, e a que escolhe $h(x) = \|x\|_\infty$. Note que as funções de mérito definidas com as funções h acima não são diferenciáveis, sendo para isso, necessário introduzir termos adicionais em (2.4.18). Uma função de mérito diferenciável usada é o Lagrangeano aumentado:

$$\mathcal{L}_A(x,\lambda,\mu) = f(x) - \lambda^T c(x) + \frac{1}{2\mu}\|c(x)\|_2^2, \tag{2.4.19}$$

que também será denotada por $\phi_F(x,\lambda,\mu)$, devido a Fletcher.

Uma alternativa para obter um parâmetro α_k para busca linear a partir de (x_k,λ_k,μ_k) com a direção de descida p_k em problemas de otimização restrita é resolver o problema

$$\alpha_k = \text{argmin } \phi_F(x_k + \alpha p_k^x, \lambda_k + \alpha p_k^\lambda, \mu_k), \text{ para } \alpha \in \mathbb{R}^+, \tag{2.4.20}$$

e as atualizações (2.4.16) e (2.4.17), são então dadas por

$$x_{k+1} = x_k + \alpha_k p_k^x \tag{2.4.21}$$

$$\lambda_{k+1} = \lambda_k + \alpha_k p_k^\lambda. \tag{2.4.22}$$

Capítulo 3

Resultados Numéricos

Este capítulo descreve os resultados obtidos na resolução dos problemas de fluxo de carga com e sem solução, respectivamente, os problemas (1.3.1) e (1.4.1), adotando metodologias explicadas nos capítulos anteriores. No primeiro problema, compara-se o desempenho de vários algoritmos iterativos para resolvê-lo através do método de Newton Inexato, enquanto no segundo, utiliza-se um método de quadrados mínimos não lineares restrito. Em todos os testes foram usadas técnicas de esparsidade [13] e realizados no Matlab 5.3 em um computador Pentium 400MHz com 64Mb de memória RAM.

3.1 Problema de fluxo de carga

Para resolver o problema (1.3.1), o critério de parada para as iterações externas para o Newton Inexato foi $\|F(x)\|_\infty < 10^{-3}$. Os sistemas-teste usados foram IEEE6, IEEE30, IEEE118 e SSB340 barras (SSB-sistema sul-sudeste brasileiro simplificado). Os métodos iterativos usados foram:

- Mínimo Resíduo Generalizado (GMRES)
- Bi-Gradient Conjugado (BiCG)
- Quasi-Mínimo Resíduo (QMR)
- Gradiente Conjugado Quadrado (CGS)
- Bi-Gradiente Conjugado Estabilizado (BiCG-Stab)

As dimensões dos Jacobianos dos sistemas testados estão na tabela (3.1).

Tabela 3.1: Dimensões dos Jacobianos

Sistema	IEEE6	IEEE30	IEEE118	SSB340
dimensão	9×9	53×53	201×201	626×626

O desempenho dos métodos para cada sistema sem o uso de um precondicionador estão nas tabelas (3.2), (3.3) e (3.4). O tempo foi relativo, ou seja, para cada tabela, divide-se todos os tempos obtidos pelo menor tempo entre eles. Cada flop significa uma operação elementar (soma, subtração, multiplicação e divisão) em aritmética de ponto flutuante. No GMRES a(b) significa b iterações no reinício a, sendo permitidas no máximo 20 iterações por reinício. A sequência de termos forçantes η_k para os métodos iterativos foi gerada por

$$\eta_k = \eta_1^k, \tag{3.1.1}$$

onde $\eta_1 \approx 0.8$, garantindo assim, a taxa de convergência superlinear. As tabelas mostram melhor eficiência dos métodos BiCG-Stab e CGS. Para o sistema elétrico SSB340, nenhum método iterativo sem precondicionamento convergiu.

Tabela 3.2: Desempenho dos métodos sem precondicionador para 6 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(2); 1(7); 1(8)	1,0	14.365
BiCG	4; 8; 9	1,0	12.757
QMR	2; 8; 9	1,0	15.485
CGS	2; 6; 8	1,0	10.617
BiCG-Stab	2; 6; 7	1,0	12.938

Tabela 3.3: Desempenho dos métodos sem precondicionador para 30 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(7); 2(18); 4(7)	1,6	649.140
BiCG	22; 22; 45	1,0	309.603
QMR	9; 36; 57	1,2	455.261
CGS	26; 28; 35	1,0	320.919
BiCG-Stab	5; 20; 30	1,0	280.696

Tabela 3.4: Desempenho dos métodos sem precondicionador para 118 barras

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(6); 2(12); 11(1); 9(14)	2,1	8.690.682
BiCG	12; 55; 109; 110	1,3	3.542.520
QMR	7; 66; 90; 104	1,4	4.289.398
CGS	11; 70; 71	1,0	1.987.524
BiCG-Stab	5; 22; 48; 56	1,1	2.339.616

Sem o uso de um precondicionador, o resíduo em cada iteração do GMRES apresenta um baixo decréscimo, principalmente na última iteração do Newton inexato, como mostrado na figura (3.1) para o sistema de 30 barras que, conforme a proposição (4.4), indica que a matriz Hessemberg H_m tende ao mal condicionamento. O problema se agrava a medida que a dimensão do sistema aumenta, explicando a não convergência dos método para o sistema de 340 barras.

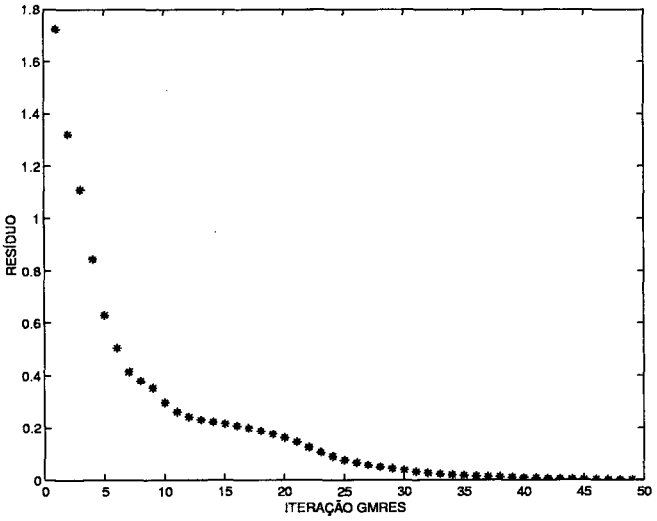

Figura 3.1: Comportamento da norma do resíduo do GMRES para 30 barras na última iteração de Newton.

No que se segue, o precondicionamento ILU(0) foi utilizado para acelerar a convergência. As tabelas (3.5),(3.6) e (3.7) apresentam os resultados, mostrando o bom funcionamento para sistemas elétricos com poucas barras. Os melhores rendimentos em flops, números de iterações e tempo foram obtidos pelos métodos BiCG-Stab e CGS. A sequência de termos forçantes para os métodos foi igual ao caso sem precondicionamento. Para diminuir o custo computacional, o precondicionador

usado na primeira iteração de Newton foi também utilizado nas iterações subsequentes. Bons resultados para sistemas pequenos foram obtidos com esta estratégia. Novamente, este tipo de precondicionamento não foi suficiente para a convergência do sistema de 340 barras, mesmo atualizando o precondicionador em todas as iterações de Newton.

Tabela 3.5: Desempenho dos métodos com ILU(0) para 6 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(1); 1(2); 1(2)	1,8	6.871
BiCG	1; 2; 2	1,8	6.541
QMR	1; 2; 2	1,8	7.963
CGS	1; 2; 3	1,0	7.139
BiCG-Stab	1; 1; 2	1,0	6.341

Tabela 3.6: Desempenho dos métodos com ILU(0) para 30 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(1); 1(6); 1(8)	1,2	97.176
BiCG	5; 3; 9	1,0	104.704
QMR	2; 6; 8	1,2	119.396
CGS	2; 4; 3	1,0	63.000
BiCG-Stab	1; 3; 3	1,0	63.884

Tabela 3.7: Desempenho dos métodos com ILU(0) para 118 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(1); 1(8); 1(14); 1(15)	1.2	900.457
BiCG	1; 12; 17; 15	1,1	943.909
QMR	1; 8; 15; 15	1,1	990.949
CGS	1; 11; 9; 16	1,0	803.174
BiCG-Stab	1; 5; 6; 8	1,0	560.988

O número de operações em ponto flutuante (flops) diminui significativamente, quando comparado com o caso não precondicionado. O número de iterações também diminui, indicando uma grande redução no custo computacional.

Nota-se que o método mais sensível a um precondicionador foi o GMRES. No sistema IEEE118, o número de flops foi reduzido significativamente comparado com o caso sem precondicionamento: de 8.690.682 flops para 900.457 com ILU(0).

Observando a estrutura do Jacobiano para 118 e 340 barras, figura (3.2), nota-se a presença de muitos elementos não nulos "longe" da diagonal principal. Este fato foi negativo para o desempenho dos métodos iterativos que usam como precondicionador a fatoração ILU(0), principalmente para grandes sistemas, pois esta leva em consideração a estrutura de esparsidade da matriz. De fato, para o sistema elétrico SSB340, o fator U da ILU(0) torna-se singular.

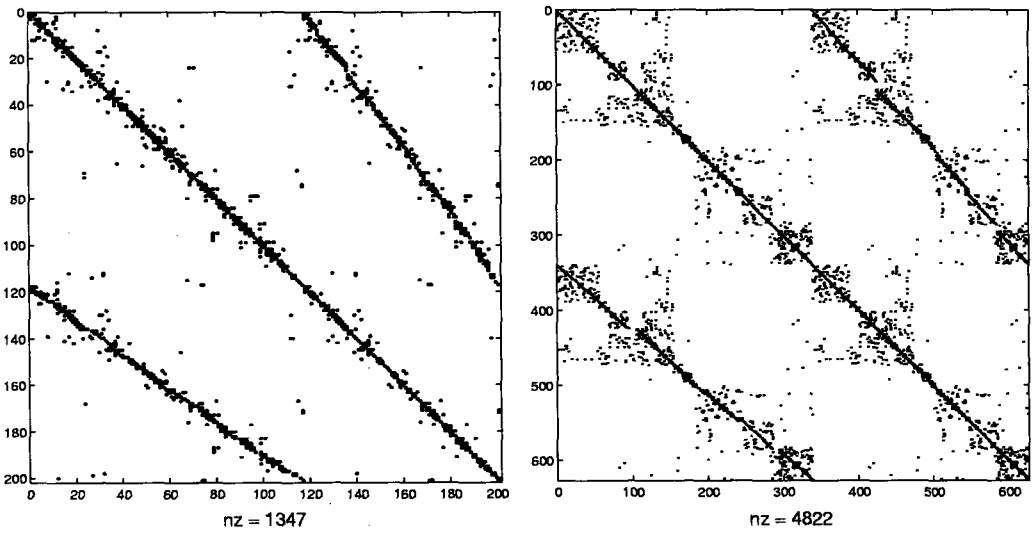

Figura 3.2: Estrutura do Jacobiano para 118 e 340 barras, respectivamente

Para obter a convergência para o sistema de 340 barras foi necessário usar uma técnica de precondicionamento que considera os valores numéricos diferentemente da ILU(0). Assim, utilizou-se a fatoração ILUT descrita anteriormente.

As tabelas (3.8), (3.9), (3.10) e (3.11) apresentam os resultados com este tipo de precondicionador. Pode-se notar o melhor desempenho dos métodos BiCG-Stab e CGS. Comparando o número de flops no IEE118 com precondicionador ILU(0), nota-se uma redução significativa. As sequências de termos forçantes foram atualizadas como em (3.1.1), com $\eta_1 \approx 0.8$ para os sistemas de IEEE6, IEEE30 e IEEE118 barras e $\eta_1 \approx 0.85$ para o SSB340. O parâmetro de tolerância τ da ILUT foi de 10^{-1} para os sistemas de 6 e 30 barras e 10^{-2} para os de 118 e 340 barras, sendo que para os sistemas de 6, 30 e 118 o precondicionador foi atualizado somente na primeira iteração, enquanto para o sistema de 340 barras foi necessário atualizar

na primeira e terceira iteração do Newton inexato.

Tabela 3.8: Desempenho dos métodos com ILUT para 6 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(1); 1(2); 1(3)	1,8	7.365
BiCG	1; 4; 3	1,8	8.570
QMR	1; 2; 3	1,8	8.770
CGS	1; 1; 2	1,0	5.752
BiCG-Stab	1; 1; 2	1,8	6.006

Tabela 3.9: Desempenho dos métodos com ILU(0) para 30 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(3); 1(6); 1(8)	1,2	99.994
BiCG	6; 7; 8	1,1	121.279
QMR	4; 8; 6	1,1	127.795
CGS	3; 6; 5	1,0	87.336
BiCG-Stab	3; 3; 4	1,0	78.878

Tabela 3.10: Desempenho dos métodos com ILUT para 118 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(1); 1(3); 1(4)	1,0	323.306
BiCG	1; 4; 4	1,0	358.219
QMR	1; 4; 4	1,0	409.213
CGS	1; 2; 3	1,0	289.109
BiCG-Stab	1; 1; 2	1,0	273.286

Para visualizar a redução do número de iterações para cada tipo de precondicionamento usado, a figura (3.3) mostra um gráfico com o número de iterações do GMRES em cada iteração do Newton inexato para os sistemas de 30 e 118 barras, comparando o desempenho do método com e sem o uso de precondicionador.

Tabela 3.11: Desempenho dos métodos com ILUT para 340 barras.

Método	Núm. de iterações em cada passo do Newton Inex.	Tempo	Núm. de flops
GMRES	1(1); 1(4); 1(9); 1(11) 1(12); 1(13)	1,0	4.947.031
BiCG	1; 2; 9; 13; 13; 12	1,2	5.087.058
QMR	1; 3; 10; 12; 13; 13	1,2	5.959.165
CGS	1; 2; 8; 5; 9; 11	1,0	4.056.836
BiCG-Stab	1; 2; 5; 5; 5; 9	1,0	3.807.421

3.2 Problema sem solução - restauração de solução

Devido às circunstâncias do sistema elétrico, o problema de fluxo de carga, abordado na seção anterior, pode não ter solução, sendo em muitos casos importante resolver um problema de quadrados mínimos não linear, encontrando portanto, uma solução de norma mínima. Este tipo de problema é conhecido na literatura técnica como *problema de restauração de solução* [3]. Neste caso, o seguinte problema deve ser abordado:

$$\begin{array}{ll} \min & \frac{1}{2}\|r(x)\|_2^2, \\ s.a. & c(x) = 0 \end{array} \tag{3.2.1}$$

Como em (1.4.1), tem-se um problema de quadrados mínimos não linear restrito.

As inicializações das variáveis foram:

- $V^0 = (1, 1, \ldots, 1)^T$;
- $\theta^0 = (0, 0, \ldots, 0)^T$;
- $\lambda^0 = (0, 0, \ldots, 0)^T$.

Diferentemente de [5], onde foi utilizado o Lagrangeano padrão, como função de mérito, aqui foi utilizado o Lagrangeano aumentado.

$$\mathcal{L}_A(x, \lambda, \mu) = \frac{1}{2}\|r(x)\|_2^2 + \lambda^T c(x) + \frac{1}{2\mu}\|c(x)\|_2^2.$$

Como sugerido em [17].

Os sistemas elétricos testados foram IEEE6, IEEE30, IEEE118 e SSB340, com respectivamente 6, 30, 118 e 340 barras, com a tolerância para a convergência

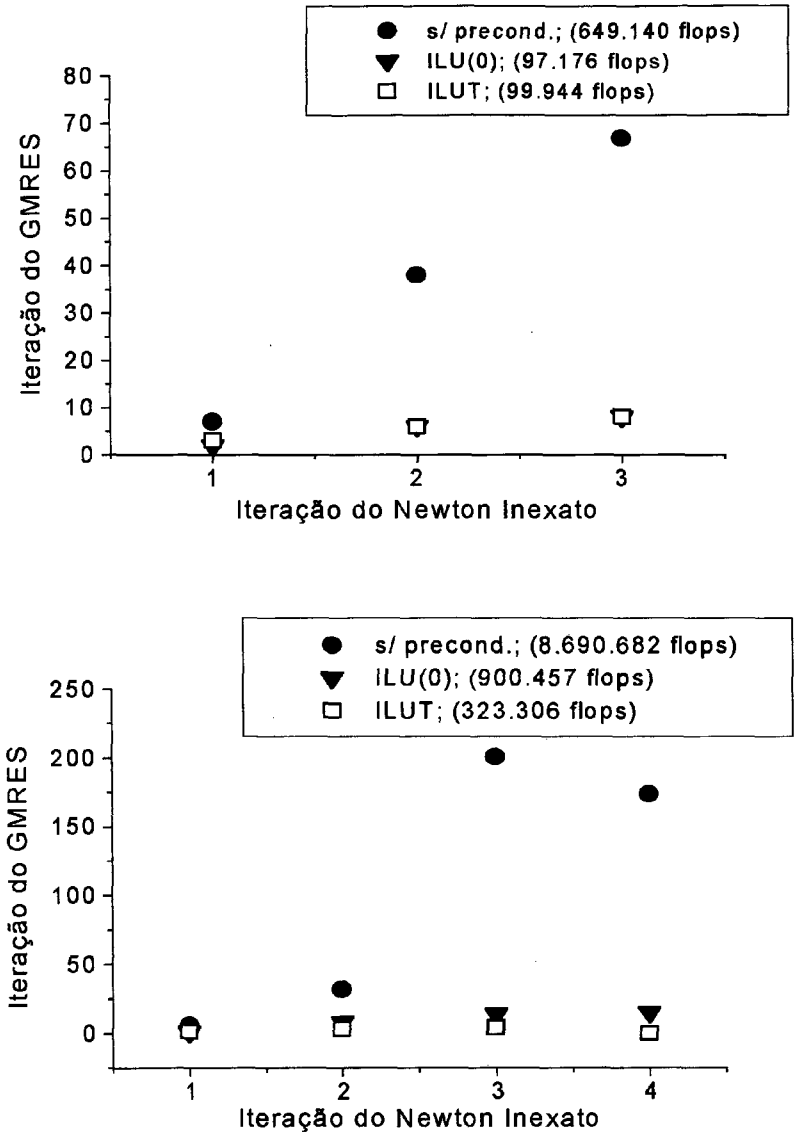

Figura 3.3: Desempenho do GMRES em cada iteração do Newton Inexato para 30 e 118 barras, respectivamente.

de $\|\nabla \mathcal{L}\|_\infty < 10^{-3}$. A dimensão da função c em (3.2.1) depende de cada sistema teste, sendo que para o IEEE6 $c \in \mathbb{R}^2$, para o IEEE30 $c \in \mathbb{R}^{12}$, IEEE118 $c \in \mathbb{R}^{20}$ e para o SSB340 $c \in \mathbb{R}^{206}$. Para nenhum sistema teste foi feito controle de reativo, o que envolveria maiores dificuldades computacionais.

A tabela (3.12) mostra o comportamento do gradiente da função lagrangeana. Para os sistemas de 118 e 340 barras, foram necessárias algumas iterações de Gauss-Newton iniciais para garantir a convergência, como em BARBOZA [3]. Será chamada de GN as iterações de Gauss-Newton e N as de Newton. Para os sistemas

de 6 e 30 barras não foi necessária busca linear.

Tabela 3.12: Comportamento do Gradiente do Lagrangeano

	6 barras		30 barras		118 barras		340 barras	
Iter	$\|\nabla\mathcal{L}\|_\infty$	tipo	$\|\nabla\mathcal{L}\|_\infty$	tipo	$\|\nabla\mathcal{L}\|_\infty$	tipo	$\|\nabla\mathcal{L}\|_\infty$	tipo
1	4,7098	N	$1{,}0494\times10$	N	$2{,}4263\times10^{2}$	GN	$1{,}5231\times10^{4}$	GN
2	1,3514	N	1,0141	N	$7{,}5553\times10^{1}$	GN	$2{,}1948\times10^{3}$	GN
3	$3{,}0659\times10^{-1}$	N	$3{,}4897\times10^{-1}$	N	2,4974	N	$1{,}2519\times10^{2}$	GN
4	$1{,}1519\times10^{-1}$	N	$1{,}1138\times10^{-1}$	N	1,8953	N	4,6269	N
5	$2{,}7824\times10^{-2}$	N	2.9974×10^{-2}	N	$4{,}4772\times10^{-1}$	N	$3{,}2312\times10^{-1}$	N
6	$1{,}5370\times10^{-3}$	N	5.6653×10^{-3}	N	$6{,}6687\times10^{-2}$	N	$2{,}2665\times10^{-2}$	N
7	$5{,}3066\times10^{-5}$	N	$1{,}1400\times10^{-3}$	N	$3{,}3115\times10^{-3}$	N	$1{,}3766\times10^{-4}$	N
8			$6{,}4668\times10^{-5}$	N	5.2351×10^{-6}	N		

Note que, para os sistemas de 6 e 30 barras, a taxa de convergência do Newton foi praticamente linear. Por outro lado, para os sistemas de 118 e 340 barras a convergência foi melhor devido às iterações de Gauss-Newton iniciais.

A taxa de convergência linear indica um mal condicionamento da matriz hessiana do lagrangeano. Sua forma matricial é dada por (2.4.15), onde

$$\nabla_{xx}\mathcal{L}(x,\lambda) = J^T(x)J(x) + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x) + \sum_{i=1}^{k} \lambda_i \nabla^2 c_i(x).$$

sendo que $J(x)$ representa o Jacobiano da função $r(x)$, e $C(x)$ o Jacobiano de $c(x)$.

Uma tentativa de melhorar o condicionamento do sistema foi trabalhar diretamente com $J(x)$, originando um sistema estendido, no qual o fator $J(x)^T J(x)$, um agravante para mal condicionamento, desaparece:

$$\begin{bmatrix} D(x_k,\lambda_k) & J(x_k)^T & C(x_k)^T \\ J(x_k) & -I & O \\ C(x_k) & O & O \end{bmatrix} \begin{bmatrix} p_k^x \\ \zeta \\ p_k^\lambda \end{bmatrix} = - \begin{bmatrix} J(x_k)^T r(x_k) + C(x_k)^T \lambda_k \\ 0 \\ c(x_k) \end{bmatrix},$$

onde $D(x_k,\lambda_k) = \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x) + \sum_{i=1}^{k} \lambda_i \nabla^2 c_i(x)$ e ζ um vetor auxiliar. O novo sistema tem um incremento de m linhas e m colunas, sendo

$$m = 2(n^{\underline{o}} \textit{ de barras } PQ) + (n^{\underline{o}} \textit{ de barras } PV).$$

Embora a dimensão do sistema aumente, o custo computacional não é muito alter-

ado se técnicas de esparsidade forem usadas, já que a nova matriz possui grau de esparsidade próximo ao sistema original.

As tabelas (3.13) e (3.14) mostram o condicionamento de ambas as matrizes para os sistemas elétricos em questão. Para os IEEE118 e SSB340, foi calculado o número de condição aproximado, que nesse trabalho corresponde a norma euclidiana [14], ou seja,

$$\textit{núm. de condição} = k_2(A) = \|A\|_2\|A^{-1}\|_2.$$

O condicionamento do sistema gradativamente melhora à medida que a dimensão do sistema aumenta. Este fato tem uma influência marcante quando sistemas de grande porte forem abordados, por exemplo SSB340. Embora haja melhoras no condicionamento do sistema linear, a sequência de iterados permaneceu a mesma, fato que pode não acontecer em problemas de grande porte, (problemas reais), como é o caso do SSB1916 com 1916 barras.

Tabela 3.13: Número de condição para os sistemas de 6 e 30 barras

	número de condição			
	sistema original		sistemas estendido	
Iter	6 barras (11 × 11)	30 barras (65 × 65)	6 barras (18 × 18)	30 barras (106 ×106)
1	$3{,}4610\times10^{2}$	$9{,}9046\times10^{4}$	$8{,}9523\times10$	$2,3139 \times 10^{3}$
2	$3,3949 \times 10^{2}$	$6,5118 \times 10^{4}$	$6,5716 \times 10$	$1,3914 \times 10^{3}$
3	$4,3078 \times 10^{2}$	$1,1869 \times 10^{5}$	$7,7508 \times 10$	$2,4321 \times 10^{3}$
4	$5,9998 \times 10^{2}$	$1,8705 \times 10^{5}$	$1,1095 \times 10^{2}$	$3,8066 \times 10^{3}$
5	$7,6902 \times 10^{2}$	$2,8916 \times 10^{5}$	$1,5019 \times 10^{2}$	$5,8966 \times 10^{3}$
6	$8,3860 \times 10^{2}$	$4,2295 \times 10^{5}$	$1,6984 \times 10^{2}$	$8,6441 \times 10^{3}$
7	$8,5402 \times 10^{2}$	$5,3590 \times 10^{5}$	$1,7447 \times 10^{2}$	$1,0958 \times 10^{4}$
8		$5{,}9145\times10^{5}$		$1,2097 \times 10^{4}$

O padrão de esparsidade da matriz hessiana do lagrangeano está representada na figura (3.4). Pode-se observar uma grande quantidade de elementos não nulos longe da diagonal principal. Este caso está longe de ser uma estrutura favorável para manipulação de dados, como seria se a matriz tivesse uma estrutura de banda por exemplo. Isso, aliado ao mal condicionamento, torna inviável usar um método iterativo linear para resolver o sistema, pois este depende de um bom precondicionador, como visto anteriormente. Uma alternativa seria usar os métodos iterativos para os sistemas estendidos, já que seus condicionamentos tornam-se mais adequados.

Tabela 3.14: Número de condição para os sistemas de 118 e 340 barras

	número de condição			
	sistema original		sistemas estendido	
Iter	118 barras (221 x 221)	340 barras (832x832)	118 barras (402 x 402)	340 barras (1252 x 1252)
1	$5{,}0058\times10^{6}$	$2{,}6208\times10^{10}$	$3{,}1438\times10^{4}$	$5{,}6269\times10^{6}$
2	$1{,}7007\times10^{7}$	$2{,}7096\times10^{10}$	$9{,}7959\times10^{4}$	$5{,}0157\times10^{6}$
3	$2{,}8755\times10^{6}$	$2{,}6300\times10^{10}$	$1{,}9463\times10^{4}$	$5{,}3708\times10^{6}$
4	$1{,}0622\times10^{7}$	$2{,}0251\times10^{10}$	$6{,}1076\times10^{4}$	$4{,}1181\times10^{6}$
5	$1{,}2733\times10^{7}$	$2{,}7374\times10^{10}$	$7{,}2350\times10^{4}$	$5{,}5681\times10^{6}$
6	$1{,}1765\times10^{7}$	$2{,}7796\times10^{10}$	$6{,}5742\times10^{4}$	$5{,}6504\times10^{6}$
7	$1{,}2337\times10^{7}$	$2{,}7927\times10^{10}$	$6{,}8533\times10^{4}$	$5{,}6763\times10^{6}$
8	$1{,}2384\times10^{7}$		$6{,}8776\times10^{4}$	

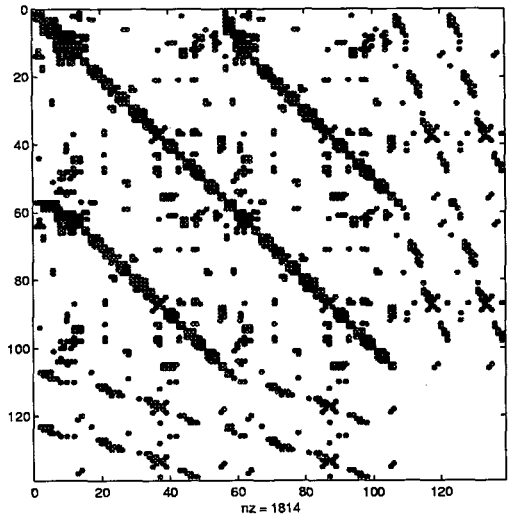

Figura 3.4: Estrutura da matriz Hessiana

Capítulo 4

Métodos Iterativos em Subespaço de Krylov

Os métodos iterativos consistem em técnicas que usam sucessivas aproximações, para que, a cada passo, obtenham-se soluções cada vez mais precisas para um sistema linear. Atualmente, os métodos mais elaborados pertencem à classe dos métodos não estacionários e, em geral, baseados na idéia de seqüência de vetores ortogonais.

Este capítulo considera alguns dos métodos não estacionários mais bem sucedidos, denominados métodos baseados em subespaçøde Krylov, dos quais o método do Gradiente Conjugado clássico, aplicado para matrizes simétricas, é o representante mais popular. Mais especificamente, são abordados os métodos de ortogonalização Completa (FOM), Mínimo Resíduo Generalizado (GMRES), Bi-Gradiente Conjugado (BiCG), Quasi-Mínimo Resíduo (QMR), Gradiente Conjugado ao quadrado (CGS) e Bi-Gradiente Conjugado Estabilizado (BiCGStab). As taxas de convergência destes métodos depende sensivelmente do espectro da matriz do sistema, envolvendo então, uma segunda matriz chamada de *precondicionador* vista no capítulo 2, transformando a matriz de coeficientes em uma matriz com o espectro mais favorável. Como visto anteriormente, o uso de um bom precondicionador deve melhorar a convergência do método, suficientemente para sobressair-se sobre o custo de seu cálculo e sua aplicação. Em geral, para muitos problemas práticos, o uso do precondicionador para o sistema é absolutamente necessário.

4.1 Introdução

Considere o sistema linear

$$Ax = b; \tag{4.1.1}$$

os métodos iterativos baseados em subespaços de Krylov consistem em buscar uma solução x_m do sistema (4.1.1) no subespaço afim $x_0 + K_m$ de dimensão m, impondo as condições de Petrov-Galerkin

$$b - Ax_m \perp L_m,$$

sendo L_m um subespaço de dimensão m. Aqui, x_0 representa a aproximação inicial para a solução. O subespaço K_m, denominado *subespaço de Krylov*, é dado por

$$K_m(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\},$$

onde $r_0 = b - Ax_0$. As diferentes versões dos métodos em subespaços de Krylov consistem das diferentes escolhas do subespaço L_m e da maneira com que o sistema é precondicionado.

Do ponto de vista de teoria de aproximação, as soluções aproximadas obtidas dos métodos em subespaço de Krylov são da forma

$$A^{-1}b \approx x_m = x_0 + q_{m-1}(A)r_0,$$

sendo q_m um polinômio de grau $m-1$. Quando $x_0 = 0$, tem-se

$$x_m \approx q_{m-1}(A)b,$$

ou seja, a solução x_m é aproximada por $q_{m-1}(A)b$.

Embora todas as técnicas forneçam os mesmo tipo de aproximação polinomial, a escolha do subespaço L_m terá um efeito importante no método iterativo. Duas classes de escolhas fornecem as técnicas mais conhecidas. A primeira consiste em tomar $L_m = K_m$ e para as variações de resíduo mínimo $L_m = AK_m$. A segunda classe consiste em definir L_m um subespaço de Krylov associado com a matriz A^T, $L_m = K_m(A^T, r_0)$.

4.2 Método de Arnoldi

O método de Arnoldi foi introduzido em 1951 com a intenção de reduzir uma matriz densa à sua forma Hessenberg. Este é um método de projeção ortogonal em K_m. Arnoldi apresentou seu método como uma maneira de encontrar autovalores da matriz Hessenberg em um número de passos menor do que n, fornecendo estimativas precisas para alguns autovalores da matriz original. Depois, esta estratégia mostrou-se muito eficiente para aproximar autovalores de matrizes esparsas.

4.2.1 O algoritmo de Arnoldi

Considere o sistema (4.1.1). O procedimento de Arnoldi é um algoritmo para construir uma base para o subespaço de Krylov K_m. A idéia é obter um conjunto de vetores ortonormais $\{v_1, v_2, \ldots, v_m\}$ de modo que

$$V_m^T A V_m = H_m, \tag{4.2.1}$$

onde V_m é formada pelos vetores $\{v_i\}_{i=1,\ldots,m}$ e $H_m \in R^{m\times m}$ é uma matriz Hessenberg. A matriz V_m é ortonormal, assim, de (4.2.1)

$$AV_m = V_m H_m.$$

Comparando as colunas de ambos os lados tem-se

$$Av_k = \sum_{i=1}^{k+1} h_{i,k} v_i.$$

Isolando o último termo na soma resulta em

$$h_{k+1,k} v_{k+1} = Av_k - \sum_{i=1}^{k} h_{i,k} v_i,$$

onde $h_{ik} = q_i^T A q_k$ para $i = 1 \ldots k$.

Estas equações definem o processo de Arnoldi:

Algoritmo 4.1. Arnoldi

1. *Escolha um vetor v_1 de norma 1.*

2. *Para j=1,2,...,m*

a) $h_{ij} =< Av_i, v_j >,\ i = 1, 2, \ldots, j,$

b) $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i,$

c) $h_{j+1,j} = \|w_j\|_2,$

d) $v_{j+1} = w_j / h_{j+1,j}.$

O Algoritmo termina quando $w_j = 0$.

Os vetores v_k são chamados de *vetores de Arnoldi* e definem uma base ortonormal para o subespaço de Krylov $K(A, v_1, m)$:

$$span\{v_1, \ldots, v_m\} = span\{v_1, Av_1, \ldots, A^{m-1}v_1\}.$$

A cada passo, o algoritmo multiplica os vetores de Arnoldi anteriores v_j por A e, então, ortogonaliza o vetor resultante w_j com relação todos os v_j's anteriores por um procedimento Gram-Schmidt padrão.

Proposição 4.1. *Denote por V_m a matriz $n \times m$ com colunas $[v_1, v_2, \ldots, v_m]$ e por H_m a matriz Hessenberg $m \times m$ cujos elementos são definidos pelo algoritmo (4.1). Então, as seguintes relações se verificam:*

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{4.2.2}$$

$$= V_{m+1} \bar{H}_m \tag{4.2.3}$$

$$V_m^T A V_m = H_m. \tag{4.2.4}$$

Demonstração: A relação (4.2.3) resulta observando no algoritmo (4.1), os ítens 2(b), 2(c) e 2(d) que

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i,\ j = 1, 2, \ldots, m; \tag{4.2.5}$$

a relação (4.2.2) é a formulação matricial de (4.2.5), já a relação (4.2.4) é obtida multiplicando ambos os lados de (4.2.2) por V_m^T e usando a ortogonalidade de V_m.

■

	GS	GSM	GSMR	HO
Flops	m^2n	m^2n	$2m^2n$	$2m^2n - \frac{2}{3}m^3$
Armazenamento	$(m+1)n$	$(m+1)n$	$(m+1)n$	$(m+1)n - \frac{1}{2}m^2$

Tabela 4.1: Desempenho das versões do algoritmo de Arnoldi

4.2.2 Implementações práticas

A descrição do processo de Arnoldi dada anteriormente é assumido a aritmética exata. Na realidade, pode-se melhorar o processo numericamente usando o Gram-Schmidt Modificado (GSM) ou o algoritmo Householder [21] no lugar do Gram-Schmidt padrão (GS). Com o GSM, o algoritmo fica da seguinte maneira:

Algoritmo 4.2. Arnoldi-GSM

1. *Escolha um vetor* v_1 *com norma 1.*
2. *Para* $j = 1, 2, \ldots, m$ *do*
   (a) $w_j := Av_j$,
   (b) *Para* $i = 1, 2, \ldots, j$ *faça*
      (i) $h_{ij} =< w_j, v_i >$,
      (ii) $w_j = r_j - h_{ij}v_i$,
   (c) $h_{j+1,j} = \|w_j\|_2$,
   (d) $v_{j+1} = w_j/h_{j+1,j}$.

Não há diferença em aritmética exata entre este algoritmo e o algoritmo (4.1), embora o último seja numericamente melhor do que o Gram-Schmidt padrão. Apesar disso, o algoritmo GSM pode não ser suficiente para todos os casos, sendo necessário recorrer para uma ortogonalização dupla. Uma outra alternativa é usar as técnicas de ortogonalização baseadas no algoritmo de Householder, sendo estas as mais confiáveis do ponto de vista numérico. O custo computacional e o armazenamento de cada uma das três versões [21] são apresentados na tabela 4.1. O GSMR é o GSM com reortogonalização e o HO com Householder.

O número de operações mostrado no GSMR é para o pior caso, desempenhando uma segunda ortogonalização a todo momento. Na prática, o número de operações fica próximo do simples GSM. O pequeno ganho no armazenamento, necessário na versão Householder, vem do fato que as transformações de Householder requerem vetores cuja dimensão diminui por um a cada passo do processo.

4.3 Método de Arnoldi para sistemas lineares

Dado um aproximação inicial x_0 para o sistema $Ax = b$, deseja-se a cada passo m uma aproximação x_m que satisfaça as condições:

$$x_m \in x_0 + K_m(A, r_0) \in \mathbb{R}^m; \tag{4.3.1}$$

$$r_m = b - Ax_m \perp K_m(A, r_0); \tag{4.3.2}$$

onde $r_0 = b - Ax_0$.

Se $v_1 = r_0/\|r_0\|_2$, e $\beta = \|r_0\|_2$ no método de Arnoldi, tem-se

$$V_m^T A V_m = H_m$$

por (4.2.4), assim,

$$V_m^T r_0 = V_m^T(\beta v_1) = \beta e_1.$$

Pela condição (4.3.1),

$$x_m = x_0 + V_m y_m \implies \tag{4.3.3}$$

$$-r_m = -r_0 + AV_m y_m, \tag{4.3.4}$$

impondo a condição (4.3.2), tem-se

$$V_m^T A V_m y_m = V_m^T r_0 = \beta e_1 \implies \tag{4.3.5}$$

$$y_m = H_m^{-1} \beta e_1 \tag{4.3.6}$$

Estas equações definem o método de Ortogonalização Completa (FOM), descrito abaixo. Gram-Schmidt modificado é usado no passo de Arnoldi.

Algoritmo 4.3. *Método de Ortogonalização Completa - FOM*

1. *Calcule* $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$ *e* $v_1 := r_0/\beta$
2. *Defina a matriz* $m \times m$ $H_m = \{h_{ij}\}_{i,j=1,\ldots,m}$ *Faça* $H_m = 0$
3. *Para* $j = 1, 2, \ldots, m$ *faça*
   (a) *Calcule* $w_j := Av_j$
   (b) *Para* $i = 1, \ldots, j$ *faça*

(i) $h_{ij} =< w_j, v_i >$

(ii) $w_j := w_j - h_{ij}v_i$

(c) *Calcule* $h_{j+1,j} = \|w_j\|_2$. *Se* $h_{j+1,j} = 0$ *faça* $m := j$ *e vá para* 12

(d) *Calcule* $v_{j+1} = w_j/h_{j+1,j}$.

4. *Calcule* $y_m = H_m^{-1}(\beta e_1)$ *e* $x_m = x_0 + V_m y_m$

É importante a disponibilidade do resíduo, a cada passo j, de maneira econômica, de modo que o algoritmo possa ser terminado, se uma precisão desejada do resíduo for alcançada. A seguinte proposição dá um resultado nessa direção.

Proposição 4.2. *O vetor residual de uma solução aproximada* x_m *gerada pelo algoritmo FOM é tal que*

$$b - Ax_m = -h_{m+1,m}e_m^T y_m v_{m+1}$$

e assim

$$\|b - Ax_m\| = h_{m+1,m}|e_m^T y_m|. \tag{4.3.7}$$

Demonstração:

$$\begin{aligned} b - Ax_m = b - A(x_0 + V_m y_m) &= r_0 - AV_m y_m \\ &= r_0 - V_m H_m y_m - h_{m+1,m}e_m^T y_m v_{m+1} \\ &= \beta v_1 - V_m H_m y_m - h_{m+1,m}e_m^T y_m v_{m+1} \end{aligned}$$

De $H_m y_m = \beta e_1$, tem-se que $\beta v_1 - V_m H_m y_m = 0$, de onde segue a proposição.

■

4.3.1 FOM com reinício

O algoritmo FOM torna-se computacionalmente impraticável à medida que m aumenta devido à ortogonalização. O custo de armazenamento aumenta na ordem de mn. Existem duas técnicas para contornar este problema. A primeira, é reiniciar o algoritmo periodicamente e a segunda, truncar a ortogonalização no algoritmo de Arnoldi. Nesta seção será abordada a primeira opção. Para a segunda opção veja [21]. O algoritmo FOM com reinício é descrito abaixo:

Algoritmo 4.4. *FOM com reinício - FOM(p):*

1. *Calcule* $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, *e* $v_1 = r_0/\beta$
2. *Gerar a base de Arnoldi e a matriz* H_p *usando o algoritmo de Arnoldi começando com* v_1
3. *Calcule* $y_p = H_p^{-1}(\beta e_1)$ *e* $x_p = x_0 + V_p y_p$. *Se convergiu então PARE.*
4. *Escolha* $x_0 = x_p$ *e vá para 1.*

4.4 GMRES

O método GMRES busca, a cada passo m, uma solução x_m tal que

$$x_m \in x_0 + K_m(A, v_1) \text{ e}$$

$$b - Ax_m \perp AK_m(A, v_1);$$

onde $v_1 = r_0/\|r_0\|_2$. As idéias básicas do GMRES são descritas abaixo.

Qualquer vetor x em $x_0 + K_m$ pode ser escrito como

$$x = x_0 + V_m y, \text{ para algum y} \in \mathbb{R}^m$$

Definido

$$J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2, \tag{4.4.1}$$

a relação (4.2.3) resulta em

$$b - Ax = r_0 - AV_m y = \beta v_1 - V_{m+1}\bar{H}_m y$$

$$= V_{m+1}(\beta e_1 - \bar{H}_m y). \tag{4.4.2}$$

Como a matriz V_m é ortonormal tem-se

$$J(y) = \|r_0 - AV_m y\|_2 = \|\beta e_1 - \bar{H}_m y\|_2. \tag{4.4.3}$$

A aproximação GMRES busca o único vetor de $x_0 + K_m$ que minimiza (4.4.1). Assim, esta aproximação é obtida fazendo $x_m = x_0 + V_m y_m$, onde

$$y_m = \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2. \tag{4.4.4}$$

Para resolver o problema (4.4.4), busca-se a solução de um problema de quadrados mínimos linear simplificado de dimensão $(m+1) \times m$, resolvido sem muitas dificuldades se m é pequeno. Desses resultados obtém-se o seguinte algoritmo:

Algoritmo 4.5. *GMRES*

1. *Calcule* $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ *e* $v_1 = r_0/\beta$
2. *Defina a matriz* $\bar{H}_m = \{h_{ij}\}_{1 \leq i \leq m+1, 1 \leq j \leq m}$. *Faça* $\bar{H}_m = 0$
3. *Para* $j = 1, \ldots, m$ *faça*
   (a) *Calcule* $w_j = Av_j$
   (b) *Para* $i = 1, \ldots, j$ *faça*
      (i) $h_{ij} =< w_j, v_i >$
      (ii) $w_j = w_j - h_{ij} v_i$
   (c) $h_{j+1,j} = \|w_j\|_2$. *Se* $h_{j+1,j} = 0$ *PARE!*
   (d) $v_{j+1} = w_j / h_{j+1,j}$
4. *Calcule* $y_m = argmin_y \|\beta e_1 - \bar{H}_m y\|_2$ *e faça* $x_m = x_0 + V_m y_m$.

No algoritmo acima, os passos 3 e 4 são baseados no processo de ortogonalização de *Gram-Schmidt*. Um algoritmo GMRES numericamente mais robusto pode ser obtido baseando-se no processo de ortogonalização de Householder. Uma desvantagem do algoritmo (4.5) é não fornecer a solução aproximada x_m explicitamente a cada passo, impossibilitando o cálculo do resíduo do sistema e dificultando saber quando parar. Uma maneira elegante e computacionalmente barata de se obter o resíduo está relacionada com a maneira pela qual o problema de quadrados mínimos (PQM) é resolvido.

Para se resolver o PQM (4.4.3), é natural transformar a matriz Hessenberg $\bar{H}_m$ em triangular superior, fazendo uso de matrizes de rotações. Para isso

considere a matriz de rotação

$$\Omega_i = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c_i & s_i & & & \\ & & -s_i & c_i & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow \textit{linha } i \\ \leftarrow \textit{linha } i+1 \\ \\ \\ \\ \end{matrix} \tag{4.4.5}$$

onde $c_i^2 + s_i^2 = 1$. Se a matriz Hessenberg H_m é $(m+1) \times m$ então Ω_i tem dimensão $(m+1)$.

Como deseja-se transformar $\bar{H}_m$ em uma matriz $\bar{H}_m^{(i)}$ triangular superior, basta escolher

$$s_i = \frac{h_{i+1,i}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}, \quad c_i = \frac{h_{ii}^{(i-1)}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}. \tag{4.4.6}$$

Será definido Q_m o produto das matrizes Ω_i

$$Q_m = \Omega_m \Omega_{m-1} \ldots \Omega_1 \tag{4.4.7}$$

e

$$\bar{R}_m = Q_m \bar{H}_m, \tag{4.4.8}$$

$$\bar{g}_m = Q_m(\beta e_1) = (\gamma_1, \ldots, \gamma_{m+1})^T. \tag{4.4.9}$$

Excluindo as últimas linhas de $\bar{R}_m$ e $\bar{g}_m$, obtém-se uma matriz quadrada triangular superior, a qual será designado por R_m, e um vetor $g_m \in \mathbb{R}^m$, respectivamente. Usando o fato de Q_m ser ortogonal, tem-se que

$$\min_y \|\beta e_1 - \bar{H}_m y\|_2 = \min_y \|\bar{g}_m - \bar{R}_m y\|_2. \tag{4.4.10}$$

Como $\|\bar{g}_m - \bar{R}_m y\|_2^2 = |\gamma_{m+1}|^2 + \|g_m - R_m y\|_2^2$, a solução de (4.4.10) é dada por

$$y_m = R_m^{-1} g_m.$$

Proposição 4.3. *Sejam Ω_i as matrizes de rotação usadas para transformar $\bar{H}_m$ em triangular superior. Sejam Q_m, $\bar{R}_m$ e $\bar{g}_m$ como em (4.4.7), (4.4.8) e (4.4.9), respectivamente. Então $r_m = b - Ax_m = \gamma_{m+1} V_{m+1} Q_{m+1} e_{m+1}$, consequentemente, $\|r_m\|_2 = |\gamma_{m+1}|$.*

Demonstração: Usando (4.2.3) tem-se que

$$b - Ax_m = b - A(x_0 + V_m y) = r_0 - V_{m+1}\bar{H}_m y = V_{m+1}(\beta e_1 - \bar{H}_m y) =$$

$$V_{m+1}(\beta e_1 - Q_m^T \bar{R}_m y) = V_{m+1} Q_m^T (\bar{g}_m - \bar{R}_m y).$$

Como y é a solução de (4.4.10), já foi visto que $y = R_m^{-1} g_m$. Assim

$$b - Ax_m = V_{m+1} Q_m^T ([g_m, \gamma_{m+1}]^T - [g_m, 0]^T) = \gamma_{m+1} V_{m+1} Q_{m+1} e_{m+1}.$$

Como V_{m+1} e Q_m são matrizes ortogonais tem-se que $\|r_m\|_2 = |\gamma_{m+1}|$.

■

Conforme a proposição acima, pode-se, a cada passo do GMRES, calcular o resíduo do sistema de uma maneira conveniente, tornando possível interromper o algoritmo no loop intermediário se o critério de parada for satisfeito. Note também que

$$\gamma_{j+1} = -s_j \gamma_j, \tag{4.4.11}$$

assim

$$\|b - Ax_m\|_2 = \beta |s_1 s_2 \dots s_m|, \tag{4.4.12}$$

se $s_j = 0$, então a solução é exata no passo j.

Poderão ocorrer situações, denominadas " break down ", onde o vetor v_j é igual a zero no algoritmo (4.5), isto é, $h_{j+1,j} = 0$ para um certo passo j, impossibilitando calcular o próximo vetor de Arnoldi. Entretanto, conforme [21], isso indica que a solução exata foi obtida.

4.4.1 Comparação teórica entre o FOM e o GMRES

Nesta seção será descrito algumas relações entre "break downs", que podem acontecer em ambos algoritmos, além de estabelecer relações entre os resíduos. Será chamado x_j^F a iteração do FOM e x_j^G a iteração do GMRES.

Primeiro, será discutido um fato que pode interromper o algoritmo FOM, quando a matriz H_m é singular em alguma iteração, impossibilitando calcular o iterado seguinte. Isto é um sério caso de "break down", podendo interromper a convergência do algoritmo FOM ou causar um fenômeno chamado "estagnação"no GMRES, quando $x_{m+1}^G = x_m^G$. Uma possível solução seria tomar alguma matriz H_j não singular, $1 \leq j \leq m-1$, calcular x_j^F e reiniciar o algoritmo com $x_0 = x_j^F$. Nem sempre é possível fazer o procedimento acima, como mostrado num exemplo em [7].

Proposição 4.4. *Suponha que m passos do FOM tenham sido realizados, e assuma que H_m é singular. Então*

$$\min_{y \in \mathbb{R}^m} \|\beta e_1 - \bar{H}_m y\| = \min_{y \in \mathbb{R}^{m-1}} \|\beta e_1 - \bar{H}_{m-1} y\|. \tag{4.4.13}$$

Se y_k é a solução de (4.4.4), $k = m$ ou $k = m-1$, então $y_m^G = (y_{m-1}^G, 0)^T$, assim $x_{m-1}^G = x_m^G$. Reciprocamente, se m passos do FOM têm sido tomados e (4.4.13) se verifica, então H_m é singular.

Demonstração: Veja BROWN [7]

■

Esta proposição mostra como o desempenho do FOM e do GMRES estão relacionados. Se H_m é singular, então x_m^F não existe e $x_m^G = x_{m-1}^G$. O contrário também é verdade. Assim, ou ambos métodos progridem, ou ambos falham. Uma importante observação da proposição (4.4) é que se o algoritmo GMRES apresentar um baixo decaimento no resíduo de uma iteração para outra, ou seja, $r_m^G \approx r_{m-1}^G$, significa que a matriz H_m está próxima da singularidade e assim é mal condicionada.

Será estabelecido a seguir interessantes relações entre os resíduos do FOM e o do GMRES. Primeiramente, será discutido uma variação do FOM a qual se usa a fatoração QR da matriz Hessemberg superior H_m para resolver (4.3.6), ao invés da fatoração LU.

Há uma interessante relação entre H_m e $\bar{H}_{m-1}$

$$H_m = [\bar{H}_{m-1}, h_m],$$

onde $h_m = (h_{1,m}, \ldots, h_{m,m})^T$. Usando a fatoração QR de $\bar{H}_{m-1}$, obtém-se a correspondente fatoração de H_m.

$$\begin{aligned} H_m = [\bar{H}_{m-1}, h_m] &= [Q_{m-1}\bar{R}_{m-1}, h_m] \\ &= Q_{m-1}[\bar{R}_{m-1}, Q_{m-1}^T h_m] \\ &= Q_{m-1}R_m, \end{aligned}$$

sendo que $R_m = [\bar{R}_{m-1}, Q_{m-1}^T h_m] \in \mathbb{R}^{m\times m}$. Assumindo que H_m é não singular, a solução do sistema (4.3.6) pode ser encontrada resolvendo o sistema triangular superior

$$R_m y = \beta Q_{m-1}^T e_1 = \beta \begin{pmatrix} q_{1,m-1}^T e_1 \\ \vdots \\ q_{m,m-1}^T e_1 \end{pmatrix}. \tag{4.4.14}$$

Seja, y_m^F a solução deste sistema. Pela proposição (4.2) tem-se

$$\|b - Ax_m^F\|_2 = h_{m+1,m}|e_m^T y_m^F|,$$

usando argumentos similares à (4.4.11) e observando (4.4.14) tem-se

$$|e_m^T y_m^F| = \beta |s_1 \ldots s_{m-1}/r_{m,m}|. \tag{4.4.15}$$

Usando (4.4.15), obtém-se um importante resultado entre as normas dos resíduos para o FOM e o GMRES.

Proposição 4.5. *Suponha que m passos do FOM tenham sido executados e que H_m é não singular. Então*

$$\|r_m^F\| = \|r_m^G\| \cdot \sqrt{1 + (h_{m+1,m}/r_{m,m})^2} \tag{4.4.16}$$

Demonstração: Usando (4.4.15) tem-se

$$\|r_m^F\| = \beta|s_1 \dots s_{m-1}|h_{m+1,m}/|r_{m,m}| = \beta|s_1 \dots s_{m-1}s_m|\frac{h_{m+1,m}}{|s_m||r_{m,m}|}. \quad (4.4.17)$$

De (4.4.6), tem-se que $|s_m| = \frac{h_{m+1,m}}{\sqrt{r_{m,m}^2+h_{m+1,m}^2}}$, assim usando (4.4.12)

$$\|r_m^F\|_2 = \|r_m^G\|_2 \cdot \sqrt{1 + (h_{m+1,m}/r_{m,m})^2}.$$

■

Uma outra relação pode ser obtida de (4.4.12) e (4.4.15)

$$\|r_m^F\|_2 = \frac{h_{m+1,m}}{|r_{m,m}|}\|r_{m-1}^G\|_2,$$

de (4.4.16)

$$\frac{h_{m+1,m}}{|r_{m,m}|} = \sqrt{\frac{\|r_m^F\|_2^2}{\|r_m^G\|_2^2} - 1}$$

e então

$$\frac{\|r_m^F\|_2^2}{\|r_{m-1}^G\|_2^2} = \frac{\|r_m^F\|_2^2}{\|r_m^G\|_2^2} - 1,$$

de onde segue que

$$\|r_m^F\|_2 = \|r_m^G\|_2 c_m^{-1} = \|r_m^G\|_2(1 - s_m^2)^{-1/2}. \quad (4.4.18)$$

Este último resultado mostra que a norma do resíduo do FOM cresce na medida que $s_m \to 1$, o que corresponde a estagnação no GMRES.

4.4.2 GMRES com reinício

Similarmente ao algoritmo FOM, o GMRES torna-se impraticável quando m cresce, aumentando custo de armazenamento e número de operações. Uma maneira eficaz de contornar este fato é reiniciar o algoritmo após um certo número de iterações, o que é descrito a seguir.

Algoritmo 4.6. *GMRES com reinício*

1. *Calcule* $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, $v_1 = r_0/\beta$ *e escolha* $m \in \mathbb{N}$

2. *Use o algoritmo de Arnoldi para gerar* m *vetores de Arnoldi e a matriz* $\bar{H}_m$ *começando com* v_1

3. *Calcule* y_m *que minimiza* $\|\beta e_1 - \bar{H}_m y\|_2$ *e faça* $x_m = x_0 + V_m y_m$

4. *Se critério é satisfeito, PARE. Senão escolha* $x_0 := x_m$ *e vá para* 1.

Uma dificuldade eventual do GMRES com reinício é a estagnação, o que ocorre quando a matriz não é positiva definida. O GMRES padrão converge em no máximo n iterações, o que é inviável para n grande. Neste caso, um precondicionador deve ser usado para reduzir o número de iterações.

4.5 Análise de convergência do GMRES

Os polinômios de Chebyshev tem papel na teoria de convergência do GMRES. Essa classe de polinômios, além de ser usada para estudar a convergência de alguns métodos iterativos, ainda pode ser usadas na prática, com o intuito de acelerar as iterações ou o processo de projeção.

4.5.1 Polinômios de Chebyshev

Caso real:

Os polinômios de Chebyshev reais de primeiro tipo de grau k são definidos por:

$$C_k(t) = cos(kcos^{-1}(t)),\ t \in [-1, 1]. \tag{4.5.1}$$

Usando a relação trigonométrica

$$cos[(k+1)\theta] + cos[(k-1)\theta] = 2cos\theta cosk\theta,$$

e o princípio de indução, verifica-se facilmente que (4.5.1) é um polinômio em t. Essa relação trigonométrica também fornece uma importante relação de recorrência de três termos

$$C_{k+1}(t) = 2tC_k(t) - C_{k-1}(t), \tag{4.5.2}$$

onde $C_0 = 1,\ C_1 = t$.

Pode-se estender a definição (4.5.1) para os casos onde $|t| \geq 1$ através da fórmula

$$C_k(t) = cosh(kcosh^{-1}(t)),\ |t| \geq 1. \tag{4.5.3}$$

A partir da definição acima, obtém-se a seguinte expressão:

$$C_k(t) = \frac{1}{2}[(t + \sqrt{t^2 - 1})^k + (t + \sqrt{t^2 - 1})^{-k}], \tag{4.5.4}$$

válida para $|t| \geq 1$, que também pode ser estendida para $|t| < 1$. Quando k é grande o segundo termo de (4.5.4) torna-se pequeno, fornecendo a aproximação:

$$C_k(t) \approx \frac{1}{2}(t + \sqrt{t^2 - 1})^k,\ |t| \geq 1. \tag{4.5.5}$$

Caso complexo

Como visto antes, quando $|t| \geq 1$, $C_k(t) = cosh(kcosh^{-1}(t))$. Essa definição pode ser unificada para o caso complexo. Assim,

$$C_k(z) = cosh(k\xi),\ \text{ onde } cosh(\xi) = z.$$

Definindo a variável $w \equiv e^{\xi}$, a fórmula acima pode ser escrita como

$$C_k(z) = \frac{1}{2}[w^k + w^{-k}] \text{ onde } z = \frac{1}{2}[w + w^{-1}]. \tag{4.5.6}$$

Esta é a definição de polinômios de Chebyshev usada em $\mathbb{C}$. Como antes, verifica-se que os C_k's são realmente polinômios e satisfazem a recorrência de três termos

$$\begin{aligned} &C_{k+1}(z) = 2zC_k(z) - C_{k-1}(z), \\ &C_0(z) = 1, \quad C_1(z) = z. \end{aligned} \tag{4.5.7}$$

Esses tipos de polinômios estão diretamente ligados com elipses no plano complexo. Seja C_ρ o círculo centrado na origem de raio ρ. Definindo $J : C_\rho \longrightarrow \mathbb{C}$ por

$$J(w) = \frac{1}{2}[w + w^{-1}],$$

chamada aplicação de Joukowski, que transforma C_ρ em uma elipse centrada na origem com focos -1, 1 e semi-eixos principais $\frac{1}{2}[\rho + \rho^{-1}]$ e $\frac{1}{2}[\rho - \rho^{-1}]$.

Os Polinômios de Chebyshev são assintoticamente ótimos, sendo ótimos em apenas alguns casos. Para provar isso, será considerado o lema seguinte, devido a Zarantonello. Considera-se $\mathbb{P}_k$ o conjunto de todos polinômios de grau k.

Lema 4.1. *Seja $C(0,\rho)$ um círculo centrado na origem e raio ρ e seja $\gamma \in \mathbb{C}$ e $\notin C(0,\rho) \cup$ int$(C(0,\rho))$. Então*

$$\min_{p\in\mathbb{P}_k, p(\gamma)=1} \max_{z\in C(0,\rho)} |p(z)| = \left(\frac{\rho}{|\gamma|}\right)^k, \tag{4.5.8}$$

o mínimo é alcançado pelo polinômio $(z/\gamma)^k$.

Demonstração: Veja RIVLIN [20]

■

O resultado anterior pode ser estendido para qualquer círculo centrado em c e raio ρ e para qualquer γ tal que $\gamma > \rho$, apenas mudando variáveis e reescalando o polinômio.

Será considerado a seguir o caso de uma elipse centrada na origem com foco 1, -1 e semi-eixo principal a, a qual pode ser considerada como a imagem de J do círculo $C(0,\rho)$. Denote por E_ρ tal elipse.

Proposição 4.6. *Seja $E_\rho = J(C(0,\rho))$ a elipse como acima, e seja $\gamma \in \mathbb{C}$ como no lema (4.1). Então*

$$\frac{\rho^k}{|w_\gamma|^k} \leq \min_{p\in\mathbb{P}_k, p(\gamma)=1} \max_{z\in E_\rho} |p(z)| \leq \frac{\rho^k + \rho^{-k}}{|w_\gamma^k + w_\gamma^{-k}|} \tag{4.5.9}$$

onde w_γ é a raiz dominante da equação $J(w) = \gamma$

Demonstração: Veja SAAD [21].

■

Quando $k \to \infty$ a diferença entre os limitantes direito e esquerdo de (4.5.9) tende a zero. Assim, um ponto importante, devido à proposição, é que, para

k grande, o polinômio de Chebyshev

$$p^*(z) = \frac{w^k + w^{-k}}{w_\gamma^k + w_\gamma^{-k}}, \text{ onde } z = \frac{w + w^{-1}}{2},$$

está próximo do polinômio ótimo, ou ainda, polinômios de Chebyshev são assintoticamente ótimos.

Novamente, pode-se estender o resultado para uma elipse $E(c, d, a)$ centrada em c, distância focal d e semi-eixo principal a fazendo uma simples mudança de variável, mostrando que o polinômio de Chebyshev mais próximo do ótimo é dado por

$$\widehat{C}_k(z) = \frac{C_k(\frac{c-z}{d})}{C_k(\frac{c-\gamma}{d})}. \tag{4.5.10}$$

Examinado a expressão $(w^k + w^{-k})/2$ para $w = \rho e^{i\theta}$, verifica-se que o máximo de $|\widehat{C}_k(z)|$ sobre a elipse, é alcançado no ponto $c + a$ localizado no eixo real [21]. Assim

$$\max_{z \in E(c,d,a)} |\widehat{C}_k(z)| = \frac{C_k(\frac{a}{d})}{C_k(\frac{c-\gamma}{d})}, \tag{4.5.11}$$

Assim, a convergência para o algoritmo GMRES pode ser demonstrada. Um resultado de convergência global é o primeiro passo.

Proposição 4.7. *Se A é positiva definida, então o algoritmo GMRES(m) converge para qualquer $m \geq 1$.*

Demonstração: Veja SAAD [21]

■

Seria interessante o conhecimento de um limitante superior para a taxa de convergência do GMRES. É disso que se trata o lema seguinte.

Lema 4.2. *Seja x_m a solução aproximada obtida do m-ésimo passo do algoritmo GMRES e tome $r_m = b - Ax_m$. Então x_m é da forma*

$$x_m = x_0 + q_{m-1}(A)r_0$$

e

$$\|r_m\|_2 = \|(I - Aq_{m-1}(A))r_0\| = \min_{q \in \mathbb{P}^{m-1}} \|(I - Aq(A))r_0\|_2.$$

Demonstração: Veja SAAD [21]

■

Proposição 4.8. *Suponha que $A \in \mathbb{R}^{n \times n}$ é uma matriz diagonalizável, assim seja $A = X\Lambda X^{-1}$ onde $\Lambda = diag\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ é a matriz diagonal de autovalores. Defina*

$$\epsilon^{(m)} = \min_{p \in \mathbb{P}^m, p(0)=1} \max_{i=1,\ldots,n} |p(\lambda_i)|.$$

Então, a norma do resíduo do m-ésimo passo do GMRES satisfaz

$$\|r_m\|_2 \leq k_2(X)\epsilon^{(m)}\|r_0\|_2.$$

onde $k_2(X) = \|X\|_2\|X^{-1}\|_2$.

Demonstração: Veja SAAD [21]

■

Os resultados das seções anteriores podem ser usados para obter um majorante para $\epsilon^{(m)}$. Suponha que o espectro da matriz A esteja contido em uma elipse $E(c, d, a)$ com centro c, distância focal d e semi-eixo principal a e que a origem está fora desta elipse. O corolário a seguir fornece um majorante para o resíduo do GMRES.

Corolário 4.1. *Seja A uma matriz diagonalizável, assim tome $A = X\Lambda X^{-1}$ onde $\Lambda = diag\{\lambda_1, \ldots, \lambda_n\}$ é a matriz diagonal de autovalores. Assuma que todos os autovalores de A estão contido em $E(c, d, a)$ que exclui a origem. Então, a norma residual formada no m-ésimo passo do GMRES satisfaz a desigualdade*

$$\|r_m\|_2 \leq k_2(X)\frac{C_k(\frac{a}{d})}{|C_k(\frac{c}{d})|}\|r_0\|_2.$$

Demonstração: Como $\{\lambda_i\}_{i=1...n} \in \mathbb{C}$,

$$\epsilon^{(m)} = \min_{p\in\mathbb{P}^m, p(0)=1} \max_{i=1,\dots,n} |p(\lambda_i)| \leq \min_{p\in\mathbb{P}^m, p(0)=1} \max_{z\in E(c,d,a)} |p(z)|$$

por (4.5.11), onde $\gamma = 0$ e, pela observação acima,

$$\epsilon^{(m)} \leq \frac{C_k(\frac{a}{d})}{|C_k(\frac{c}{d})|}$$

Usando a proposição anterior, obtém-se o resultado desejado.

■

A seguir, descreve-se os métodos iterativos em espaços de Krylov baseados no processo de biortogonalização de Lanczos. Estes são métodos de projeção que são intrinsicamente não ortogonais, possuindo algumas propriedades interessantes e com uma análise teórica bem elaborada.

4.6 Biortogonalização de Lanczos

O algoritmo da biortogonalização de Lanczos é uma extensão para matrizes não simétricas do algoritmo Lanczos [22] simétrico, que é equivalente ao algoritmo de Arnoldi para matrizes simétricas. O processo do Lanczos não simétrico é completamente diferente ao de Arnoldi, pois busca uma sequência biortogonal ao invés de uma sequência ortogonal.

O algoritmo proposto por Lanczos para matrizes não simétricas constrói um par de bases biortogonais para dois subespaços

$$K_m(A, v_1) = span\{v_1, Av_1, \dots A^{m-1}v_1\}$$

e

$$K_m(A^T, w_1) = span\{w_1, A^Tw_1, \dots (A^T)^{m-1}w_1\}.$$

conforme o algoritmo a seguir.

Algoritmo 4.7. *Biortogonalização de Lanczos*

1. *Escolha dois vetores* v_1, w_1 *tal que* $< v_1, w_1 >= 1$

2. *Escolha* $\beta_1 = \delta_1 = 0$, $w_0 = v_0 = 0$

3. *Para* $j = 1, 2, \ldots, m$ *faça*

   (a) $\alpha_j =< Av_j, w_j >$

   (b) $\widehat{v}_{j+1} = Av_j - \alpha v_j - \beta_j v_{j-1}$

   (c) $\widehat{w}_{j+1} = A^T w_j - \alpha w_j - \delta_j w_{j-1}$

   (d) $\delta_{j+1} = | < \widehat{v}_{j+1}, \widehat{w}_{j+1} > |^{1/2}$. *Se* $\delta_{j+1} = 0$ *então PARE!*

   (e) $\beta_{j+1} =< \widehat{v}_{j+1}, \widehat{w}_{j+1} > /\delta_{j+1}$

   (f) $w_{j+1} = \widehat{w}_{j+1}/\beta_{j+1}$

   (g) $v_{j+1} = \widehat{v}_{j+1}/\delta_{j+1}$

Os escalares δ_{j+1}, β_{j+1}, definidos nas linhas 3(d) e 3(e), são para garantir que $< v_{j+1}, w_{j+1} >= 1$. Como resultado das linhas 3(f) e 3(g) do algoritmo (4.7), é necessário apenas escolher δ_{j+1}, β_{j+1} de modo que

$$\delta_{j+1}\beta_{j+1} =< \widehat{v}_{j+1}, \widehat{w}_{j+1} > . \tag{4.6.1}$$

Será denotado por T_m a matriz tridiagonal

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{bmatrix},$$

sendo que β_{j+1}, δ_{j+1} satisfaz a relação (4.6.1). Se forem usados estes escalares como no algoritmo (4.7), então δ_j's são positivos e $\beta_j = \pm\delta_j$. Observe do algoritmo que os vetores v_i's estão em $K_m(A, v_1)$ e os w_i's em $K_m(A^T, w_1)$. Isto é confirmado na proposição a seguir.

Proposição 4.9. *Se o algoritmo (4.7) não interrompe o processo até o passo* m, *então os vetores* $\{v_i\}_{i=1,\ldots,m}$ *e* $\{w_j\}_{j=1\ldots,m}$ *formam um sistema biortogonal, isto é,*

$$< v_i, w_j >= \begin{cases} 1, & se\ i = j \\ 0, & se\ i \neq j \end{cases} \quad 1 \leq i, j \leq m.$$

Além disso, $\{v_i\}_{i=1,\ldots,m}$ *é uma base para* $K_m(A, v_1)$ *e* $\{w_j\}_{j=1,\ldots,m}$ *é uma base para* $K_m(A^T, w_1)$ *sendo que as seguintes relações são válidas,*

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T, \tag{4.6.2}$$

$$A^T W_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T, \tag{4.6.3}$$

$$W_m^T A V_m = T_m. \tag{4.6.4}$$

Demonstração: Veja SAAD [21]

■

Existem vantagens e desvantagens do algoritmo Lanczos sobre o algoritmo Arnoldi. O Lanczos requer pouco armazenamento de vetores, tendo uma significante vantagem sobre o Arnoldi. Por outro lado, possui maior potencialidade de ocorrer um "break down", sempre que no algoritmo (4.7), linha 3(d) se verificar

$$< \widehat{v}_{j+1}, \widehat{w}_{j+1} >= 0. \tag{4.6.5}$$

Isto ocorre quando um dos dois vetores $\widehat{v}_{j+1}$, $\widehat{w}_{j+1}$ se anula, ou quando ambos são não nulos e o produto interno é nulo. Se ocorrer o primeiro caso, com $\widehat{v}_{j+1} = 0$ então $span\{V_j\}$ é invariante e, assim, a solução aproximada é exata; se $\widehat{w}_{j+1} = 0$ então $span\{W_j\}$ é invariante e nada pode ser afirmado sobre a solução aproximada para o sistema com A, apenas que é exata para o sistema dual (A^T). O segundo tipo de "break down" é mais grave e, a primeira vista, impossibilita calcular a iteração seguinte. Felizmente, existem modificações do algoritmo que permitem sua continuação em muitos casos. Essas modificações consistem nos denominados *algoritmos de Lanczos look-ahead.*

A principal idéia do *look-ahead* é definir o par de vetores v_{j+2}, w_{j+2}, mesmo que o par v_{j+1}, w_{j+1} não esteja definido. Se o par v_{j+2}, w_{j+2} não pode ser definido, então tenta-se v_{j+3}, w_{j+3} e assim por diante. Para mais detalhes veja [22].

4.6.1 O algoritmo Lanczos e sistemas lineares

Considere o seguinte sistema linear

$$Ax = b,\ A \in \mathbb{R}^{n \times n} \text{ não simétrica,} \tag{4.6.6}$$

Seja x_0 uma aproximação inicial e tome $r_0 = b - Ax_0$. O algoritmo Lanczos para resolver (4.6.6) é descrito a seguir.

Algoritmo 4.8. *Algoritmo Lanczos para sistemas lineares não simétricos*

1. *Calcule* $r_0 = b - Ax_0$ *e* $\beta = \|r_0\|_2$
2. *Faça m passos do algoritmo Lanczos, ou seja,*
   (a) *Seja* $v_1 = r_0/\beta$ *e escolha* w_1 *tal que* $< v_1, w_1 >= 1$
   (b) *Gere os vetores de Lanczos* $v_1, \ldots, v_m, w_1, \ldots, w_m$
   (c) *Considere a matriz* T_m *do algoritmo (4.7)*
3. *Calcule* $y_m = T_m^{-1}(\beta e_1)$ *e* $x_m = x_0 + V_m y_m$

Como no algoritmo FOM, pode-se calcular a norma do resíduo a cada iteração, o que é firmado pela proposição a seguir.

Proposição 4.10. *O vetor residual de uma solução aproximada* x_m *gerada pelo algoritmo (4.8) é dado por*

$$b - Ax_j = -\delta_{j+1} e_j^T y_j v_{j+1}. \tag{4.6.7}$$

Demonstração: Pelo algoritmo (4.8), tem-se que $x_j = x_0 + V_j y_j$, assim, usando (4.6.2) obtém-se

$$\begin{aligned} b - Ax_j = b - A(x_0 + V_j y_j) = b - Ax_0 - AV_j y_j = \\ r_0 - V_j T_j y_j - \delta_{j+1} e_j^T y_j v_{j+1} = -\delta_{j+1} e_j^T y_j v_{j+1}. \end{aligned}$$

■

Como consequência da proposição acima, tem-se

$$\|b - Ax_j\|_2 = |\delta_{j+1} e_j^T y_j| \|v_{j+1}\|_2. \tag{4.6.8}$$

4.6.2 Os algoritmos Bi-CG e QMR

Do algoritmo (4.8), pode-se derivar o algoritmo Bi-Gradiente Conjugado (Bi-CG), proposto por Lanczos em 1952 e o Quasi-Mínimo Resíduo (QMR). A seguir, ambos os algoritmos são explicados resumidamente.

O algoritmo Bi-CG:

O algoritmo consiste em um processo de projeção em

$$K_m = span\{v_1, Av_1, \ldots, A^{m-1}v_1\}$$

ortogonalmente a

$$\mathcal{L}_m = span\{w_1, A^T w_1, \ldots, (A^T)^{m-1}w_1\}$$

sendo que $v_1 = r_0/\|r_0\|_2$ e w_1 tal que $< v_1, w_1 > \neq 0$, geralmente escolhe-se igual a v_1.

Seja

$$T_m = L_m U_m \tag{4.6.9}$$

a decomposição LU de T_m e

$$P_m = V_m U_m^{-1}, \quad z_m = L_m^{-1}(\beta e_1). \tag{4.6.10}$$

A solução aproximada é escrita por:

$$\begin{aligned} x_m &= x_0 + V_m T_m^{-1}(\beta e_1) \\ &= x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1) \\ &= x_0 + P_m L_m^{-1}(\beta e_1) = x_0 + P_m z_m. \end{aligned}$$

A partir das equações acima, pode-se atualizar x_{m+1} a partir de x_m e p_{m+1} a partir de p_m.

Considere a matriz

$$P_m^* = W_m (L_m^{-1})^T. \tag{4.6.11}$$

Observa-se que os vetores coluna p_i^* de P_m^* e os p_i de P_m são A-conjugados, pois

$$(P_m^*)^T A P_m = L_m^{-1} W_m^T A V_m U_m^{-1} = L_m^{-1} T_m U_m^{-1} = I.$$

A partir destas informações, pode-se derivar o Bi-CG do processo de Lanczos.

Algoritmo 4.9. *Bi-Gradiente Conjugado (Bi-CG)*

1. *Calcule* $r_0 = b - Ax_0$ *e escolha* r_0^* *tal que* $< r_0, r_0^* > \neq 0$

2. *Escolha* $p_0 = r_0,\ p_0^* = r_0^*$

3. *Para* $j = 0, 1, \ldots$, *até convergir faça*

   (a) $\alpha_j = < r_j, r_j^* > / < Ap_j, p_j^* >$

   (b) $x_{j+1} = x_j + \alpha_j p_j$

   (c) $r_{j+1} = r_j - \alpha_j A p_j$

   (d) $r_{j+1}^* = r_j^* - \alpha A^T p_j$

   (e) $\beta_j = < r_{j+1}, r_{j+1}^* > / < r_j, r_j^* >$

   (f) $p_{j+1} = r_{j+1} + \beta_j p_j$

   (g) $p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*$

Note que, se for de interesse resolver também o sistema dual, deve-se definir na linha 1 $r_0 = b^* - A^T x_0^*$ e atualizar $x_{j+1}^* = x_j^* + \alpha_j p_j^*$ depois da linha 5.

O algoritmo QMR:

Do algoritmo de Lanczos tem-se que

$$AV_m = V_{m+1}\bar{T}_m \tag{4.6.12}$$

onde $\bar{T}_m \in \mathbb{R}^{(m+1)\times m}$ é uma matriz tridiagonal

$$\bar{T}_m = \begin{pmatrix} T_m \\ \delta_{m+1} e_m^T \end{pmatrix}$$

Definindo-se o vetor $v_1 = r_0/\beta$ e usando o fato que $x_m = x_0 + V_m y$, tem-se

$$b - Ax = b - A(x_0 + V_m y) = r_0 - AV_m y = \beta v_1 - V_{m+1}\bar{T}_m y = V_{m+1}(\beta e_1 - \bar{T}_m y),$$

assim,

$$\|b - Ax_m\|_2 = \|V_{m+1}(\beta e_1 - \bar{T}_m y)\|_2. \tag{4.6.13}$$

Se a matriz V_{m+1} for ortonormal, tem-se $\|b - Ax_m\|_2 = \|\beta e_1 - \bar{T}_m y\|_2$, resultando em um problema de quadrados mínimos linear, como no GMRES. A idéia do Quasi-Mínimo Resíduo (QMR) é tomar o argumento y que minimiza a função

$$J(y) \equiv \|\beta e_1 - \bar{T}_m y\|_2$$

sobre y, e então calcular a solução aproximada $x_0 + V_m y$. Assim se $y_m = \text{argmin}_y J(y)$, tem-se $x_m = x_0 + V_m y_m$. O algoritmo QMR é bastante similar ao GMRES, a única diferença é que o Arnoldi é trocado pelo Lanczos.

Por causa da estrutura tridiagonal da matriz $\bar{T}_m$, pode-se atualizar a solução x_m a partir de x_{m-1}, como mostra o algoritmo abaixo.

Algoritmo 4.10. *Algoritmo QMR*

1. *Calcule* $r_0 = b - Ax_0$ *e* $\gamma_1 = \|r_0\|_2$, $w_1 = v_1 = r_0/\gamma_1$
2. *Para* $m = 1 \ldots$, *até convergência faça*
   - (a) *calcule* α_m, δ_{m+1} *e* v_{m+1}, w_{m+1} *como no algoritmo (4.7)*
   - (b) *Atualize a fatoração QR de* $\bar{T}_m$, *ou seja*
     - (i) *Aplique as rotações* Ω_i, $i = m-2, m-1$ *para a m-ésima coluna de* $\bar{T}_m$
     - (ii) *Calcule os coeficientes de rotação* c_m, s_m *por (4.4.6)*
   - (c) *Aplique a rotação* Ω_m *a* $\bar{T}_m$ *e* $\bar{g}_m$, *isto é, calcule:*
     - (i) $\gamma_{m+1} = -s_m \gamma_m$
     - (ii) $\gamma_m = c_m \gamma$
     - (iii) $\alpha_m = c_m \alpha_m + s_m \delta_{m+1}$
   - (d) $p_m = (v_m - \sum_{i=m-2}^{m-1} t_{im} p_i)/t_{mm}$
   - (e) $x_m = x_{m-1} + \gamma_m p_m$
   - (f) *Se* $|\gamma_{m+1}|$ *é suficiente pequeno, PARE!*

4.6.3 Variações da biortogonalização de Lanczos

Os métodos Bi-CG e QMR requerem, a cada passo, um produto matriz vetor envolvendo A^T, sendo que nem sempre esta é disponível. Como os vetores

p_i^*, w_j gerados com A^T não contribuem diretamente para a solução aproximada, sendo apenas necessários para obter alguns escalares do algoritmo como α_j e β_j no Bi-CG, existem métodos que ficam livre do uso de A^T. Entre eles, pode-se citar o CGS (Gradiente Conjugado ao Quadrado) e o Bi-CGStab (Bi-Gradiente Conjugado Estabilizado), que serão tratados a seguir.

O algoritmo CGS:

O algoritmo CGS [23] é baseado no Bi-CG, evitando usar a matriz A^T e assim obter uma convergência mais rápida. A idéia central é baseada em uma simples observação. Do algoritmo Bi-CG, o vetor residual na iteração j pode ser espresso como

$$r_j = \phi_j(A)r_0, \tag{4.6.14}$$

sendo ϕ_j um certo polinômio de grau j tal que $\phi_j(0) = 1$. Similarmente, existe um polinômio π_j de grau j tal que

$$p_j = \pi_j(A)r_0. \tag{4.6.15}$$

Observe que r_j^* e p_j^* no algoritmo Bi-CG são definidos com os mesmos escalares na recorrência, sendo que A é substituída por A^T, assim

$$r_j^* = \phi_j(A^T)r_0^*, \quad p_j^* = \pi_j(A^T)r_0^*.$$

Como resultado, tem-se que o escalar α_j no algoritmo é dado por

$$\alpha_j = \frac{< \phi_j(A)r_0, \phi_j(A^T)r_0^* >}{< A\pi_j(A)r_0, \pi_j(A^T)r_0^* >} = \frac{< \phi_j^2(A)r_0, r_0^* >}{< A\pi_j^2(A)r_0, r_0^* >},$$

mostrando que se fosse conhecida uma fórmula de recorrência para os vetores $\phi_j^2(A)r_0$ e $\pi_j^2(A)r_0$, não se teria problemas em calcular α_j e de uma maneira análoga, β_j. A idéia é encontrar uma sequência de iterados cujos os resíduos satisfazem

$$r_j' = \phi_j^2(A)r_0. \tag{4.6.16}$$

Para estabelecer a recorrência desejada, começa-se a recorrência que

define ϕ_j e π_j, que são

$$\phi_{j+1}(t) = \phi_j(t) - \alpha_j t\pi_j(t), \tag{4.6.17}$$
$$\pi_{j+1}(t) = \phi_{j+1}(t) + \beta_j\pi_j(t), \tag{4.6.18}$$

assim,

$$\begin{aligned}\phi^2_{j+1}(t) &= \phi^2_j(t) - 2\alpha_j t\pi_j(t)\phi_j(t) + \alpha_j^2 t^2\pi_j^2(t),\\ \pi^2_{j+1}(t) &= \phi^2_{j+1}(t) + 2\beta_j\phi_{j+1}(t)\pi_j(t) + \beta_j^2\pi_j^2(t).\end{aligned}$$

Se as relações acima não tivessem os termos cruzados $\pi_j(t)\phi_j(t)$ e $\phi_{j+1}(t)\pi_j(t)$ do lado direito, teria-se uma fórmula de recorrência. Uma solução é introduzir um desses termos cruzados, por exemplo $\phi_{j+1}(t)\pi_j(t)$, como um terceiro membro de recorrência. Para o outro termo, $\pi_j(t)\phi_j(t)$, omitindo a variável t, tem-se a seguinte relação

$$\phi_j\pi_j = \phi_j(\phi_j + \beta_{j-1}\pi_{j-1}) = \phi_j^2 + \beta_{j-1}\phi_j\pi_{j-1}.$$

Resumindo todas essas relações, obtém-se as recorrências

$$\phi^2_{j+1} = \phi^2_j - \alpha_j t(2\phi^2_j + 2\beta_{j-1}\phi_j\pi_{j-1} - \alpha_j t\pi^2_j) \tag{4.6.19}$$
$$\phi_{j+1}\pi_j = \phi^2_j + \beta_{j-1}\phi_j\pi_{j-1} - \alpha_j t\pi^2_j \tag{4.6.20}$$
$$\pi^2_{j+1} = \phi^2_{j+1} + 2\beta_j\phi_{j+1}\pi_j + \beta^2_j\pi^2_j. \tag{4.6.21}$$

Definindo

$$r_j = \phi^2_j(A)r_0 \tag{4.6.22}$$
$$p_j = \pi^2_j(A)r_0 \tag{4.6.23}$$
$$q_j = \phi_{j+1}(A)\pi_j(A)r_0, \tag{4.6.24}$$

as recorrências anteriores ficam

$$r_{j+1} = r_j - \alpha_j A(2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j Ap_j), \tag{4.6.25}$$
$$q_j = r_j + \beta_{j-1}q_{j-1} - \alpha_j Ap_j, \tag{4.6.26}$$
$$p_{j+1} = r_{j+1} + 2\beta_j q_j + \beta^2_j p_j. \tag{4.6.27}$$

É conveniente definir alguns vetores auxiliares para simplificar o algoritmo. Seja

$$d_j = 2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j A p_j$$

e

$$u_j = r_j + \beta_{j-1}q_{j-1},$$

assim, tem-se as relações

$$\begin{aligned} d_j &= u_j + q_j, \\ q_j &= u_j - \alpha_j A p_j, \\ p_{j+1} &= u_{j+1} + \beta_j(q_j + \beta_j p_j), \end{aligned}$$

resultando no algoritmo que segue.

Algoritmo 4.11. *Algoritmo CGS*

1. *Calcule* $r_0 = b - Ax_0$; r_0 *arbitrário*
2. *Seja* $p_0 = u_0 = r_0$.
3. *Para* $j = 0, 1, \ldots,$ *até convergir faça*
   (a) $\alpha_j = < r_j, r_j^* > / < Ap_j, r_0^* >$
   (b) $q_j = u_j - \alpha_j A p_j$
   (c) $x_{j+1} = x_j + \alpha_j(u_j + q_j)$
   (d) $r_{j+1} = r_j - \alpha_j A(u_j + q_j)$
   (e) $\beta_j = < r_{j+1}, r_0^* > / < r_j, r_0^* >$
   (f) $u_{j+1} = r_{j+1} + \beta_j q_j$
   (g) $p_{j+1} = u_{j+1} + \beta_j(q_j + \beta_j p_j)$

Uma desvantagem do algoritmo CGS é que, como os polinômios são elevados ao quadrado, os erros de arredondamento tornam-se mais danosos no algoritmo Bi-CG. Em particular, ocorre grandes variações dos vetores residuais, refletindo imprecisões no cálculo dos resíduos na linha 3(d) do algoritmo (4.11).

O algoritmo Bi-CGStab:

O algoritmo Bi-CGStab é uma variação do CGS desenvolvido para contornar as dificuldades citadas anteriormente. Ao invés de procurar um vetor residual como em (4.6.16), o Bi-CGStab produz iterados cujo vetores residuais são da forma

$$r_j' = \psi_j(A)\phi_j(A)r_0, \tag{4.6.28}$$

sendo $\phi_j(t)$ o polinômio associado ao resíduo no algoritmo Bi-CG e $\psi_j(t)$ um novo polinômio, definido recursivamente com o objetivo de "suavizar"o comportamento da convergência. A fórmula de recorrência é dada por:

$$\psi_{j+1}(t) = (1 - \omega_j t)\psi_j(t). \tag{4.6.29}$$

O escalar ω_j é determinado a posteriori. A maneira de deduzir as relações de recorrência é similar ao do algoritmo CGS. Será iniciado com o polinômio residual $\psi_{j+1}\phi_{j+1}$, obtendo

$$\begin{aligned} \psi_{j+1}\phi_{j+1} &= (1 - \omega_j t)\psi_j\phi_{j+1}, & (4.6.30)\\ &= (1 - \omega_j t)(\psi_j\phi_j - \alpha_j t\psi_j\pi_j). & (4.6.31) \end{aligned}$$

Para o termo $\psi_j\pi_j$ pode-se escrever

$$\begin{aligned} \psi_j\pi_j &= \psi_j(\phi_j + \beta_{j-1}\pi_{j-1}) & (4.6.32)\\ &= \psi_j\phi_j + \beta_{j-1}(1 - \omega_{j-1}t)\psi_{j-1}\pi_{j-1}. & (4.6.33) \end{aligned}$$

Define-se

$$\begin{aligned} r_j &= \phi_j(A)\psi_(A)r_0,\\ p_j &= \psi_j(A)\pi_j(A)r_0. \end{aligned}$$

Pelas relações acima, supondo que os escalares α_j e β_j estão disponíveis, esses vetores podem ser atualizados pelas seguintes fórmulas de recorrência

$$\begin{aligned} r_{j+1} &= (I - \omega_j A)(r_j - \alpha_j Ap_j)\\ p_{j+1} &= r_{j+1} + \beta_j(I - \omega_j A)p_j. \end{aligned} \tag{4.6.34}$$

A seguir, será deduzido o cálculo dos escalares necessários. Pelo algoritmo Bi-CG, tem-se que $\beta_j = \rho_{j+1}/\rho_j$ onde

$$\rho_j =< \phi_j(A)r_0, \phi_j(A^T)r_0^* >=< \phi_j^2(A)r_0, r_0^* > .$$

Note que o vetor $\phi_j^2(A)r_0$ não está disponível. Para contornar essa dificuldade, define-se

$$\begin{aligned} \widetilde{\rho}_j &= < \phi_j(A)r_0, \psi_j(A^T)r_0 > \\ &= < \psi_j(A)\phi_j(A)r_0, r_0^* > \\ &= < r_j, r_0^* > . \end{aligned}$$

Para relacionar os escalares ρ_j e $\widetilde{\rho}_j$, expandir-se-á o polinômio $\psi_j(A)$, obtendo-se

$$\widetilde{\rho}_j =< \phi_j(A)r_0, \eta_1^{(j)}(A^T)^j r_0^*, \eta_2^{(j)}(A^T)^{j-1}r_0^*, \dots > .$$

Como $\phi_j(A)r_0$ é ortogonal a todos vetores $(A^T)^k r_0^*$, com $k < j$, e se $\gamma_1^{(j)}$ é o primeiro coeficiente de ϕ_j, tem-se então

$$\widetilde{\rho}_j =< \phi_j(A)r_0, \frac{\eta_1^{(j)}}{\gamma_1^{(j)}}\phi_j(A^T)r_0 >= \frac{\eta_1^{(j)}}{\gamma_1^{(j)}}\rho_j.$$

Observando as relações de recorrência para ϕ_{j+1} e ψ_{j+1}, tem-se que

$$\eta_1^{(j+1)} = -\omega_j\eta_1^{(j)}, \quad \gamma_1^{j+1} = -\alpha_j\gamma_1^{(j)},$$

e como resultado

$$\frac{\widetilde{\rho}_{j+1}}{\widetilde{\rho}_j} = \frac{\omega_j}{\alpha_j}\frac{\rho_{j+1}}{\rho_j},$$

dando a seguinte relação para β_j :

$$\beta_j = (\frac{\widetilde{\rho}_{j+1}}{\widetilde{\rho}_j})(\frac{\alpha_j}{\omega_j}).$$

De uma maneira análoga, pode-se deduzir uma fórmula para α_j. Pelo algoritmo

$$\alpha_j = \frac{< \phi_j(A)r_0, \phi_j(A^T)r_0^* >}{< A\pi_j(A)r_0, \pi_j(A^T)r_0^* >}.$$

Como no caso anterior, os polinômios no lado direito dos produtos internos, tanto no numerador como no denominador, podem ser trocados por seus termos de maior grau. Portanto, neste caso, os coeficientes destes termos para $\phi_j(A^T)r_0^*$ e $\pi_j(A^T)r_0*$ são idênticos e, portanto,

$$\begin{aligned}\alpha_j &= \frac{<\phi_j(A)r_0, \phi_j(A^T)r_0^*>}{<A\pi_j(A)r_0, \pi_j(A^T)r_0^*>} \\ &= \frac{<\phi_j(A)r_0, \psi_j(A^T)r_0^*>}{<A\pi_j(A)r_0, \psi_j(A^T)r_0^*>} \\ &= \frac{<\psi_j(A)\phi_j(A)r_0, r_0^*>}{<A\psi_j(A)\pi_j(A)r_0, r_0^*>}.\end{aligned}$$

Como $p_j = \psi_j(A)\pi_j(A)r_0$, tem-se que

$$\alpha_j = \frac{\widetilde{\rho}_j}{<Ap_j, r_0^*>}. \tag{4.6.35}$$

É necessário, agora, definir o parâmetro ω_j. A escolha mais natural é escolhê-lo para minimizar $\|(I - \omega A)\psi_j(A)\phi_{j+1}(A)r_0\|_2$ sobre ω. Pela equação (4.6.34) tem-se que

$$r_{j+1} = (I - \omega_j A)s_j$$

onde

$$s_j \equiv r_j - \alpha_j A p_j.$$

Então, o valor ótimo de ω é dado por

$$\omega_j = \frac{<As_j, s_j>}{<As_j, As_j>}.$$

Assim, a equação (4.6.34) pode ser escrita como

$$r_{j+1} = s_j - \omega_j A s_j = r_j - \alpha_j A p_j - \omega_j A s_j,$$

e a atualização para a solução aproximada é dada por

$$x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j.$$

Com essas relações, pode-se escrever o seguinte algoritmo.

Algoritmo 4.12. *Algoritmo Bi-CGStab*

1. *Calcule* $r_0 = b - Ax_0$, r_0^* *arbitrário*
2. *Seja* $p_0 = r_0$
3. *Para* $j = 0, 1, \ldots,$ *até convergência faça*
   (a) $\alpha_j = < r_j, r_0^* > / < Ap_j, r_0^* >$
   (b) $s_j = r_j - \alpha_j Ap_j$
   (c) $\omega_j = < As_j, s_j > / < As_j, As_j >$
   (d) $x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j$
   (e) $r_{j+1} = s_j - \omega_j As_j$
   (f) $\beta_j = \frac{<r_{j+1}, r_0^*>}{<r_j, r_0^*>} \frac{\alpha_j}{\omega_j}$
   (g) $p_{j+1} = r_{j+1} + \beta_j < p_j - \omega_j Ap_j >$

Conclusão

Neste trabalho, duas formulações importantes das equações da rede elétrica foram resolvidas usando várias técnicas numéricas. No problema de fluxo de cargas, métodos de Newton Inexatos com diferentes algoritmos iterativos lineares foram empregados. A segunda formulação, abordada mais recentemente na literatura, empregou um método para o problema de quadrados mínimos não lineares com restrições.

Verificou-se, para os sistemas não lineares esparsos que representam o problema de fluxo de carga, que os precondicionadores tem um papel importante para a convergência da sequência de sistemas lineares subjacentes, mostrando um bom resultado ao usar precondicionadores baseados em fatoração ILU(0) e ILUT, este último responsável pela convergência do SSB340, onde a matriz jacobiana não apresenta uma estrutura de esparsidade favorável. Pode-se constatar pelos testes, que os métodos iterativos que tiveram um melhor desempenho foram os BiCG-Stab e o CGS, que são baseados no processo de biortogonalização de Lanczos.

À medida que a dimensão aumenta, melhor investimento deve ser feito no precondicionador. Para problemas de grande porte, como é o caso do SSB1916 com 1916 barras, considerado em BARBOZA [3], esquemas que combinam características de diferentes precondicionadores devem ser utilizados. Isto não é uma tarefa fácil, pois requer uma implementação sofisticada. Duas abordagens poderiam ser investigadas: o comportamento dos diferentes métodos iterativos em problemas de grande porte, que representam o caso real dos problemas em sistema de potência brasileiro e como os precondicionadores podem ser obtidos considerando os diferentes métodos iterativos e os diferentes sistemas. Em ambos os casos, um investimento importante em álgebra linear numérica, tanto do ponto de vista teórico quanto computacional, poderá fornecer respostas satisfatórias.

A não convergência de alguns casos para o problema de fluxo de carga sem o uso de precondicionamento já é um indício de que o problema sem solução apresenta mal condicionamento. De fato, isso foi identificado nos experimentos com-

putacionais. Não é uma tarefa trivial abordar um problema mal condicionado em computação de grande porte com métodos iterativos e esquemas de precondicionamento. Nos testes numéricos, mesmo usando uma técnica para diminuir o número de condição do sistema, o qual chamamos de sistema estendido, o processo de convergência à solução não é afetado favoravelmente. Este é mais um indício de como as equações da rede elétrica constituem um desafio para os analistas numéricos. No problema de fluxo de carga sem solução, lançamos mão do Lagrangeano aumentado como função de mérito, resultando em algumas melhorias, principalmente na taxa de convergência do método de Newton. Embora os resultados tenham sido melhores, as iterações de Gauss-Newton iniciais foram indispensáveis para a convergência.

Os métodos iterativos em subespaço de Krylov são a base dos modernos e eficientes programas de alto desempenho em análise numérica. Portanto, o estudo desses métodos desenvolvido neste trabalho deve servir de base para as futuras investigações, não só para o modelo em questão, mas para qualquer investimento em resolução de sistemas de grande porte de caráter geral.

Referências Bibliográficas

[1] MONTICELLI, A. J. *Fluxo de Carga em Redes de Energia Elétrica.* Edgard Blücher, São Paulo, 1983.

[2] GREENBAUM, A. *Iterative Methods for Solving Linear Systems.* SIAM Philadelphia, 1997.

[3] BARBOZA, L. V. *Análise e Desenvolvimento de Metodologias Corretivas para a Restauração da Solução das Equações da Rede Elétrica,* 2001. Tese de Doutorado. Dept. Eng. Elétrica, Universidade Federal de Santa Catarina - UFSC.

[4] BARBOZA, L. V.;FRANCISCO, J. B.; ZAMBALDI, M. C. Método de quadarados mínimos não lineares com restrições aplicado ao problema da rede elétrica. *Anais do* 53º *Seminário Brasileiro de Análise* 365-378 Maringá, maio 2001.

[5] BARBOZA, L. V.; SALGADO, R. Corrective Solutions of Steady State Power Systems via Newton Optimization Method. *Revista SBA Controle e & automação,* vol. 11, 182-186, 2000.

[6] BARRET, R.; BERRY M.; CHAN, T. F.; DEMMEL, J.; DONATO, J.; DONGARRA, J.; EIJKHOUT, V.; POZO, R.; ROMINE, C.; VAN DER VORST, H. *Templates.* SIAM Philadelphia, 1994.

[7] BROWN, P. N. A theoretical comparison of the Arnoldi and GMRES algorithms. *SIAM Journal on Scientific and Statistics Computing.* vol 12, 58-78, 1991.

[8] CHENEY, C. C. *Introduction to Approximation Theory.* McGraw Hill, NY, 1996.

[9] DEHNEL, M.; DOMMEL, H. W. A method for identifying weak nodes in non-convergent load flows. *IEEE Transactions on Power Systems.* vol. 4, no. 2, 801-807, 1989.

[10] DEMMEL, J. W. *Applied Numerical Linear Algebra.* SIAM Philadelphia, 1997.

[11] DENNIS, J. E. jr.; SCHNABEL, R. B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* SIAM Philadelphia, 1996.

[12] DONGARRA, J. J.; DUFF, I. S.; SORENSEN, D. C.; VAN DER VORST, H. A. *Numerical Linear Algebra for High-Performance Computers.* SIAM Philadelphia 1998.

[13] DUFF, I. S.; ERISMAN, A. M.; REID, J. K. . *Direct Methods for Sparse Matrices.* Oxford University Press Inc. . New York, 1992.

[14] GOLUB, G. A. and VAN LOAN, C. F. *Matrix Computations.* 3rd. Edition. The John Hopkins University Press Ltda. London, 1996.

[15] IWAMOTO, S.; TAMURA, Y. A load flow calculation method for ill-conditioned power systems. *IEEE Transactions on Power Apparatus and Systems.* vol. PAS-100, no. 4, 1736-1743, 1981.

[16] LUENBERGER, D. G. *Introduction to Linear and Nonlinear Programming.* Addison Wesley Publishing Company. Massachusetts, USA, 1973.

[17] NOCEDAL, J.; WRIGHT, J. *Numerical Optimization.* Springer Series in Operations Research, Springer Verlag New York Inc, 1999.

[18] OVERBYE, T. J. A power flow measure for unsolvable cases. *IEEE Transactions on Power Systems.* vol. 09, no. 3, 1359-1365, 1994.

[19] PÄRT-ENANDER, E.; SJÖBERG, A. *The Matlab Handbook 5.* Addison Wesley, Harlow UK, 1999.

[20] RIVLIN, T. J. *The Chebyshev Polynomials:* from Approximation Theory to Algebra and Number Theory. John Wiley and Sons, NY, 1990.

[21] SAAD, Y. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, Boston, USA, 1996.

[22] SAAD, Y. *Numerical Methods For Large Eigenvalue Problems.* John Wiley and Sons, NY, 1992.

[23] SONEVELD, P. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistics Computing.* vol. 10, 36-52, 1989.