**UNIVERSIDADE DE BRASÍLIA**

**FACULDADE DE TECNOLOGIA**
**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

# UMA ESTRUTURA DE SOLUCIONADOR ITERATIVO LINEAR COM APLICAÇÃO À SOLUÇÃO DE EQUAÇÕES DO PROBLEMA DE FLUXO DE CARGA

**ABIEZER AMARILIA FERNANDES**

**ORIENTADOR: JOÃO YOSHIYUKI ISHIHARA**

**TESE DE DOUTORADO EM ENGENHARIA ELÉTRICA**

**PPGEE.TD - 085/2014**

**BRASILIA: FEVEREIRO 2014**

Universidade de Brasília

Faculdade de Tecnologia
Departamento de Engenharia Elétrica

TESE DE DOUTORADO

# UMA ESTRUTURA DE SOLUCIONADOR ITERATIVO LINEAR COM APLICAÇÃO À SOLUÇÃO DE EQUAÇÕES DO PROBLEMA DE FLUXO DE CARGA

Abiezer Amarilia Fernandes

Tese submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Doutor em Engenharia Elétrica

Orientador: Prof. João Yoshiyuki Ishihara
Co-orientador: Prof. Francisco Damasceno Freitas
Banca Examinadora

Prof. João Yoshiyuki Ishihara, Dr., ENE/UnB – orientador

Prof. George Lauro Ribeiro de Brito, Dr., UFT – examinador externo

Prof. João Odilon Freitas e Silva, Dr., ONS - examinador externo

Prof. Luís Filomeno de Jesus Fernandes, Dr., FGA/ UnB - examinador interno

Prof. Henrique Cézar Ferreira, Dr., ENE/UnB – examinador interno

Brasília, Fevereiro/2014

## FICHA CATALOGRÁFICA


FERNANDES, ABIEZER AMARILIA

Uma estrutura de solucionador iterativo linear com aplicação à solução de equações do problema de fluxo de carga. [Distrito Federal] 2014.

xvii, 112p., 297 mm (ENE/FT/UnB, Doutor, Engenharia Elétrica, 2014).

Tese de Doutorado – Universidade de Brasília.

Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

| | |
|---|---|
| 1.Pré-condicionador | 2.Fatoração |
| 3.Solucionador | 4.Fluxo de carga |
| I.ENE/FT/UnB | II.Título (série) |


**REFERÊNCIA BIBLIOGRÁFICA**

FERNANDES, A. A. (2014). Uma estrutura de solucionador iterativo linear com aplicação à solução de equações do problema de fluxo de carga. Tese de Doutorado em Engenharia Elétrica, Publicação PPGEE.TD – 085/2014, Departamento de Engenharia Elétrica, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 112p.

**CESSÃO DE DIREITOS**

NOME DO AUTOR: Abiezer Amarilia Fernandes.

TÍTULO DA TESE DE DOUTORADO: Uma estrutura de solucionador iterativo linear com aplicação à solução de equações do problema de fluxo de carga.

GRAU / ANO: Doutor / 2014



_____


Abiezer Amarilia Fernandes

SQN 310, Bl. L aptº 603, Asa Norte

70.752-120 - Brasília – DF - Brasil

*À minha esposa Carmen*

# Agradecimentos

Nesta oportunidade expresso meus agradecimentos ao corpo docente da UnB que contribuiu para a formação do conhecimento adquirido ao longo da duração deste doutorado.

Expresso meus sinceros agradecimentos aos meus orientadores Prof. Dr. João Yoshiyuki Ishihara e Prof. Dr. Francisco Damasceno Freitas, pela confiança, dedicação, competência e trabalho conjunto. Sempre em busca do melhor trabalho possível.

Meu reconhecimento e agradecimento ao Prof. Dr.Marco Terada pela confiança e apoio nessa jornada.

Aos colegas e gestores da Eletronorte que acompanharam esse período de estudos em especial a Walter Muller e Leonardo Carvalho que conciliaram e confiaram na seriedade da proposta de evolução acadêmica.

Aos docentes do UniCEUB e em especial ao prof. José Pereira pelo incentivo.

Finalmente, os meus agradecimentos aos professores examinadores pelo aceite ao convite de participar da banca.

# Resumo

Nesta tese, motivado pelo problema de fluxo de carga em sistemas elétricos de potência, foi realizado um estudo sistemático de técnicas computacionais para solução de sistemas de equações lineares de grande dimensões, não-simétricos e esparsos. Foi proposta uma técnica geral para construção de solucionadores de sistemas lineares que combina métodos numéricos usuais com uma técnica de desacoplamento. Além de uma nova estrutura geral para solucionador de sistemas lineares, uma das contribuições deste trabalho foi um novo algoritmo para a identificação de subsistemas fracamente acoplados em sistemas lineares genéricos quaisquer. A efetividade da técnica proposta foi verificada por meio de simulações numéricas na resolução do problema de fluxo de carga para o sistema elétrico MATPOWER 3375 barras.

# Abstract

In this thesis, motivated by the power flow problem in power systems, it was performed a systematic study on computational techniques for solution of large, non-symmetric and sparse systems of linear equations. A general technique for setting solvers for linear systems was proposed by properly combining usual numerical methods for these systems and a decoupling technique. Besides a new general framework for linear solver, as one contribution of this work, it was proposed a new algorithm for identification of weakly coupled subsystems in a given generic linear system. The effectiveness of the proposed technique was verified by numerical simulations for resolution of the power flow problem for the MATPOWER, a power system with 3375 buses.

# Índice

## Lista de Figuras

# Lista de Tabelas

# Lista de Símbolos e Siglas

| | |
|---|---|
| $\boldsymbol{A}$ | matriz de coeficientes |
| $\boldsymbol{A}^{-1}$ | inversa da matriz de coeficientes |
| $\boldsymbol{A}^{T}$ | transposta da matriz de coeficientes |
| $\boldsymbol{A}_1, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ | submatrizes de $\boldsymbol{A}$ |
| $AMD$ | *Approximate Minimum Degree* (Grau de Aproximação Mínimo) |
| $AT$ | Alta Tensão |
| BiCG | *BiConjugate Gradient* (Gradiente Biconjugado) |
| BiCGStab | *BiConjugate Gradient Stabilized* (Gradiente Biconjugado Estabilizado) |
| $CA$ | corrente alternada |
| $CC$ | corrente contínua |
| CG | *Conjugate Gradient* (Gradiente Conjugado) |
| CGS | *Conjugate Gradient Squared* (Gradiente Conjugado Quadrado) |
| $cputime$ | tempo de *CPU* |
| $\Delta \boldsymbol{P}_i$ | resíduo de potência ativa na barra *i* |
| $\Delta \boldsymbol{Q}_i$ | resíduo de potência reativa na barra *i* |
| $\mathcal{D}_C(\cdot)$ | dominância de elementos de coluna |
| $\mathcal{D}_R(\cdot)$ | dominância de elementos de linha |
| $\widetilde{\mathcal{D}}_C(\cdot)$ | dominância relativa de elementos de coluna |
| $\widetilde{\mathcal{D}}_R(\cdot)$ | dominância relativa de elementos de linha |
| $EAT$ | Extra Alta Tensão |
| $etime$ | *elapsed time* (tempo decorrido) |
| GMRES | *Generalized Minimal Residual* (Resíduo Mínimo Generalizado) |
| $G(\cdot)$ | índice do ganho de esparsidade |
| $\boldsymbol{I}$ | matriz identidade |

| | |
|---|---|
| $ILU$ | fatores incompletos: inferior $(\boldsymbol{L})$ e superior $(\boldsymbol{U})$ de uma matriz ou submatriz |
| $\boldsymbol{J}$, $\boldsymbol{J}_0$ | matriz jacobiana, matriz jacobiana da iteração inicial |
| $K$ | subespaço de Krylov |
| $\boldsymbol{L}(\cdot)$ | fator inferior de uma matriz ou submatriz |
| $LDU$ | fatores: inferior $(\boldsymbol{L})$, diagonal $(\boldsymbol{D})$ e superior $(\boldsymbol{U})$ de uma matriz |
| $LU$ | fatores: inferior $(\boldsymbol{L})$ e superior $(\boldsymbol{U})$ de uma matriz ou submatriz |
| $MT$ | Média Tensão |
| $m$ | parâmetro indicador de nível de preenchimento |
| $n$ | dimensão de uma matriz |
| $\kappa(\boldsymbol{T})$ | número de condição da matriz $\boldsymbol{T}$ |
| $n_{nz}(\cdot)$ | número de elementos não-zeros de uma matriz |
| $norm(\cdot)$ | função Matlab - norma-2 de um vetor |
| $norm(.,1)$ | função Matlab - norma-1 de um vetor |
| $N$ | número de configurações das submatrizes de $\boldsymbol{A}$ |
| $NR$ | método Newton-Raphson |
| $PC$ | pré-condicionador |
| $PDE$ | *Parcial Differential Equation* (Equação Diferencial Parcial) |
| $PFC$ | Problema de Fluxo de Carga |
| $\mathcal{P}_{RC}$ | Produto do número de linha pelo número de coluna |
| $\boldsymbol{r}$ | resíduo |
| $RCM$ | *Reverse* Cuthill-McKee (Cuthill-McKee reverso) |
| $\mathbb{R}$ | conjunto dos números reais |
| $\mathcal{S}(\cdot)$ | grau de esparsidade |
| $\boldsymbol{S}$ | submatriz genérica de $\boldsymbol{A}$ |
| $tol$ | tolerância ao erro |
| $\boldsymbol{T}$ | matriz truncada de $\boldsymbol{A}$ |

| | |
|---|---|
| $\boldsymbol{U}(\cdot)$ | fator superior de uma matriz ou submatriz |
| $\boldsymbol{x}_0$ | valor inicial estimado |
| | |
| $\alpha$ | fator diferença soma de linhas |
| $\beta$ | fator diferença soma de colunas |
| $\varepsilon$ | tolerância |
| $\epsilon$ | tamanho de elemento da matriz |
| $\rho$ | parâmetro limitador de número máximo de elementos |
| $\tau$ | *drop tolerance* (tolerância de descarte) |
| | |
| + | promove sentido associativo à frase quando aplicado |
| $\lvert\cdot\rvert$ | valor absoluto ou módulo de vetor |
| $\lVert\cdot\rVert$ | símbolo associado à norma |
| $\ll$ | símbolo de muito menor que |

# Capítulo 1

# Introdução

## 1.1 INTRODUÇÃO GERAL

Em sistemas elétricos de potência, aspectos econômicos e de segurança, restrições físicas, requisitos de controle mais rígidos, entre outros fatores, ensejam níveis de exigência cada vez mais elevados [1]. Dentre os vários aspectos relacionados aos sistemas elétricos afetados por estas exigências, destaca-se o problema de fluxo de carga (*PFC*) ou de potência. Trata-se do tipo de estudo mais executado, porque é a partir dele que são determinados os estados do sistema, caracterizados pela magnitude e ângulo de fase de tensões. Portanto, constitui-se em importante ferramenta para análise de sistemas elétricos de potência quanto aos aspectos de planejamento, operação e controle [2]. Em geral, o *PFC* é resolvido pelo tradicional método de Newton-Raphson [3]. A cada iteração, este método requer a solução de um sistema linear, denominado nesta tese de *subproblema linear do PFC,* e representado genericamente como

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1.1}$$

em que $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ é uma matriz de coeficientes das variáveis, enquanto $\boldsymbol{x}$, $\boldsymbol{b} \in \mathbb{R}^{n}$ representam os vetores incógnita e independente, respectivamente. Vários métodos podem ser utizados para a resolução do subproblema linear associado ao *PFC* (ver por exemplo,[1,4-7]).

Durante a condição usual do sistema elétrico, a tendência é ocorrer convergência no método de Newton-Raphson para um ponto de operação (ponto de equilíbrio) que necessita ser calculado. Porém, há situações, como durante contingências ou mesmo condições severas de operação, em que a matriz $\boldsymbol{A}$ torna-se mal condicionada. Então, nestas condições severas, o processo iterativo não-linear poderá divergir. Em vista dessa

característica numérica indesejada, é importante ter um método confiável e eficaz para resolver o sistema linear de equações que é gerado a cada iteração.

Um algoritmo que fornece uma solução para o problema dado pela expressão (1.1), denomina-se solucionador[1]. Um dos aspectos que deve ser considerado para um solucionador dedicado para o *PFC* é a topologia da rede elétrica, uma vez que esta influencia na estrutura e no conteúdo das matrizes de coeficientes manipuladas durante a resolução do subproblema linear. Na maioria dos casos, estas matrizes resultam assimétricas em valores, embora sejam simétricas em estrutura. Além disso, em geral, também apresentam autovalores no semi-plano direito e esquerdo do plano complexo.

Muitos solucionadores da literatura foram desenvolvidos para casos particulares onde a matriz $\boldsymbol{A}$ satisfaz propriedades como ser de pequena dimensão, apresentar simetria, ser definitiva positiva ou ter autovalores apenas no semipleno esquerdo. Sistemas (1.1) que apresentam não-simetria, grande dimensão e indefinição na localização dos autovalores são um caso particularmente difícil para o processo de convergência dos solucionadores. Para estes sistemas, são utilizadas técnicas como pré-condicionamento [8-13] e reordenamento (reordenamento *reverse* Cuthill-McKee [4, 3, 5] e *Approximate Minimum Degree* [16-20]) na tentativa de aproxima-los a sistemas simétricos e com autovalores em lugares desejados e assim, melhorar as características de convergência. No entanto, cada problema particular demanda um estudo prospectivo *ad hoc* de qual pré-condicionador e reordenamento é o mais adequado.

Concluindo, os aspectos computacionais em sistemas elétricos de potência geralmente incluem a solução de sistemas não-lineares e lineares de grandes dimensões. Em geral, as matrizes no subproblema linear do *PFC* apresentam não-simetria, grande dimensão, indefinição na localização dos autovalores e estrutura que depende da topologia da rede. Desta forma, solucionadores lineares dedicados para o *PFC* são de grande interesse e importância prática. A necessidade de resolução de sistemas lineares

[1] O termo *solucionador* assumido nesta tese está associado ao termo "*solver*" em inglês e corresponde a um conjunto de métodos e estratégias matemáticas ordenadas de forma a solucionar um determinado problema [6, 22].

de grandes dimensões e não simétricas são também encontradas em diversas outras áreas como análise de transitórios eletromagnéticos, controle e estabilidade de sistema de potência, entre outras [6, 7, 8, 9, 10]. Assim, o problema principal considerado nesta tese é o desenvolvimento de uma estrutura geral de solucionador que possa ser particularizada conforme o tipo específico de estudo.

## 1.2 ESTRUTURA DA TESE

A presente tese está organizada da seguinte forma:

No Capítulo 2 apresenta-se um panorama sucinto dos métodos numéricos utilizados em processos de solução de sistemas lineares.

No Capítulo 3, apresenta-se uma proposta de pré-condicionamento de matrizes que incorpora desacoplamento de subsistemas.

No Capítulo 4, fundamenta-se o equacionamento do *PFC* e discorre-se sobre a geração de matrizes que serão alvo de avaliação via métodos de solução de sistemas lineares.

No Capítulo 5, são apresentados em detalhes quatro sistemas-teste: um modelo de sistema brasileiro, IEEE 118 barras, IEEE 300 barras, e um modelo de sistema integrante do aplicativo MATPOWER [11, 12].

No Capítulo 6 são apresentadas as simulações e resultados mais gerais sobre o desempenho na resolução de sistemas lineares iterativos, enquanto o Capítulo 7 é dedicado à apresentação dos experimentos e resultados associados aos aspectos teóricos discutidos no Capítulo 4.

Finalmente, as conclusões e recomendações para futuros trabalhos estão presentes no Capítulo 8.

Nos Apêndices 1 e 2 são encontrados os programas utilizados nas simulações nos ambientes Matlab e MATPOWER, respectivamente.

# Capítulo 2

# Métodos Iterativos

## 2.1 INTRODUÇÃO

Neste Capítulo, apresenta-se um panorama sucinto sobre o uso de métodos iterativos em processos de solução de sistemas lineares. Adicionalmente, é apresentado um breve histórico de experiências de aplicações dos métodos iterativos em sistemas elétricos de potência nos cálculos de fluxo de carga. Os fundamentos dos métodos iterativos com destaque ao parâmetro relativo ao resíduo e sua influência no número de condição da matriz de coeficientes são objeto de atenção especial. Em virtude do interesse dos estudos em métodos baseados em subespaço de Krylov, priorizam-se também os conceitos desta técnica. Por fim, como subsídio de escolha dos principais métodos visando aplicação na solução de problema em sistemas elétricos de potência, as principais características, vantagens e desvantagens são relatadas com base na literatura.

## 2.2 VISÃO GERAL

Sabe-se que o processo de solução de um sistema linear de grandes dimensões envolve um esforço computacional bastante elevado. Uma abordagem via método direto (eliminação gaussiana) pode se tornar proibitiva. Este fato já fora reconhecido por Gauss, em 1823 [13]. Nesse tempo, Gauss propôs um método iterativo para resolver um sistema linear com quatro equações e quatro incógnitas. Trata-se do primeiro método iterativo que se tem conhecimento. O mesmo ficou conhecido como método de Gauss-Jacobi, devido ao uso pelo matemático-astrônomo alemão Jacobi no cálculo de perturbação de órbitas de planetas do sistema solar. A aproximação proporcionada por esse método clássico evoluiu dando origem ao conhecido método Gauss-Seidel.

Avalia-se que a complexidade computacional para o processo de eliminação gaussiana necessária nos métodos de solução é da ordem de $n^3$, enquanto para calcular uma solução aproximada via Gauss-Seidel é da ordem de $n^2$ [13]. Destes dados, já é possível concluir sobre as vantagens dos métodos iterativos sobre os métodos diretos

notadamente para grandes dimensões. Porém, a utilização tanto do método de Gauss-Jacobi, quanto de Gauss-Seidel ocorre para as aplicações muito específicas, em que a matriz de coeficientes precisa ter dominância diagonal[2].

A crescente evolução em busca de melhorias nos métodos ao longo dos anos pode ser constatada pelo sumário histórico descrito a seguir.

Em 1971, Young [14], visando superar a lentidão de convergência causada para a condição de raio espectral tendendo a unidade, produziu um dos mais expressivos tratamentos a respeito do método de sobre relaxação sucessiva (*Successive Over-Relaxation* (SOR)), que é uma técnica oriunda de modificação no algoritmo de Gauss-Seidel [15]. Em 1955, Sheldon [16] discutiu a possibilidade de "simetrizar" um método iterativo com o intuito de simplificar o processo de aceleração de convergência. A partir da técnica SOR, na condição de que todos os autovalores relevantes sejam reais, foi proposto o método sobre relaxação sucessiva simétrica (*Symmetric Successive Over-Relaxation* (SSOR)). As experiências evidenciam que dificuldades apontadas nesses métodos estão associadas à escolha dos parâmetros de relaxação, difíceis de serem escolhidos adequadamente [15].

O método iterativo denominado gradiente conjugado (*Conjugate Gradient* (CG)) não apresenta as dificuldades antes citadas, pois pertence à classe de métodos que são relacionados aos algoritmos de direção conjugada. Nos algoritmos de direção conjugada, a busca de direções que apontam para a provável solução depende do conjugado da matriz ***A*.** As referências clássicas e as análises matemáticas sobre esse método são encontradas em[30-32]. O método CG, por sua vez restringe-se à aplicação de sistemas do tipo simétrico e positivo-definido [15].

Outros métodos são os baseados em subespaço de Krylov, como o método do resíduo mínimo (*Minimum Residual* (MINRES)) [17] e o método *LQ* simétrico (*Symmetric LQ* (SYMMLQ)) [17]. Estes métodos também são restritos a sistemas

[2] O termo *dominancia diagonal* relacionado a uma matriz pode ser aplicado, se para toda linha da matriz, a magnitude da entrada diagonal em uma linha é maior ou igual à soma das magnitudes de todas as outras entradas não diagonais naquela linha [15].

simétricos, mas podendo ser indefinidos [15]. Métodos para sistemas assimétricos tradicionais, incluem o GMRES, o BiCG, o BiCGStab, o CGS [15, 18].

## 2.3 APLICAÇÕES EM SISTEMAS DE POTÊNCIA

Visando elencar a aplicação de alguns métodos lineares iterativos utilizados na solução do *PFC*, apresentam-se alguns a seguir na ordem cronológica das pesquisas sobre o tema.

Resultados apresentados em trabalho publicado em 1993 por Galiana *et al.* [19] ilustram a utilização do método CG pré-condicionado. Em 1996, Semlyen [8] apresentou resultados da proposta metodológica para fluxo de carga, baseado no subespaço de Krylov. Para este fim, foram formuladas duas alternativas na aproximação da matriz jacobiana do processo não-linear: uma através de matriz constante e outra, baseada em processo Quase-Newton. Em Borges *et al.* [20], 1996, é utilizado o método BiCGStab pré-condicionado. A sua utilização é voltada para determinação da solução do problema de fluxo de carga, cujos cálculos são executados em ambiente computacional vetorial-paralelo.

Em 1997, novamente, Borges *et al.* [2] apresentaram e destacaram as vantagens do uso dos métodos iterativos em ambientes computacionais paralelos. Em Pai & Dag 1997 [6], foi proposta metodologia em que os métodos iterativos do subespaço de Krylov são tratados com alternativas de solução dos problemas de fluxo de carga, estimação de estado e segurança de sistemas elétricos de potência. Como conclusão importante, os autores recomendam o uso de pré-condicionadores baseados em fatoração *ILU*.

Em 1998, como uma alternativa promissora para solucionar equações do fluxo de carga, Flueck e Chiang [21] apresentaram o método Newton-GMRES resultante da combinação do método inexato de Newton e o GMRES pré-condicionado com fatoração incompleta. As conclusões do trabalho indicam que quando combinado com pré-condicionadores de qualidade, o método Newton-GMRES pode proporcionar redução da

ordem de 50% nos cálculos comparados ao método Newton-LU. Nesse trabalho, os estudos consideraram testes em um sistema de 3493 barras e outro de 8027 barras. Para o adequado desempenho da técnica, estratégias de reordenamento foram avaliadas.

Em 2002, León e Semlyen [22] realizaram comparação entre métodos iterativos GMRES, BiCGStab, CGS, BiCG, QMR e o método direto de fatoração *LU*. É apresentada uma técnica de atualização parcial da matriz Jacobiana em cada iteração do método Newton, constatando em dois sistemas de 3186 e 6372 barras, que o método iterativo BiCGStab é computacionalmente mais eficiente que o método direto.

Em 2003, Dag e Semlyen [10] apresentaram uma solução para o problema de fluxo de carga baseada no método desacoplado rápido, desta vez, substituindo o método direto pelo método iterativo Gradiente Conjugado com pré-condicionador para aproximação da jacobiana inversa. Nesse trabalho, são usadas técnicas de escalonamento, obtendo-se resultados satisfatórios para a aplicação.

Em 2006, Cheng e Shen [7] apresentam um pré-condicionador adaptativo baseado nos métodos de aproximação da inversa da matriz jacobiana. É usado o método de Newton-GMRES(m)[3], onde o pré-condicionador é construído considerando a aproximação da matriz Jacobiana (Jacobian-Free) do método Newton inexato. Sua atualização é verificada durante ambas as iterações do problema não-linear e GMRES por meio de um algoritmo atualizador denominado *rank*-1.

Em 2012, Idema *et al.* [1] utilizaram um pré-condicionamento com fatoração incompleta, com desacoplamento de matriz aplicado ao método GMRES. Alguns esquemas de reordenamento foram estudados, indicando a eficiência do método *AMD*. O trabalho destacou a aplicação da técnica com sucesso em sistema de cerca de um milhão de variáveis, na qual foi constatado que a alternativa apresentada pode ser mais rápida a até 120 vezes que o método direto nos sistemas maiores testados.

[3] A sigla GMRES(m) refere-se ao método GMRES onde são realizadas $q$ reinicializações, com um *loop* interno tendo $m$ iterações.

## 2.4 CONCEITOS GERAIS

Existem duas categorias de métodos iterativos: estacionário e não-estacionário. São classificados como estacionários os métodos que executam, a cada iteração, operações de cálculos similares com vetores. Exemplos desta natureza são os métodos de: Jacobi, Gauss-Seidel, SOR. Os métodos não-estacionários buscam um valor da variável utilizando direções de busca baseadas na teoria de subespaços de Krylov [23]. Nesta busca, os coeficientes são atualizados a cada iteração e o resultado depende de uma sequência de vetores que são formados pelo produto de uma potência da matriz $\boldsymbol{A}$ pelo resíduo inicial. Estes vetores formam uma base ortogonal para um subespaço, razão da denominação subespaço de Krylov.

Assim, uma solução $\boldsymbol{x}$ para a equação (1.1) pode ser calculada com determinada precisão numérica dada. A partir de uma solução inicial $\boldsymbol{x}^{(0)}$, é gerada uma sequencia de soluções $\{\boldsymbol{x}^{(k)}\}$, a cada iteração $k$, tal que após um determinado número finito de iterações e atendida a tolerância definida resulte na solução próxima de $\boldsymbol{x}$:

$$\boldsymbol{x} \approx \boldsymbol{A}^{-1}\boldsymbol{b} \tag{2.1}$$

Embora os métodos associados ao subespaço de Krylov, teoricamente, convirjam à solução com o número de iterações definido pela dimensão da matriz, na prática, os erros de arredondamento podem impedir essa convergência, principalmente se a matriz for mal condicionada.

Logo, um critério racional para decidir sobre a parada no processo de cálculo iterativo de um método deve levar em conta parâmetros para avaliação da solução, como os baseados no resíduo $\boldsymbol{r}$ da solução (erros associados à solução) e acompanhamento do processo de convergência [24].

Com base na equação (1.1), define-se o resíduo como sendo:

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \tag{2.2}$$

Consequentemente, o resíduo será nulo, quando $\boldsymbol{x}$ é a solução de $\boldsymbol{Ax} = \boldsymbol{b}$. Para um método iterativo o resíduo em uma iteração $k$ é dado por:

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)} \quad k = 0,1,2, \ldots \tag{2.3}$$

Se um método iterativo converge, então $\boldsymbol{r}^{(k)} \to 0 \ \text{com} \ k \to \infty$.

Assumindo que $\boldsymbol{x}^*$ seja a solução exata do sistema linear (1.1), então $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}^*$. Logo,

$$\boldsymbol{r}^{(k)} = \boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{A}\boldsymbol{x}^{(k)} = \boldsymbol{A}\left(\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\right) \ \rightarrow \ \boldsymbol{x}^* - \boldsymbol{x}^{(k)} = \boldsymbol{A}^{-1}\boldsymbol{r}^{(k)} \tag{2.4}$$

Logo, a dimensão do erro absoluto na *k*–ésima iteração será:

$$\left\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\right\| = \left\|\boldsymbol{A}^{-1}\boldsymbol{r}^{(k)}\right\| \tag{2.5}$$

Da definição de normas de matriz apresentada em [25], tem-se:

$$\left\|\boldsymbol{A}^{-1}\boldsymbol{r}^{(k)}\right\| \leq \|\boldsymbol{A}^{-1}\|\left\|\boldsymbol{r}^{(k)}\right\| \tag{2.6}$$

Consequentemente,

$$\left\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\right\| \leq \|\boldsymbol{A}^{-1}\|\left\|\boldsymbol{r}^{(k)}\right\| \tag{2.7}$$

Dividindo a equação (2.7) pelo escalar $\|\boldsymbol{x}^*\|$, tem-se:

$$\frac{\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}^*\|} \leq \|\boldsymbol{A}^{-1}\| \frac{\|\boldsymbol{r}^{(k)}\|}{\|\boldsymbol{x}^*\|} \tag{2.8}$$

Mas,

$$\|\boldsymbol{b}\| = \|\boldsymbol{A}\boldsymbol{x}^*\| \leq \|\boldsymbol{A}\|\|\boldsymbol{x}^*\| \;\rightarrow\; \frac{1}{\|\boldsymbol{x}^*\|} \leq \frac{\|\boldsymbol{A}\|}{\|\boldsymbol{b}\|} \tag{2.9}$$

Substituindo a equação (2.9) em (2.8), tem-se que:

$$\frac{\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}^*\|} \leq \|\boldsymbol{A}^{-1}\|\left\|\boldsymbol{r}^{(k)}\right\| \frac{\|\boldsymbol{A}\|}{\|\boldsymbol{b}\|} \tag{2.10}$$

Introduzindo a expressão da definição do número de condição de $\boldsymbol{A}$,

$$\kappa(\boldsymbol{A}) \coloneqq \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\| \tag{2.11}$$

então a equação (2.10) pode ser escrita como:

$$\frac{\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}^*\|} \leq \kappa(\boldsymbol{A}) \frac{\|\boldsymbol{r}^{(k)}\|}{\|\boldsymbol{b}\|} \tag{2.12}$$

Assim, o erro relativo na solução de $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ decorrente da parada na iteração $k$ pode ser estimado pelo escalar $\left\|\boldsymbol{r}^{(k)}\right\|/\|\boldsymbol{b}\|$. Se a matriz $\boldsymbol{A}$ for mal condicionada, $\kappa(\boldsymbol{A})$ é significativamente elevado. Logo, é possível afirmar que $\left\|\boldsymbol{r}^{(k)}\right\|/\|\boldsymbol{b}\|$ não é um indicador seguro de convergência. No entanto, em aplicações prática é sugerido como critério de parada que $\|\boldsymbol{r}\| \ll \varepsilon\|\boldsymbol{b}\|$, onde $\varepsilon$ é a tolerância admitida para convergência. Desta forma,

$$\frac{\|\boldsymbol{r}^{(k)}\|}{\|\boldsymbol{b}\|} \ll \varepsilon \tag{2.13}$$

sendo $\varepsilon$ um valor positivo pequeno, mas muito maior do que a precisão da máquina. Tipicamente o valor de $\varepsilon$ encontra-se na faixa de $1 \times 10^{-6} < \varepsilon < 5 \times 10^{-3}$, dependendo do problema a ser resolvido e do custo das iterações [24].

## 2.5 MÉTODOS NO SUBESPAÇO DE KRYLOV

A definição do método no subespaço de Krylov depende do tipo de matriz de coeficientes que cada método é capaz de solucionar [26]. Considerando o aspecto de desempenho do problema, elege-se o método cujas propriedades sejam mais vantajosas para solucioná-lo. Também, que tenha propriedades que permitam explorar de forma eficiente as características numéricas da matriz de coeficientes. Parâmetros avaliativos de qualidade do método passam pela medida de quão rápido $\boldsymbol{x}^{(k)}$ converge, isto é, segundo o número de iterações (robustez) e segundo a taxa de redução do erro com as iterações (tempo computacional).

Assim, Krylov em 1931 [27] propôs a técnica para construção de polinômios característicos de matrizes que tem por fundamento a construção da matriz $\boldsymbol{K}$, regular (não-singular) e da matriz $\boldsymbol{H}$ (matriz de Hessenberg), tal que o produto $\boldsymbol{K}^{-1}\boldsymbol{A}\boldsymbol{K} = \boldsymbol{H}$ se verifique [27]. Por tratar-se de relação de similaridade, $\boldsymbol{A}$ e $\boldsymbol{H}$ têm o mesmo polinômio característico. As colunas da matriz $\boldsymbol{K}$, em uma primeira fase da técnica, são construídas pela multiplicação de um vetor $\boldsymbol{r}$ por $\boldsymbol{A}$, onde, a *j*-ésima coluna de $\boldsymbol{K}$ (notação $\boldsymbol{K}(:, j)$) é dada por $\boldsymbol{A}^{j-1}\boldsymbol{r}$, sendo $\boldsymbol{K} = [\boldsymbol{r}\ \ \boldsymbol{A}\boldsymbol{r}\ \ \boldsymbol{A}^2\boldsymbol{r}\ \ \ldots\ \boldsymbol{A}^{k-1}\boldsymbol{r}]$ [27]. Assim, é possível de se definir o subespaço de Krylov por uma sequência de vetores cuja base é formada por [18]:

$$K_k\left(\boldsymbol{A}, \boldsymbol{r}^{(0)}\right) \coloneqq span\left(\boldsymbol{r}^{(0)}, \boldsymbol{A}\boldsymbol{r}^{(0)}, \boldsymbol{A}^2\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{A}^{k-1}\boldsymbol{r}^{(0)}\right) \tag{2.14}$$

$$\boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)} \tag{2.15}$$

onde $\boldsymbol{A}$ é a matriz de coeficientes, $\boldsymbol{r}^{(0)}$ o resíduo inicial e $\boldsymbol{x}^{(0)}$ a estimativa inicial.

O método do subespaço de Krylov para resolver a equação (1.1) é baseado em uma solução aproximada $\boldsymbol{x}^{(k)}$ que pertence a um subespaço $K_k$. Nestas condições, é estabelecida a condição de norma mínima de erro da solução exata da equação (1.1)

$$\left\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\right\|_2 = min\{\|\boldsymbol{x}^* - \boldsymbol{x}\|_2 : \boldsymbol{x} \in \boldsymbol{x}_0 + K_k\} \tag{2.16}$$

Métodos que fornecem melhores aproximações no subespaço de Krylov podem ser classificados de acordo com quatro diferentes condições de busca da solução: Ritz-Galerkin, Petrov-Galerkin e normas mínimas: do tipo residual e de erro [13]. A primeira condição, calcula $\boldsymbol{x}^{(k)}$ na condição em que o resíduo é ortogonal com relação ao mais recente subespaço construído, ou seja, $\left(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k)}\right) \perp K_k\left(\boldsymbol{A},\boldsymbol{r}^{(0)}\right)$; na segunda, é gerado um $\boldsymbol{x}^{(k)}$, cujo cálculo requer que o resíduo $\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k)}$ seja ortogonal a algum subespaço k-dimensional aceitável; a terceira, identifica-se $\boldsymbol{x}^{(k)}$ supondo que a norma euclidiana $\left\|\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k)}\right\|_2$ seja mínima sobre $K_k\left(\boldsymbol{A},\boldsymbol{r}^{(0)}\right)$; finalmente, a norma mínima do erro determina $\boldsymbol{x}^{(k)}$ assumindo que $\boldsymbol{A}^T K_k\left(\boldsymbol{A}^T,\boldsymbol{r}^{(0)}\right)$ de tal forma que a norma euclidiana $\left\|\boldsymbol{x}^*-\boldsymbol{x}^{(k)}\right\|_2$ seja mínima.

Dos diversos métodos baseados no subespaço de Krylov, neste trabalho, priorizou-se os estudos dos métodos GMRES, BiCG, BiCGStab e CGS. Estas escolhas são justificadas pelas características comuns de fornecerem melhores aproximações no subespaço de Krylov [18, 13], devido às classes de projeções (ortogonais ou bi-ortogonais) usadas normalmente para reduzir o resíduo. Portanto, apenas métodos com base em subespaço de Krylov serão investigados.

## 2.6 MÉTODOS ITERATIVOS EM ESTUDO

Com o objetivo de aplicar um método iterativo adequado que possa integrar a proposta de um novo solucionador iterativo para a aplicação neste trabalho, nas subseções seguintes, são apresentadas as principais características (vantagens e desvantagens) dos métodos em estudo, em geral aplicados a matrizes assimétricas e indefinidas. Também é apresentada, uma síntese evolutiva de busca de melhorias de desempenho dos métodos.

### 2.6.1 Métodos iterativos – características principais

Considera-se que o favorecimento de um ou outro método na solução do problema objeto de estudo, é função do tipo da matriz $\boldsymbol{A}$ e dos seus algoritmos de construção [26]. A Tabela 2.1 apresenta uma síntese das características dos algoritmos

associados à construção e à busca de soluções de cada método, todos para matriz de coeficientes assimétrica. A denominação híbrida para o algoritmo de condição de busca, leva em conta tratamentos matemáticos envolvendo algoritmos dos métodos de Ritz-Galerkin, Petrov-Galerkin e normas mínimas: residual e de erro.

**Tabela 2.1** – Características principais dos quatro métodos iterativos

| *Método* | *Algoritmo característico* | |
|---|---|---|
| | *Construção* | *Solução* |
| GMRES | Arnoldi | Norma mínima residual |
| BiCGStab | Bi-Lanczos modificado | Híbrida |
| BiCG | Bi-Lanczos | Petrov-Galerkin |
| CGS | Lanczos não simétrico | Híbrida |

Na busca pela aplicação do método mais adequado, o conhecimento da estrutura básica de seu algoritmo se faz necessário. Os detalhamentos a seguir cumprem esta finalidade.

O GMRES, método que por concepção tem por objetivo minimizar um resíduo, requer um elevado custo computacional para manter todos os resíduos ortogonais, ocasionando em aumento do espaço de memória requerido [28, 23]. Seu algoritmo de construção tem como base processos de Arnoldi [18]. Este método apresenta-se como alternativa para sistemas com matrizes de coeficientes indefinidas [23] (autovalores com parte real negativa e positiva). O método GMRES é empregado nas mais diversas aplicações físicas [29, 7, 30]. O Algoritmo 2.1 ilustra códigos de *scripts*, em que são destacadas as principais etapas para implementação do método.

**Algoritmo 2.1** - Método GMRES sem reinicialização [18]

**ENTRADA:** vetor $\boldsymbol{b}$, matriz $\boldsymbol{A}$ de coeficientes, estimativa inicial $\boldsymbol{x}_0$

**SAÍDA:** solução $\boldsymbol{x} = \boldsymbol{x}_m$

1. Calcular $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\beta := \|\boldsymbol{r}_0\|_2$ e $\boldsymbol{v}_1 := \boldsymbol{r}_0/\beta$
2. Repetir $j = 1,2 \ldots, m$ Faça:
3. Calcular $\boldsymbol{w}_j := \boldsymbol{A}\boldsymbol{v}_j$
4. Repetir $i = 1, \ldots, j$ Faça:
5. $\boldsymbol{h}_{ij} := (\boldsymbol{w}_j, \boldsymbol{v}_i)$
6. $\boldsymbol{w}_j := \boldsymbol{w}_j + \boldsymbol{h}_{ij}\boldsymbol{v}_i$
7. Fim-Repetir
8. $\boldsymbol{h}_{j+1,j} := \|\boldsymbol{w}_j\|_2$. Se $\boldsymbol{h}_{j+1,j} = \boldsymbol{0}$ Definir $m := j$ *e vá para 11*
9. $\boldsymbol{v}_{j+1} := \boldsymbol{w}_j/\boldsymbol{h}_{j+1,j}$
10. Fim-Repetir
11. Definir a matriz de Hessenberg $(m+1) \times m$, $\bar{\boldsymbol{H}}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$ e $\boldsymbol{V}_m$ formada com os vetores $\boldsymbol{v}_i$, $i = 1,2, \ldots, m$
12. Calcular $\boldsymbol{y}_m$ que minimiza $\|\beta e_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}\|_2$ e obtenha $\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m$

Na situação em que o Algoritmo 2.1 é repetido (reinicializado) $q$ vezes com uma nova estimativa $\boldsymbol{x}_0 = \boldsymbol{x}_m$ correspondente, portanto, na última solução, o método é dito GMRES com $q$ reinicializações, com um *loop* interno tendo $m$ iterações. Neste caso, utiliza-se a notação GMRES(m).

Em princípio, o algoritmo de Arnoldi utilizado na construção do subespaço de Krylov no método GMRES não apresenta interrupções nem problema de estabilidade numérica durante sua execução, o que destaca o método entre os mais robustos [23, 31, 32]. Por esta razão, este método é amplamente utilizado em diversas aplicações [33, 23, 34].

O método BiCG difere do GMRES no fato que não minimiza um resíduo. Sua carga computacional é alta, pois demanda duas vezes o produto matriz-vetor por passo iterativo podendo ocasionar em baixo desempenho [28]. Além disso, o BiCG precisa realizar, em cada iteração, uma multiplicação matriz-vetor com cada matriz $\boldsymbol{A}$ e $\boldsymbol{A}^T$, aumentando o custo computacional. No método BiCG em vez de cada ortogonalização ser realizada em sequência, ela é feita mutuamente ortogonal ou bi-ortogonal. As duas sequências dos passos 6 e 7 do Algoritmo 2.2 são relativas aos resíduos enquanto os passos 9 e 10 são de busca de direções.

**Algoritmo 2.2** – Método BiCG [18]

---

**ENTRADA:** vetor $\boldsymbol{b}$, matriz $\boldsymbol{A}$ de coeficientes, estimativa inicial de $\boldsymbol{x}_0$

**SAÍDA:** solução $\boldsymbol{x} = \boldsymbol{x}_{j+1}$

---

1. Calcular $\boldsymbol{r}_0 \coloneqq \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$. Escolher $\boldsymbol{r}_0^*$ tal que $(\boldsymbol{r}_0, \boldsymbol{r}_0^*) \neq 0$
2. Definir $\boldsymbol{p}_0 \coloneqq \boldsymbol{r}_0, \boldsymbol{p}_0^* \coloneqq \boldsymbol{r}_0^*$
3. Repetir $j = 0,1, \ldots$, até a convergência Faça:
4. $\quad \boldsymbol{\alpha}_j \coloneqq (\boldsymbol{r}_j, \boldsymbol{r}_j^*)/(\boldsymbol{A}\boldsymbol{p}_j, \boldsymbol{p}_j^*)$
5. $\quad \boldsymbol{x}_{j+1} \coloneqq \boldsymbol{x}_j + \boldsymbol{\alpha}_j \boldsymbol{p}_j$
6. $\quad \boldsymbol{r}_{j+1} \coloneqq \boldsymbol{r}_j + \boldsymbol{\alpha}_j \boldsymbol{A}\boldsymbol{p}_j$
7. $\quad \boldsymbol{r}_{j+1}^* \coloneqq \boldsymbol{r}_j^* + \boldsymbol{\alpha}_j \boldsymbol{A}^T \boldsymbol{p}_j^*$
8. $\quad \boldsymbol{\beta}_j \coloneqq (\boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1}^*)/(\boldsymbol{r}_j, \boldsymbol{r}_j^*)$
9. $\quad \boldsymbol{p}_{j+1} \coloneqq \boldsymbol{r}_{j+1} + \boldsymbol{\beta}_j \boldsymbol{p}_j$
10. $\quad \boldsymbol{p}_{j+1}^* \coloneqq \boldsymbol{r}_{j+1}^* + \boldsymbol{\beta}_j \boldsymbol{p}_j^*$
11. Fim-Repetir

---

Observa-se que o método BiCG requer a multiplicação da matriz $\boldsymbol{A}$ de coeficientes e de sua transposta a cada iteração, decorrendo em exigência de memória computacional. Propostas de melhorias na eficiência utilizando variantes do BiCG surgiram como alternativa. Os métodos CGS [28, 35] e por extensão o BiCGStab [28, 33] são dois exemplos.

O CGS é um método que requer apenas uma operação matriz-vetor por iteração. Trata-se de uma evolução em relação ao BiCG sendo que esta modificação tornou-o computacionalmente mais eficiente que o BiCG [35]. Uma das vantagens práticas é que o método não necessita de multiplicações com a transposta da matriz de coeficientes. As etapas básicas requeridas pelo método são apresentadas no Algoritmo 2.3.

**Algoritmo 2.3** – Método CGS [18]

---

**ENTRADA:** vetor $\boldsymbol{b}$, matriz $\boldsymbol{A}$ de coeficientes, estimativa inicial de $\boldsymbol{x}_0$

**SAÍDA:** solução $\boldsymbol{x} = \boldsymbol{x}_{j+1}$

---

1. Calcular $\boldsymbol{r}_0 \coloneqq \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$;
2. Definir, $\boldsymbol{p}_0 \coloneqq \boldsymbol{u}_0 \coloneqq \boldsymbol{r}_0$
3. Repetir $j = 0,1,2$ ..., até a convergência Faça:
4. $\boldsymbol{\alpha}_j \coloneqq (\boldsymbol{r}_j, \boldsymbol{r}_0)/(\boldsymbol{A}\boldsymbol{p}_j, \boldsymbol{r}_0)$
5. $\boldsymbol{q}_j \coloneqq \boldsymbol{u}_j + \boldsymbol{\alpha}_j \boldsymbol{A}\boldsymbol{p}_j$
6. $\boldsymbol{x}_{j+1} \coloneqq \boldsymbol{x}_j + \boldsymbol{\alpha}_j(\boldsymbol{u}_j + \boldsymbol{q}_j)$
7. $\boldsymbol{r}_{j+1} \coloneqq \boldsymbol{r}_j + \boldsymbol{\alpha}_j \boldsymbol{A}(\boldsymbol{u}_j + \boldsymbol{q}_j)$
8. $\boldsymbol{\beta}_j \coloneqq (\boldsymbol{r}_{j+1}, \boldsymbol{r}_0)/(\boldsymbol{r}_j, \boldsymbol{r}_0)$
9. $\boldsymbol{u}_{j+1} \coloneqq \boldsymbol{r}_{j+1} + \boldsymbol{\beta}_j \boldsymbol{q}_j$
10. $\boldsymbol{p}_{j+1} \coloneqq \boldsymbol{u}_{j+1} + \boldsymbol{\beta}_j(\boldsymbol{q}_j + \boldsymbol{\beta}_j \boldsymbol{p}_j)$
11. Fim-Repetir

---

O BiCGStab é um método variante do CGS, onde o algoritmo Bi-Lanczos é modificado com a intenção de controlar e minimizar o resíduo. Isto resulta em uma convergência mais suave e rápida se comparada à obtida no CGS, evitando-se irregularidades e oscilações. Pode ser considerado como uma evolução bem sucedida dentre os métodos baseados nos algoritmos de Lanczos. E superior, se comparado ao BiCG e CGS, dado ao bom desempenho em termos de robustez e eficiência computacional [33]. Em Saad [18], cita-se que são previstos erros de arredondamento

devido a procedimentos de correções de falhas de convergências irregulares [18]. O Algoritmo 2.4 elenca as principais etapas e procedimentos do método.

**Algoritmo 2.4** - Método BiCGStab [18]

**ENTRADA:** vetor $\boldsymbol{b}$, matriz $\boldsymbol{A}$ de coeficientes, valor estimado $\boldsymbol{x}_0$

**SAÍDA:** solução $\boldsymbol{x} = \boldsymbol{x}_{j+1}$

1. Calcular $\boldsymbol{r}_0 \coloneqq \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$.
2. Definir $\boldsymbol{p}_0 \coloneqq \boldsymbol{r}_0$
3. Repetir $j = 0,1, \ldots,$ até a convergência Faça*:*
4. $\quad \boldsymbol{\alpha}_j \coloneqq (\boldsymbol{r}_j, \boldsymbol{r}_0)/(\boldsymbol{A}\boldsymbol{p}_j, \boldsymbol{r}_0)$
5. $\quad \boldsymbol{s}_j \coloneqq \boldsymbol{r}_j + \boldsymbol{\alpha}_j \boldsymbol{A}\boldsymbol{p}_j$
6. $\quad \boldsymbol{w}_j \coloneqq (\boldsymbol{A}\boldsymbol{s}_j, \boldsymbol{s}_j)/(\boldsymbol{A}\boldsymbol{s}_j, \boldsymbol{A}\boldsymbol{s}_j)$
7. $\quad \boldsymbol{x}_{j+1} \coloneqq \boldsymbol{x}_j + \boldsymbol{\alpha}_j \boldsymbol{p}_j + \boldsymbol{w}_j \boldsymbol{s}_j$
8. $\quad \boldsymbol{r}_{j+1} \coloneqq \boldsymbol{s}_j - \boldsymbol{w}_j \boldsymbol{A}\boldsymbol{s}_j$
9. $\quad \boldsymbol{\beta}_j \coloneqq \frac{(\boldsymbol{r}_{j+1}, \boldsymbol{r}_0)}{(\boldsymbol{r}_j, \boldsymbol{r}_0)} \times \frac{\boldsymbol{\alpha}_j}{\boldsymbol{w}_j}$
10. $\quad \boldsymbol{p}_{j+1} \coloneqq \boldsymbol{r}_{j+1} + \boldsymbol{\beta}_j (\boldsymbol{p}_j - \boldsymbol{w}_j \boldsymbol{A}\boldsymbol{p}_j)$
11. Fim-Repetir

Após a apresentação das principais características dos métodos iterativos, investigações são aprofundadas na próxima subseção com objetivo de destacar as vantagens e desvantagens destes métodos com base nos resultados em pesquisas realizadas.

### 2.6.2 Métodos iterativos – vantagens e desvantagens

Nesta subseção, descreve-se uma síntese das evoluções construtivas dos algoritmos dos métodos iterativos, destacando-se vantagens e fragilidades na busca assertiva do método que poderá contribuir competitivamente na solução para o problema de fluxo de carga. São destacadas as principais características observadas nos algoritmos de construção dos métodos em estudos.

Em Lanczos 1952 [36], registra-se a experiência de solucionar sistemas lineares através do método BiCG. Mais tarde, Fletcher em 1976 [37], envida esforços de melhorias. Este método, embora destinado a solucionar sistemas assimétricos, apresenta desvantagens em termos de robustez associadas ao seu baixo desempenho computacional [38]. Utiliza a forma de bi-ortogonalização de Lanczos ou Bi-Lanczos, para construir as bases ortogonais do subespaço de Krylov. No entanto, este algoritmo pode apresentar interrupções e quase-interrupções, bem como interrupção do tipo divisão por um pivô zero associado com a fatoração *LDU* requerida internamente pelo algoritmo Bi-Lanczos [38]. Tais problemas podem ocasionar a falha do BiCG ou provocar instabilidades numéricas que impeçam sua convergência. Durante o processo iterativo a taxa de convergência varia drasticamente (aumenta e diminui) de uma iteração para outra. Portanto, o BiCG apresenta uma convergência irregular e oscilatória [38, 34]. Conclui-se preliminarmente que se trata de um método com robustez e eficiência computacional baixos.

O CGS é o método resultante de melhorias no BiCG, casos de ocorrências de interrupções e quase-interrupções ainda são possíveis [35], a exemplo do BiCG. Em 2000, Saad e Van der Vorst [39] citam que as mudanças realizadas no CGS amplificaram mais ainda as irregularidades e oscilações da sua convergência. Embora seja apontado como o método que apenas melhorou a eficiência computacional do BiCG, podendo ainda apresentar problemas de robustez (interrupções).

O GMRES é um método que não apresenta problema de interrupções nem quase-interrupções, como aqueles apresentados nos métodos baseados nos algoritmos de Lanczos e suas variações [18].

O BiCGStab, proposto por Henk van der Vorst em 1992 [33], surgiu como uma variante do CGS. Ocorreu modificação no algoritmo Bi-Lanczos com intuito de controlar e minimizar o resíduo. A finalidade foi proporcionar convergência mais rápida e suave em relação ao CGS, evitando-se irregularidades e oscilações. As alterações produzidas reduziram as interrupções e instabilidades numéricas embora existam relatos

de casos onde o processo de convergência deste método é interrompido [13]. No BiCGStab, a transposta da matriz de coeficientes não é calculada explicitamente o que permite obter um custo computacional mais aceitável [33].

A experiência obtida após sucessivos experimentos numéricos nestes estudos, mostra que a velocidade de convergência dos métodos iterativos para algumas aplicações é lenta. A taxa de convergência depende da qualidade de pré-condicionador com características próprias relacionadas ao problema em estudo [40]. Com esta finalidade, o próximo capítulo visa discutir e apresentar técnicas numéricas voltadas para o pré-condicionamento de sistemas lineares.

# Capítulo 3

# Proposta de Estrutura de Pré-condicionadores

## 3.1 INTRODUÇÃO

O uso direto por si só de métodos iterativos para solução de sistemas lineares, caracterizados por matrizes assimétricas e indefinidas, em geral, não é atrativo, por causa de possíveis dificuldades de convergência. Para contornar este inconveniente, utiliza-se o recurso de pré-condicionamento do sistema [26, 18]. Esta é uma prática reconhecida e encontrada em diversas aplicações [2, 6, 41, 21, 10, 7, 42, 43].

Além do processo de pré-condicionamento em si, o reordenamento da disposição das equações, bem como dos elementos das variáveis $\boldsymbol{x}$, desempenham papel relevante no processo de convergência dos métodos iterativos [44, 18].

Neste capítulo, com a finalidade de se determinar uma estrutura de pré-condicionador para um problema caracterizado por uma matriz de coeficientes de sistema linear específico, estuda-se e propõe-se estruturas baseadas em desacoplamento parcial da matriz $\boldsymbol{A}$. Os resultados são avaliados nos métodos CGS, GMRES, BiCG e BiCGStab.

## 3.2 CONCEPÇÃO DE PRÉ-CONDICIONAMENTO

Um pré-condicionador (*PC*) para o sistema da equação (1.1) consiste de uma matriz $\boldsymbol{T}$, cuja inversa multiplicada pela matriz de coeficientes do sistema linear, pela direita e(ou) esquerda, deveria gerar como resultado uma matriz próxima à identidade. Na situação em que o *PC* é multiplicado pela esquerda, deve-se resolver o sistema linear da equação (1.1) como:

$$\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{T}^{-1}\boldsymbol{b} \quad \Rightarrow \quad \overline{\boldsymbol{A}}\boldsymbol{x} = \overline{\boldsymbol{b}}\,, \text{ com } \overline{\boldsymbol{A}} = \boldsymbol{T}^{-1}\boldsymbol{A}, \quad \overline{\boldsymbol{b}} = \boldsymbol{T}^{-1}\boldsymbol{b} \tag{3.1}$$

e quando o *PC* é multiplicado pela direita, tem-se:

$$\boldsymbol{AT^{-1}y = b} \quad \Rightarrow \quad \boldsymbol{\widehat{A}y = b}, \text{ com } \boldsymbol{\widehat{A} = AT^{-1}}, \text{ sendo } \boldsymbol{x = T^{-1}y}. \qquad (3.2)$$

Tanto à esquerda em (3.1) quanto à direita em (3.2), um *PC* ideal seria o que tivesse $\boldsymbol{T = A}$. Nesta situação, ter-se-ia a solução direta na equação (3.1), mas, na equação (3.2), seria necessário resolver novo sistema linear, com grau de complexidade similar ao original. Além disso, em ambos os casos de *PC*, haveria necessidade do cálculo das matrizes $\boldsymbol{\overline{A}}$ e $\boldsymbol{\widehat{A}}$. No entanto, estas matrizes resultariam em matrizes cheias, mesmo que $\boldsymbol{A}$ fosse esparsa. Em função desses argumentos, não é conveniente se utilizar a própria matriz de coeficientes como *PC*. Por outro lado, procura-se definir $\boldsymbol{T}$ com: elementos próximos aos de $\boldsymbol{A}$ e inversa de fácil resolução.

Todavia, na prática, é mais conveniente se definir matrizes pré-condicionadoras à esquerda, $\boldsymbol{T_L}$, e à direita, $\boldsymbol{T_R}$, ambas com as características desejáveis mencionadas anteriormente, de modo que $\boldsymbol{T = T_L T_R}$ [26]. Com esta nova situação, os problemas das equações (3.1) e (3.2) são transformados em um único sistema linear com pré-condicionamentos simultâneos à esquerda e à direita:

$$\boldsymbol{T_L^{-1} A T_R^{-1} y = \widetilde{b}} \;\Rightarrow \boldsymbol{\widetilde{A}y = \widetilde{b}}, \; \boldsymbol{\widetilde{A} = T_L^{-1} A T_R^{-1}}, \; \boldsymbol{\widetilde{b} = T_L^{-1} b} \;\; \boldsymbol{x = T_R^{-1} y} \qquad (3.3)$$

### 3.2.1 Estrutura de pré-condicionador baseado em desacoplamento de matriz

A determinação de uma matriz de pré-condicionamento $\boldsymbol{T}$ não é uma tarefa trivial. No entanto, algumas classes de sistemas possuem estruturas peculiares, permitindo se tirar proveito de características benéficas para um *PC*.

Neste trabalho, propõe-se uma estrutura de um pré-condicionador montada com base na dominância de alguns blocos da matriz de coeficientes do sistema linear. Assume-se que os blocos removidos sejam substituídos por matrizes nulas, de igual dimensão. Este processo é desejável, principalmente quando se tem em mente aplicações envolvendo sistemas de grande porte. Para melhor compreensão do assunto, apresenta-se a seguir alguns conceitos que serão utilizados mais à frente.

Seja $\boldsymbol{A}$ uma matriz não-singular, assimétrica, quadrada, esparsa de ordem $n$ e $\boldsymbol{S}$ uma submatriz de $\boldsymbol{A}$, de ordem $p \times q$. Conforme $\boldsymbol{S}$, particiona-se $\boldsymbol{A}$ da seguinte forma:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \cdots & \boldsymbol{B} \\ \vdots & \boldsymbol{S} & \vdots \\ \boldsymbol{C} & \cdots & \boldsymbol{D} \end{bmatrix} \tag{3.4}$$

Experimentalmente, diz-se que a submatriz $\boldsymbol{S}$ é fracamente acoplada à matriz $\boldsymbol{A}$, caso a matriz resultante da substituição de $\boldsymbol{S}$ por uma matriz de zeros, definida por

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{A}_1 & \cdots & \boldsymbol{B} \\ \vdots & \boldsymbol{0} & \vdots \\ \boldsymbol{C} & \cdots & \boldsymbol{D} \end{bmatrix} \tag{3.5}$$

é não-singular e aproximadamente igual à $\boldsymbol{A}$, no sentido matemático de uma norma. Em outras palavras, a influência de $\boldsymbol{S}$ em $\boldsymbol{A}$ pode ser desprezível.

Uma avaliação do acoplamento relativo da submatriz $\boldsymbol{S}$ com respeito à matriz $\boldsymbol{A}$ é estudada neste trabalho. Propõem-se três indicadores, conforme definições a seguir.

**Definição 3.1:** o grau de esparsidade da matriz $\boldsymbol{T} \in \mathbb{R}^{n \times n}$ é definido por

$$\mathcal{S}(\boldsymbol{T}) \coloneqq \frac{\left(n^2 - n_{nz}(\boldsymbol{T})\right)}{n^2} \times 100\% \tag{3.6}$$

em que $n_{nz}(\boldsymbol{T})$ é o número de elementos não-nulos de $\boldsymbol{T}$. Em função de (3.6), apresenta-se uma definição complementar, denominada índice de ganho de esparsidade, $G(\boldsymbol{T})$, a qual consiste na razão entre o número de elementos nulos de $\boldsymbol{T}$, pela quantidade de zeros de $\boldsymbol{A}$. Esse índice pode ser definido como:

$$G(\boldsymbol{T}) \coloneqq \frac{\left(n^2 - n_{nz}(\boldsymbol{T})\right)}{(n^2 - n_{nz}(\boldsymbol{A}))} = 1 + \frac{(n_{nz}(\boldsymbol{A}) - n_{nz}(\boldsymbol{T}))}{(n^2 - n_{nz}(\boldsymbol{A}))} \geq 1 \tag{3.7}$$

Portanto, de acordo com (3.7), quanto mais $G(\boldsymbol{T})$ estiver afastado da unidade, maior a esparsidade da matriz $\boldsymbol{T}$. No entanto, esta característica apenas, em geral, não é suficiente para promover adequados benefícios ao *PC*.

**Definição 3.2:** Dominância de coluna $\mathcal{D}_C(\boldsymbol{S})$ e dominância de linha $\mathcal{D}_R(\boldsymbol{S})$, são definidos, respectivamente, por

$$\mathcal{D}_C(\boldsymbol{S}) = \|\boldsymbol{S}\|_1 \coloneqq \max_{1\le j\le q} \sum_{i=1}^{p} |s_{i,j}| \tag{3.8}$$

$$\mathcal{D}_R(\boldsymbol{S}) = \|\boldsymbol{S}\|_\infty \coloneqq \max_{1\le i\le p} \sum_{j=1}^{q} |s_{i,j}| \tag{3.9}$$

É possível também definir a dominância relativa de coluna, $\widetilde{\mathcal{D}}_C(\boldsymbol{S})$, e a dominância relativa de linha, $\widetilde{\mathcal{D}}_R(\boldsymbol{S})$, respectivamente, por

$$\widetilde{\mathcal{D}}_C(\boldsymbol{S}) = \frac{\|\boldsymbol{S}\|_1}{\|\boldsymbol{A}\|_1} \coloneqq \frac{\max_{1\le j\le q} \sum_{i=1}^{p} |s_{i,j}|}{\max_{1\le j\le n} \sum_{i=1}^{n} |a_{i,j}|} \tag{3.10}$$

$$\widetilde{\mathcal{D}}_R(\boldsymbol{S}) = \frac{\|\boldsymbol{S}\|_\infty}{\|\boldsymbol{A}\|_1} \coloneqq \frac{\max_{1\le i\le p} \sum_{j=1}^{q} |s_{i,j}|}{\max_{1\le j\le n} \sum_{i=1}^{n} |a_{i,j}|} \tag{3.10}$$

Neste trabalho, serão considerados apenas os índices $\mathcal{S}(\boldsymbol{T})$, $\mathcal{D}_C(\boldsymbol{S})$ e $\mathcal{D}_R(\boldsymbol{S})$.

Para uma matriz $\boldsymbol{A}$ particionada conforme em (3.4), diz-se que a submatriz $\boldsymbol{S}$ é fracamente acoplada em relação à matriz $\boldsymbol{A}$, quando os seguintes critérios são observados:

$$\mathcal{S}(\boldsymbol{A}) < \mathcal{S}(\boldsymbol{T}) \tag{3.11}$$

$$\mathcal{D}_C(\boldsymbol{S}) \ll \min_{i\ge 1, j\le q}\{\mathcal{D}_C(\boldsymbol{A}_{ij})\} \tag{3.12}$$

$$\mathcal{D}_R(\boldsymbol{S}) \ll \min_{i\geq 1, j\leq p}\{\mathcal{D}_R(\boldsymbol{A}_{ij})\} \tag{3.13}$$

Observa-se que a matriz $\boldsymbol{T}$ é sempre mais esparsa que a matriz original $\boldsymbol{A}$.

Considere que a matriz $\boldsymbol{A}$ possa ser particionada em quatro blocos, formados pelas submatrizes $\boldsymbol{A}_1$, $\boldsymbol{B}$, $\boldsymbol{C}$ e $\boldsymbol{D}$, de modo que o sistema linear (1.1) seja reapresentado da seguinte forma

$$\begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} \tag{3.14}$$

A avaliação da substituição da submatriz $\boldsymbol{B}$ por zeros a fim de se determinar uma matriz $\boldsymbol{T}$ para *PC* requer as seguintes condições:

$$\mathcal{D}_C(\boldsymbol{B}) \ll min\{\mathcal{D}_C(\boldsymbol{A}_1), \mathcal{D}_C(\boldsymbol{D})\} \tag{3.15}$$

$$\mathcal{D}_R(\boldsymbol{B}) \ll min\{\mathcal{D}_R(\boldsymbol{A}_1), \mathcal{D}_R(\boldsymbol{D})\} \tag{3.16}$$

e, para a submatriz $\boldsymbol{C}$,

$$\mathcal{D}_C(\boldsymbol{C}) \ll min\{\mathcal{D}_C(\boldsymbol{A}_1), \mathcal{D}_C(\boldsymbol{D})\} \tag{3.17}$$

$$\mathcal{D}_R(\boldsymbol{C}) \ll min\{\mathcal{D}_R(\boldsymbol{A}_1), \mathcal{D}_R(\boldsymbol{D})\}. \tag{3.18}$$

Evidentemente, as condições (3.15) a (3.18) são apenas necessárias. Elas fornecem informações preliminares sobre a possibilidade de desacoplamento entre os blocos da diagonal principal. Por outro lado, o insucesso no atendimento dessas condições evidencia que o referido bloco deve ser preservado na estrutura de pré-condicionador. Portanto, os indicadores fornecem subsídios numéricos que indicam possibilidade de descarte do bloco (substituição por zeros) avaliado.

### 3.2.2 Pré-condicionador baseado em desacoplamento de matriz

Considerando a estrutura da matriz de coeficientes do sistema linear em (3.14) e as condições elencadas de (3.15) a (3.18), são propostas três estruturas de *PC* baseadas em dominância de blocos da diagonal principal:

$$\boldsymbol{T}_1 = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D} \end{bmatrix}, \qquad \boldsymbol{T}_2 = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}, \quad \text{ou} \qquad \boldsymbol{T}_3 = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{D} \end{bmatrix} \tag{3.19}$$

Aplicações envolvendo o desempenho de pré-condicionadores baseados nessas matrizes serão estudadas nos Capítulos 6 e 7.

Em (3.19), deve se atentar ao requisito das matrizes $\boldsymbol{T}_1, \boldsymbol{T}_2$ e $\boldsymbol{T}_3$ serem não-singulares.

## 3.3 PRÉ-CONDICIONADOR BASEADO EM FATORAÇÃO *ILU*

As matrizes pré-condicionadoras $\boldsymbol{T}_1, \boldsymbol{T}_2$ e $\boldsymbol{T}_3$ utilizadas diretamente no sistema linear, como nas equações (3.1) ou (3.2), podem levar a convergência para a solução do sistema linear calculado iterativamente, mas este processo pode ser lento. Nesta situação, o tempo esperado para o cálculo pode ser inaceitável, se comparado a outras opções. Com o intuito de contornar essa dificuldade, propõe-se fatorar uma das três matrizes candidatas a *PC* em (3.19), na forma de um produto de duas matrizes. Assim, será possível utilizar a metodologia de resolução de sistema proposta na equação (3.2). A dificuldade consiste em se fatorar uma matriz em um produto de duas outras e ainda atender as condições esperadas de qualidade para *PC*.

Uma das ferramentas matemáticas amplamente utilizadas para se determinar as matrizes do produto consiste em se aplicar a técnica conhecida na literatura como fatoração $LU$ incompleta $(ILU)$ [18].

A determinação da matriz candidata $\boldsymbol{T}$ a pré-condicionador fundamentada em decomposição genérica deve atender a duas características fundamentais: o algoritmo de pré-condicionamento e a regra de preenchimento da matriz.

O algoritmo de pré-condicionadores baseados em fatoração $ILU$, normalmente, requer fatoração triangular (do tipo $IKJ$ ou $KIJ$) [45] para construir implicitamente $\boldsymbol{T}$ na forma de duas matrizes triangulares $\boldsymbol{L}_T$ e $\boldsymbol{U}_T$ $(\boldsymbol{T} = \boldsymbol{L}_T\boldsymbol{U}_T \approx \boldsymbol{A})$ [46]. Já a regra de preenchimento traduz o critério usado para decidir que elementos não-nulos devem fazer parte do pré-condicionador. As regras de preenchimento, em geral, usam parâmetros escalares cujos valores são escolhidos antes de se iniciar o processo de solução. O significado de cada parâmetro depende do critério associado a cada regra de preenchimento. A ideia fundamental é escolher os valores dos parâmetros que introduzam o mínimo de elementos não-nulos sem prejudicar a qualidade do pré-condicionador, isto é, sem torná-lo muito diferente de $\boldsymbol{A}$ e cujas inversas de $\boldsymbol{L}_T$ e $\boldsymbol{U}_T$ sejam mais fáceis de serem resolvidas que os termos $\boldsymbol{L}$ e $\boldsymbol{U}$ obtidos da fatoração $LU$ tradicional. A quantidade de parâmetros associados aos métodos iterativos com pré-condicionadores pode ser visto como uma desvantagem se comparados aos métodos diretos, já que estes não precisam de parâmetro algum.

Existem diversas modalidades de fatoração $ILU$ definidas de acordo com um parâmetro $m$, inteiro positivo, o qual especifica o nível de complexidade de preenchimento da matriz [18] usada para pré-condicionador. A notação usual adotada é $ILU(m)$.

### 3.3.1 Pré-condicionador *ILU* (0)

Esta estrutura é a que apresenta fatoração mais simples e de baixo custo computacional. O processo envolve uma decomposição da forma $\boldsymbol{A} = \boldsymbol{LU} - \boldsymbol{R}$, em que as matrizes triangulares $\boldsymbol{L}$ e $\boldsymbol{U}$, parte inferior e superior, respectivamente, somadas, têm a mesma estrutura de elementos não-nulos da matriz $\boldsymbol{A}$. A matriz $\boldsymbol{R}$ é o resíduo ou erro resultante da aproximação pelo produto $\boldsymbol{LU}$ [18].

A esparsidade padrão de uma matriz $\boldsymbol{A}$ é definida como:

$$P_A \coloneqq \{(i,j) | a_{(i,j)} \neq 0\} \tag{3.20}$$

onde $a_{(i,j)}$ é o $(i,j)$-ésimo elemento da matriz $\boldsymbol{A}$. Se todos os *fill-ins* nos fatores $\boldsymbol{L}$ e $\boldsymbol{U}$ são descartados, o algoritmo $ILU(0)$ é obtido.

Normalmente, em problemas reais envolvendo matrizes de grandes dimensões, é necessário preencher mais posições nos fatores triangulares do que os permitidos na metodologia do tipo $ILU(0)$. Portanto, para estes casos são necessárias regras de preenchimento mais complexas e sofisticadas que permitam adicionar novos elementos não-nulos aos fatores triangulares. No entanto, a escolha de quais novos elementos não-nulos devem ser adicionados, a fim de preencher apenas aqueles elementos que vão diminuir significativamente o erro. Este problema está associado às deficiências das heurísticas das regras de preenchimento dos pré-condicionadores convencionais. A seguir, apresentam-se outras metodologias de pré-condicionadores de fatoração incompleta denominadas $ILU(m)$ e $ILUT(\tau, p)$. Ambas utilizam o algoritmo de eliminação de Gauss $IKJ$ [45] com regras de preenchimento diferentes.

### 3.3.2 Pré-condicionador *ILU* (*m*)

Fatorações triangulares incompletas mais precisas são geralmente mais eficientes e confiáveis por se aproximarem mais da matriz de coeficientes, e por produzirem uma taxa de convergência da solução iterativa mais adequada. Este tipo de pré-condicionador se diferencia da classe $ILU(0)$ por permitir alguns preenchimentos de novos elementos não-nulos.

Por essa metodologia, a cada elemento que é processado através da eliminação de Gauss, associa-se um nível de preenchimento, sendo que, dependendo do valor desse nível, o elemento pode ser preenchido ou descartado.

Para melhorar a eficiência do pré-condicionador $ILU$, esquemas especiais para descartar os *fill-ins* tem sido desenvolvidos [18]. Os *fill-ins* são descartados baseados em diferentes critérios, tais como: posição, valor, ou uma combinação dos dois [26]. Uma escolha é descartar os *fill-ins* baseado no conceito conhecido como nível de *fill-in*.

O nível de preenchimento indica de forma aproximada a magnitude dos elementos; quanto maior o nível, menor a magnitude dos elementos. Em [18] emprega-se o quantificador $\epsilon < 1$ para se avaliar a pertinência de descarte ou não de um elemento, ao qual se atribui um nível de preenchimento $m$. Um elemento não-nulo inicia com um nível de preenchimento 1 (esse valor muda durante a implementação computacional) e um elemento zero tem nível de preenchimento infinito [18]. A regra de preenchimento é utilizada apenas para decidir quais os novos elementos não-nulos devem ser preenchidos ou substituídos por zero. Os aspectos negativos apresentados por [13, 18] associados ao pré-condicionador $ILU(m)$ são os seguintes:

a) Devido ao fato da quantidade de novos elementos não-nulos preenchidos depender do valor de $m$, então, para $m > 0$, não é possível estimar a quantidade de elementos não-nulos e o esforço computacional necessário para se obter a fatoração $ILU(m)$.

b) O custo associado à atualização dos níveis de preenchimento de elementos não-nulos pode ser elevado.

c) O nível de preenchimento de elementos não-nulos para matrizes indefinidas pode não ser um bom indicador da magnitude dos elementos que estão sendo descartados e o algoritmo pode descartar elementos de grandes magnitudes, resultando em imprecisão da fatoração incompleta, no sentido que $\|\boldsymbol{R}\| = \|\boldsymbol{LU} - \boldsymbol{A}\|$ pode não ser pequeno. A prática tem mostrado que em média isto pode levar a um grande número de iterações para se obter a convergência.

d) Não existe um método que permita escolher apropriadamente o valor de $m$. Pois, a escolha de $m$ depende do problema que está sendo solucionado. Normalmente $m$ é escolhido de forma empírica para apenas um sistema linear específico. Em problemas que dependam da solução de diversos sistemas lineares, sequencialmente, uma restrição é um valor ou faixa de valores adequados para $m$ tal que a solução de todos os sistemas lineares seja alcançada no menor tempo de execução possível. Estes aspectos negativos

sugerem que o pré-condicionador $ILU(m)$ com uma regra baseada nos níveis de preenchimento, pode não ser o mais indicado para alguns sistemas.

Alternativas mais econômicas similares à técnica $ILU(m)$ foram desenvolvidas. Estes métodos são denominados $ILUT$ e $ILUTP$ [45, 47, 48, 49]. São métodos relativamente robustos e eficientes mas, como os demais podem falhar [18]. O método $ILUT$ usa uma estratégia *dual dropping*[4] para controlar a quantidade de *fill-in* gerada durante a fatoração. Sua implementação é baseada na variante $IKJ$ da eliminação gaussiana [45]. Na decomposição conhecida como $ILUT(\tau, p)$, o controle de *fill-in* é feito mediante a aplicação da estratégia *dual dropping* nos fatores gerados durante a eliminação gaussiana. A exatidão do processo de fatoração $ILUT(\tau, p)$ é função dos dois parâmetros de fatoração de truncamento; a *drop tolerance* $(\tau)$ e o nível de *fill-in* $(p)$. A seguir, descrevem-se as características deste tipo de pré-condicionador com maiores detalhes.

### 3.3.3 Pré-condicionador $ILUT$ $(\boldsymbol{\tau}, \boldsymbol{p})$

Pré-condicionadores cujas regras são baseadas somente no nível de preenchimento consideram apenas a posição dos elementos (estrutura da matriz $\boldsymbol{A}$), sem levar em conta o valor numérico dos elementos dos fatores triangulares [18]. Portanto, este tipo de pré-condicionador pode levar à perda da robustez e velocidade de convergência dos métodos em estudo e causar problemas identificados na seção anterior em aplicações práticas e reais. Uma estratégia alternativa conhecida como $ILUT(\tau, p)$ é apresentada em [45], também baseada na eliminação de Gauss que além da regra de preenchimento, considera também a magnitude dos elementos. Aqui, $p$ é um número inteiro que limita a quantidade máxima de elementos permitidos em cada linha dos fatores triangulares $\boldsymbol{L}$ e $\boldsymbol{U}$. Já $\tau$ é um número real (tolerância) usado para descarte dos elementos considerados de pequena magnitude. Basicamente, $p$ está associado ao controle do uso da memória, enquanto que $\tau$ contribui para reduzir o esforço computacional [26, 18].

[4] *Dual dropping* relaciona-se a duplo truncamento, ou seja, está associado a dois parâmetros de truncamento

A regra de preenchimento com duplo parâmetro limitante obedece a uma ideia central de descartar apenas os menores elementos (em valor absoluto) dos fatores triangulares. Neste caso, aparentemente, quando os menores elementos são descartados se produz um menor erro do que quando se descarta os maiores. A regra tem como base duas heurísticas: uso do parâmetro $\tau$ para decidir quais elementos de pequena magnitude devem ser descartados; e a outra, usa $p$ para decidir quantos elementos de grande magnitude serão preenchidos em cada linha dos fatores triangulares. Esta regra de preenchimento é chamada de *duplo parâmetro limitante*. Em síntese, dado um elemento $w_k$ este é substituído por zero se seu módulo for inferior a uma tolerância relativa $\tau_i$ obtida multiplicando-se a tolerância $\tau$ pela norma da i-ésima linha da matriz de coeficientes original. Entretanto, são mantidos apenas os $p$ maiores elementos de cada linha no fator $\boldsymbol{L}$ e no fator $\boldsymbol{U}$, em adição aos elementos diagonais, que são sempre mantidos. O cálculo de $w_k$ é efetuado de acordo com o Algoritmo 3.1.

**Algoritmo 3.1** – Fatoração padrão $ILUT(\tau, p)$ [18]

**ENTRADA:** Matriz $\boldsymbol{T}$

**SAÍDA:** Fatores $\boldsymbol{L}_T$ e $\boldsymbol{U}_T$

1. Executar uma fatoração $LU$ completa de $\boldsymbol{T}$
2. Repetir $i = 2{:}\,n$, Faça
3. $w := t_{i,:}$
4. Repetir $k = 1{:}\,i-1$ e quando $w_k \neq 0$, Faça
5. $w_k := w_k / t_{k,k}$ (quando $t_{k,k} \neq 0$)
6. Aplicar uma regra de truncamento para $w_k$
7. Se $|w_k| < \tau_i * nvg(|t_{i,:}|)$
8. Definir $w_k := 0$
9. Caso contrário
10. $w := w - w_k * u_{k,:}$
11. Fim-Se
12. Fim-Repetir $k$
13. Aplicar uma regra de truncamento para linha $w$ – conservar os maiores $p$ elementos em $\boldsymbol{L}_T$ e $\boldsymbol{U}_T$
14. Definir $l_{i,j} := w_j$, repetir $j = 1, \ldots, i-1$ enquanto $w_j \neq 0$
15. Definir $u_{i,j} := w_j$, repetir $j = i, \ldots, n$ enquanto $w_j \neq 0$
16. Definir $w := 0$
17. Fim-Repetir $i$

No Algoritmo 3.1, $w$ é um elemento de matriz, $t_{i,:}$ e $u_{k,:}$ denotam a i-ésima e k-ésima linhas de $\boldsymbol{T}$ e $\boldsymbol{U}$, respectivamente; e $nvg(|t_{i,:}|)$ é uma função que retorna a magnitude média dos valores absolutos dos elementos não-nulos da i-ésima linha da matriz $\boldsymbol{T}$. Assim, no passo 6 aplica-se o truncamento (substituição por zero) do elemento $w_k$, caso ele seja inferior à tolerância relativa $\tau_i$. No passo 13, uma regra de truncamento diferente é aplicada, ou seja, mantém o truncamento para qualquer elemento de linha com magnitude inferior à tolerância relativa $\tau_i$ e conserva apenas os $p$ maiores

elementos em $\boldsymbol{L}_T$ e $p$ maiores elementos em $\boldsymbol{U}_T$ em adição aos elementos diagonais que são sempre conservados.

A fatoração $ILUT$ poderá falhar por algumas razões. A mais óbvia é aquela onde a matriz original a ser fatorada contenha um pivô zero, como é o caso de elemento nulo na diagonal. Uma alternativa possível para contornar este problema é aplicar a técnica de pivoteamento por coluna. Este procedimento leva a uma variante de pivoteamento por coluna do método $ILUT$ que é denominada de $ILUTP$ [50]. A decomposição em questão é mais robusta que $ILUT$, mas frequentemente ao custo adicional no uso de memória, visto que o procedimento tende a gerar mais *fill-in*. Para ambos os casos, uma regra empírica eficaz é tomar um valor de $p$ elevado e usar o valor de $\tau$ para controlar a quantidade de *fill-in*. Isso geralmente resulta em bons resultados sem comprometer a eficiência da memória [50]. Por isso, este será o tipo de fatoração adotado nesta tese.

## 3.4 REORDENAMENTO

Uma vez que o pré-condicionador é baseado em fatoração triangular, torna-se imprescindível considerar alguma estratégia de reordenamento que permita reduzir o número de novos elementos não-nulos nos fatores triangulares e, consequentemente, reduzir o número de operações realizadas permitindo a melhoria na estabilidade da fatoração incompleta [26]. Assume-se que o reordenamento seja efetuado tendo como referência a matriz $\boldsymbol{A}$.

Técnicas de reordenamento da matriz $\boldsymbol{A}$ [51, 52, 53, 54], tendem a melhorar a convergência dos métodos iterativos reduzindo o número de iterações e o tempo de processamento [44]. Como consequência, a obtenção de matrizes pré-condicionadoras baseadas nesta matriz reordenada também tendem a melhorar o desempenho de algoritmos para resolução iterativa de sistemas lineares [26, 18].

A técnica de reordenamento é realizada mediante permutações de linhas, colunas ou ambas de uma matriz. Esta permutação é entendida como o produto da matriz $\boldsymbol{A}$ com matrizes de permutação $\boldsymbol{R}_L$ e $\boldsymbol{R}_R$, definidas a partir de permutações de linhas e colunas,

respectivamente, da matriz identidade. Caso $\boldsymbol{R}_L$ seja usada para multiplicar $\boldsymbol{A}$ à esquerda, trocam-se as linhas de $\boldsymbol{A}$. No caso da multiplicação envolver $\boldsymbol{R}_R$ pela direita, as colunas são trocadas e para ambas as multiplicações, as linhas e colunas são reposicionadas simultaneamente [18, 26]. Assim, (1.1) pode ser substituído pelo seguinte sistema equivalente reordenado

$$\boldsymbol{R}_L\boldsymbol{A}\boldsymbol{R}_R\boldsymbol{y} = \boldsymbol{R}_L\boldsymbol{b} \qquad \boldsymbol{x} = \boldsymbol{R}_R\boldsymbol{y} \tag{3.21}$$

O efeito produzido é uma alteração estrutural da matriz $\boldsymbol{A}$ de coeficientes sem modificar suas propriedades espectrais, ou seja, permanecem inalterados o número de condicionamento e os autovalores, mantendo assim o sistema resultante equivalente ao sistema linear original. No entanto, os benefícios oriundos do processo de reordenamento são bastante significativos, como será demonstrado em capítulos subsequentes.

Alguns métodos de reordenamento são encontrados na literatura [52, 55, 4, 56]. Estratégias de reordenamento baseadas nos métodos: *reverse* Cuthill-McKee (*RCM*) e variantes de ordenamento de grau mínimo, conhecidas por *Approximate Minimum Degree* (*AMD*) são ilustradas a seguir.

A Fig. 3.1a ilustra a visualização de estrutura típica de uma matriz $\boldsymbol{A}$ com $n = 6355$ e com estruturas reordenadas por *RCM* (Fig. 3b) e *AMD* (Fig.3c).

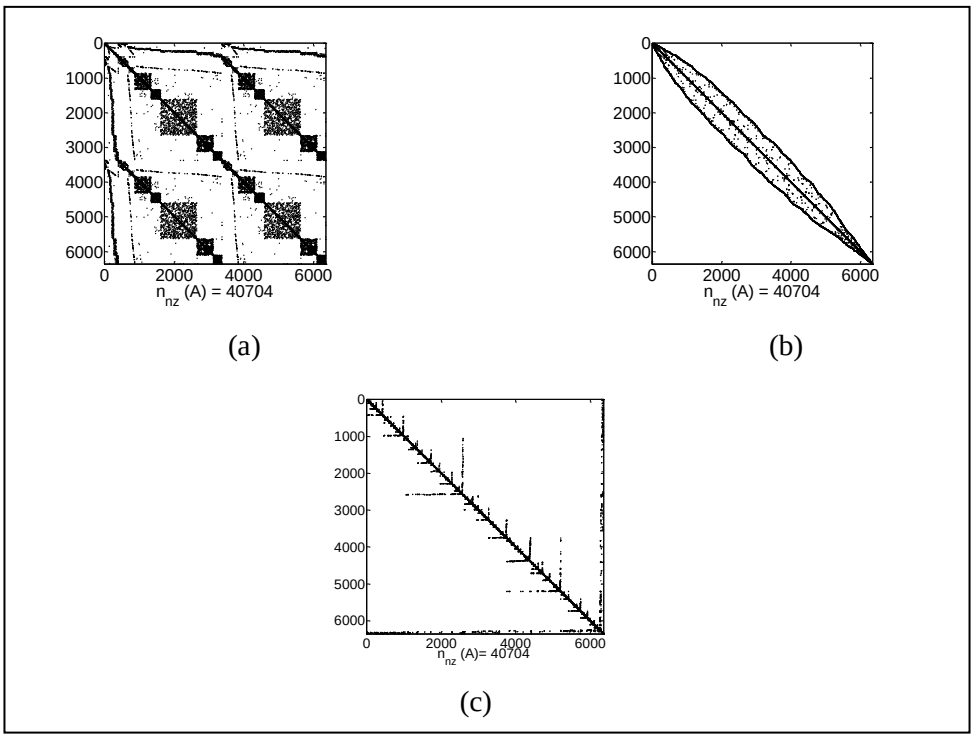



**Fig. 3.1** – Estruturas de uma matriz exemplo e suas formas reordenadas:
a) matriz ***A*** b) após *RCM* c) após *AMD*

Observam-se efeitos distintos dos reordenadores. Em todos os casos, o número de elementos não-nulos é o mesmo da matriz original. No método *RCM* os elementos não-nulos são confinados a uma banda estreita da matriz (altera largura de banda). Já no *AMD*, o padrão de esparsidade da matriz é definido com base no conceito de quociente dos grafos [57], evitando o estreitamento de banda. Portanto, a aplicação de um ou de outro método pode ser vantajosa [58, 59].

Nos próximos capítulos, são efetuadas avaliações de desempenho dos métodos iterativos para sistemas lineares, em uma aplicação de sistemas elétricos de potência, considerando-se uma nova estrutura de pré-condicionador e testes levando em conta efeitos de reordenamento.

# Capítulo 4

# Aplicação ao Problema de Fluxo de Carga

## 4.1 INTRODUÇÃO

Os métodos iterativos propostos nos capítulos anteriores, consistindo de uma combinação adequada de pré-condicionador baseado em desacoplamento, reordenamento e métodos iterativos clássicos, podem ser utilizados para obter solução de um sistema linear esparso invertível genérico. Como decorrência, os métodos encontrados na literatura possuem ampla aplicação, como na resolução dos subproblemas lineares associados aos algoritmos de resolução de sistemas não-lineares [7, 60, 1, 51, 61], algoritmos de otimização quadrática [62, 63], resolução de equações diferenciais parciais [64, 16], entre outros [6, 65].

Para ilustrar a potencial melhoria de desempenho de algoritmos para solução de equações não-lineares, utilizam-se os métodos propostos para se formular o problema de fluxo de carga [51] e destacar os pontos que serão alvo de interesse para avaliação da eficácia das metodologias propostas nesta tese. Com este objetivo, formula-se o problema de fluxo de carga e sua versão estendida associada ao cálculo de máximo carregamento no sistema [66].

## 4.2 FORMULAÇÃO DO PROBLEMA DE FLUXO DE CARGA

Para a formulação do problema é apresentado a seguir, inicialmente, o método mais utilizado na solução de sistemas não-lineares bem como sua aplicação no problema de fluxo de carga.

### 4.2.1 Método de Newton-Raphson

O método de Newton-Raphson [3, 67] é o mais utilizado para se obter as raízes reais de sistemas de equações não-lineares da forma

$$\mathbf{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(x_1, x_2, \cdots, x_n) \\ f_2(x_1, x_2, \cdots, x_n) \\ \vdots \\ f_n(x_1, x_2, \cdots, x_n) \end{bmatrix} = 0 \tag{4.1}$$

onde o vetor real $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^{\mathrm{T}}$ é a incógnita e $\mathbf{f}(\boldsymbol{x})$ é uma função não-linear com jacobiana invertível na região de busca das raízes de interesse.

O método de Newton-Raphson consiste em fornecer iterativamente estimativas para essas raízes, por meio de linearizações sucessivas da função $\mathbf{f}(\boldsymbol{x})$ em torno da raiz aproximada na iteração $k$, $\boldsymbol{x}^{(k)}$. Uma nova e possivelmente melhor aproximação para a raiz é dada por

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta\boldsymbol{x}^{(k)} \tag{4.2}$$

onde a correção $\Delta\mathbf{x}^{(k)}$ é solução do sistema linear

$$\boldsymbol{J}^{(k)}\Delta\boldsymbol{x}^{(k)} = -\mathbf{f}\left(\boldsymbol{x}^{(k)}\right) \tag{4.3}$$

em que $\boldsymbol{J}^{(k)} = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}^{(k)}}$ é a matriz jacobiana de $\mathbf{f}(\boldsymbol{x})$ calculada em $\boldsymbol{x}^{(k)}$. O critério de parada é dado por $\| \mathbf{f}\left(\boldsymbol{x}^{(k)}\right) \| < \varepsilon,$ para $\varepsilon > 0$ suficientemente pequeno, uma vez que havendo convergência, naturalmente, ao longo das iterações $\mathbf{f}(\boldsymbol{x})$ tenderá a zero.

Observa-se que o algoritmo para ser implementado no método de Newton-Raphson depende do particular método de resolução do problema linear da equação (4.3) implementado. Como a equação (4.3) precisa ser resolvida a cada iteração, é óbvio que a diminuição no tempo computacional no método linear leva a diminuição significativa no tempo computacional total gasto para se resolver o problema não-linear. Em algumas estratégias de solução, além de utilizar o algoritmo linear mais rápido, costuma-se acelerar as iterações pelo expediente de se calcular a jacobiana (e consequentemente, obter o pré-condicionador, o ordenamento e fatoração correspondentes) apenas

ocasionalmente. O algoritmo assim obtido é conhecido como Newton-Raphson desonesto ou inexato [60, 61]. Um caso extremo ocorre quando o algoritmo calcula a jacobiana apenas na primeira iteração.

A demonstração da efetividade do algoritmo resultante de Newton-Raphson com o método proposto para o subproblema linear será apresentado mais à frente, aplicando-o para resolver o problema de fluxo de carga.

### 4.2.2 Método de Newton-Raphson aplicado ao problema de fluxo de carga

O problema de fluxo de carga (*PFC*) em sistemas elétricos de potência é formulado com base em equações algébricas não-lineares do tipo expresso em (4.1). A solução $\boldsymbol{x}$ deste sistema fornece o ponto de operação do sistema, cujos elementos são chamados de estados do sistema [3]. O resultado é verificado para um determinado instante de tempo (*snapshot*), no qual assume-se que o sistema elétrico funcione em regime permanente. É neste contexto, que se insere a aplicação do método de Newton-Raphson. Na sequência, são levantadas as equações do sistema não-linear, na forma (4.1) e, a partir destas, deduzem-se as equações que caracterizam e definem o subproblema linear na expressão (4.3).

#### 4.2.2.1 Equações de balanço de potência

Na formulação do *PFC*, é necessário definir os tipos de barra. Na abordagem clássica, definem-se três tipos, conforme haja carga e (ou) geração conectada(s) a um barramento. O objetivo é se determinar os estados em todos os barramentos do sistema. Com isto, torna-se possível calcular fluxos e correntes pelos equipamentos. Os estados do sistema são a magnitude, $V_i$, e o ângulo de fase, $\theta_i$, da tensão da barra $i$. Portanto a partir destas duas variáveis, define-se o fasor da tensão na barra $i$, na forma polar, como $\overline{V_i} = V_i e^{j\theta_i}$.

Na metodologia aqui adotada, supõe-se que o sistema seja síncrono. Ou seja, tenha uma única barra *swing* (barra $V\theta$). Neste tipo de barra, os estados, definidos como a magnitude da tensão e o ângulo de fase da barra, são dados conhecidos. Já nas barras ditas de carga (barras $PQ$), é preciso calcular tanto a magnitude quanto o ângulo de fase

da tensão. Além disso, nesse tipo de barra, são escritas duas equações, correspondentes ao balanço de potências ativa e reativa no barramento. Por fim, define-se o terceiro tipo de barra, conhecida como de geração (barra $PV$). Aqui, a potência ativa que é injetada na barra é conhecida, assim como a magnitude da tensão na barra. Mas, a potência reativa é uma incógnita. Por esta razão, existe apenas uma equação de balanço de potência e uma única incógnita que precisa ser calculada (a potência reativa é uma variável redundante, função das magnitudes e ângulo de fase). Neste caso, o estado é o ângulo de fase da tensão.

Em vista do exposto, para um sistema com $N_b$ barras, sendo $N_{PQ}$ barras $PQ$ e $N_{PV}$ barras $PV$, são necessárias $n = 2N_{PQ} + N_{PV}$ equações. Consequentemente, devendo ser calculados $n$ estados.

As equações de balanço de potência em uma barra $PQ$ são [16]:

$$\Delta \boldsymbol{P}_i = (1 + p)\boldsymbol{P}_i^{esp} - \boldsymbol{P}_i(\boldsymbol{V}, \boldsymbol{\theta}) = 0 \tag{4.4}$$

$$\Delta \boldsymbol{Q}_i = (1 + p)\boldsymbol{Q}_i^{esp} - \boldsymbol{Q}_i(\boldsymbol{V}, \boldsymbol{\theta}) = 0 \tag{4.5}$$

onde $\boldsymbol{P}_i^{esp}$ e $\boldsymbol{Q}_i^{esp}$ são valores especificados de potência ativa e reativa líquidas injetadas na barra $i$, respectivamente; $i = 1,2, \dots, N_{PQ}$; $\boldsymbol{P}_i(\boldsymbol{V}, \boldsymbol{\theta})$ e $\boldsymbol{Q}_i(\boldsymbol{V}, \boldsymbol{\theta})$ são expressões obtidas a partir das leis de Kirchhoff e lei de Ohm, e que são dependentes das interligações e componentes da rede elétrica; $\Delta \boldsymbol{P}_i$ e $\Delta \boldsymbol{Q}_i$ são *mismatches* (desvios) de potência ativa e reativa, respectivamente, os quais, numericamente, devem ser próximos de zero já que os estados são calculados de forma iterativa e, portanto, devem ser precisos, considerada uma tolerância mínima aceitável; $p \geq 0$ é um escalar, denominado parâmetro da continuação, utilizado para se avaliar o efeito do carregamento e, consequentemente, o ponto de máximo carregamento (ponto de colapso de tensão) [68, 66].

A rigor, nas expressões $\boldsymbol{P}_i^{esp}$ e $\boldsymbol{Q}_i^{esp}$ poderiam ser incluídas também expressões das quais dependem a carga. Em geral, influenciadas pela magnitude da tensão da barra de carga. No entanto, para efeito de simplificação, sem perda de generalidade para análise do *PFC*, assume-se que *as cargas serão representadas somente por potência constante* (parte ativa e reativa). Adicionalmente, em $\boldsymbol{P}_i^{esp}$ e $\boldsymbol{Q}_i^{esp}$ poderão ser inseridas contribuições de geração equivalente. Mas, para efeito da análise neste trabalho, todos os dados convencionados como do tipo $PQ$, serão assumidos como dados de carga.

As expressões para $P_i(\boldsymbol{V}, \boldsymbol{\theta})$ e $Q_i(\boldsymbol{V}, \boldsymbol{\theta})$ são dadas por [3]:

$$P_i(V, \theta) = V_i \sum_{\kappa \in \Omega_i} (G_{i\kappa} cos\theta_{i\kappa} + \mathrm{B}_{i\kappa} sen\theta_{i\kappa})\, V_\kappa \tag{4.6}$$

$$Q_i(V, \theta) = V_i \sum_{\kappa \in \Omega_i} (G_{i\kappa} sen\theta_{i\kappa} + \mathrm{B}_{i\kappa} cos\theta_{i\kappa})\, V_\kappa \tag{4.7}$$

em que $\Omega_i$ é definido como um conjunto de barras κ, inclusive a *swing*, que possuem conexão com a barra $i,\ i = 1,2,\dots,(N_b - 1)$; $G_{i\kappa}$ e $\mathrm{B}_{i\kappa}$ são à condutância (parte real) e a susceptância (parte imaginária) relativas ao elemento $Y_{i\kappa} = G_{i\kappa} + j\mathrm{B}_{i\kappa}$ da matriz de admitância de barras do sistema (matriz $\boldsymbol{Y}_{bus}$); $\theta_{i\kappa} = \theta_i - \theta_\kappa$ é a abertura angular entre as fases das barras *i* e *k*.

Em barras $PV$, utilizam-se apenas equações de balanço de potência ativa do tipo (4.4), consequentemente, também da forma (4.6).

Na forma linearizada da equação (4.3), considerando-se os respectivos tipos de barras, as equações (4.4) e (4.5), em torno de $\boldsymbol{x}^{(k)} = \left[\left[\boldsymbol{\theta}^{(k)}\right]^T \left[\boldsymbol{V}^{(k)}\right]^T\right]^T$, por conveniência, são arranjadas e colocadas na seguinte forma:

$$\begin{bmatrix} \Delta \boldsymbol{P}_{barras\ PV\ e\ PQ} \\ \Delta \boldsymbol{Q}_{barras\ PQ} \end{bmatrix} = -\boldsymbol{J}^{(k)} \Delta \boldsymbol{x}^{(k)} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\theta}_{barras\ PV\ e\ PQ} \\ \Delta \boldsymbol{V}_{barras\ PQ} \end{bmatrix} = \boldsymbol{b} \tag{4.8}$$

onde,

$$\boldsymbol{A}_1 = \partial \boldsymbol{P}_{barras\ PV\ e\ PQ} / \partial \boldsymbol{\theta}_{barras\ PV\ e\ PQ} \tag{4.9}$$

$$\boldsymbol{B} = \partial \boldsymbol{P}_{barras\ PV\ e\ PQ} / \partial \boldsymbol{V}_{barras\ \ PQ} \tag{4.10}$$

$$\boldsymbol{C} = \partial \boldsymbol{Q}_{barras\ PQ} / \partial \boldsymbol{\theta}_{barras\ PV\ e\ PQ} \tag{4.11}$$

$$\boldsymbol{D} = \partial \boldsymbol{Q}_{barras\ \ PQ} / \partial \boldsymbol{V}_{barras\ \ PQ} \tag{4.12}$$

Em (4.8), $\boldsymbol{b}$ representa os *mismatches*, sendo o vetor independente do sistema linear. $\boldsymbol{A}_1, \boldsymbol{B}, \boldsymbol{C}$ e $\boldsymbol{D}$ são submatrizes da matriz jacobiana, cujos elementos são definidos da seguinte forma [3]:

**Matriz $\boldsymbol{A}_1$**

$$A_{1ii} = \frac{\partial P_i}{\partial \theta_i} = -V_i^2 B_{ii} - V_i \sum_{\kappa \in \Omega_{i \neq k}} V_\kappa (G_{i\kappa} sen\theta_{i\kappa} - \mathrm{B}_{i\kappa} cos\theta_{i\kappa}) \tag{4.13}$$

$$A_{1i\kappa} = \frac{\partial P_i}{\partial \theta_\kappa} = V_i V_\kappa (G_{i\kappa} sen\theta_{i\kappa} - \mathrm{B}_{i\kappa} cos\theta_{i\kappa}) \tag{4.14}$$

$i = 1,2, \dots, (N_b - 1)$ e $\kappa \in \Omega_i$

**Matriz $\boldsymbol{B}$**

$$B_{ii} = \frac{\partial P_i}{\partial V_i} = V_i G_{ii} + \sum_{\kappa \in \Omega_{i \neq k}} V_\kappa (G_{i\kappa} cos\theta_{i\kappa} + \mathrm{B}_{i\kappa} sen\theta_{i\kappa}) \tag{4.15}$$

$$B_{i\kappa} = \frac{\partial P_i}{\partial V_\kappa} = V_i (G_{i\kappa} cos\theta_{i\kappa} + \mathrm{B}_{i\kappa} sen\theta_{i\kappa}) \tag{4.16}$$

$i = 1,2, \dots, (N_b - 1)$ e $\kappa \in PQ$

**Matriz $\boldsymbol{C}$**

$$C_{ii} = \frac{\partial Q_i}{\partial \theta_i} = -V_i^2 G_{ii} + V_i \sum_{\kappa \in \Omega_{i \neq k}} V_\kappa (G_{i\kappa} cos\theta_{i\kappa} + \mathrm{B}_{i\kappa} sen\theta_{i\kappa}) \tag{4.17}$$

$$C_{i\kappa} = \frac{\partial Q_i}{\partial \theta_\kappa} = -V_i V_\kappa (G_{i\kappa} cos\theta_{i\kappa} + \mathrm{B}_{i\kappa} sen\theta_{i\kappa}) \tag{4.18}$$

$i = 1,2, \dots, N_{PQ}$ e $\kappa \in \Omega_i$

**Matriz *D***

$$D_{ii} = \frac{\partial Q_i}{\partial V_i} = -V_i B_{ii} + \sum_{\kappa \in \Omega_{i \neq k}} V_\kappa (G_{i\kappa} sen\theta_{i\kappa} - B_{i\kappa} cos\theta_{i\kappa}) \quad (4.19)$$

$$D_{i\kappa} = \frac{\partial Q_i}{\partial V_\kappa} = V_i (G_{i\kappa} sen\theta_{i\kappa} - B_{i\kappa} cos\theta_{i\kappa}) \quad (4.20)$$

$$i = 1,2,\dots,N_{PQ} \text{ e } \kappa \in PQ$$

4.2.2.2 Cálculo iterativo da solução do sistema de equações não-lineares

Nas expressões (4.4) e (4.5), será denominado fluxo de carga base aquele que ocorre quando $p = 0$. No caso em que $p$ é variado, o tratamento é dado de outra forma, porém, também através da resolução de problema de fluxo de carga e será discutido na Seção 4.4.

A estratégia para se determinar a solução do sistema de equações não-lineares (4.4) e (4.5) pelo método de Newton-Raphson, que proporciona o cálculo dos estados do sistema, é destacada no fluxograma da Figura 4.1 e estruturada no Algoritmo 4.1. No fluxograma, os dados de rede servem para montagem da matriz de admitância de rede (matriz $\boldsymbol{Y}_{bus}$). Os tipos de barra servem para identificar barras de carga, de geração e a *swing*. Os parâmetros de simulação são necessários para o controle de processo de convergência do processo não-linear. Conforme já definido, $\varepsilon$ corresponde à tolerância admitida para a norma dos *mismatches*; *k* é um contador do número de iterações e $k_{max}$ é o número máximo de iterações admitidas para convergência. A estimativa dos estados iniciais do sistema, em geral, é definida a partir do conhecimento de estados relativos a pontos de operação anteriores do sistema. Na ausência desta informação, é prático se atribuir magnitude de tensão igual a 1 pu (tensão nominal) nos barramentos de carga; e ângulo de fase igual a zero em todos os barramentos.

Apesar dos vários blocos no fluxograma da Figura 4.1, o maior volume de cálculo ocorre no procedimento que está no bloco que indica a resolução do sistema linear de equações.

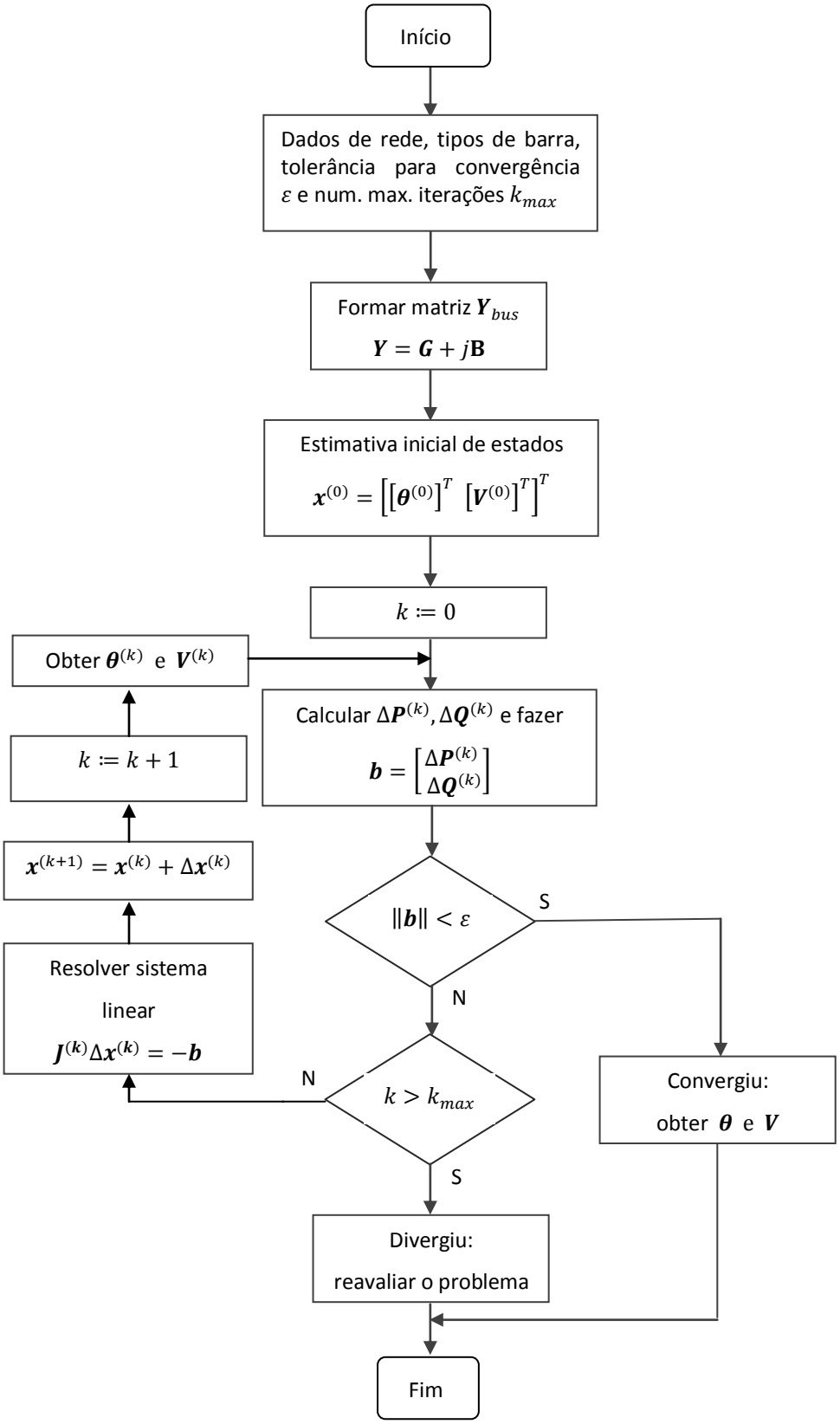



**Figura 4.1** – Fluxograma para resolução do problema de fluxo de carga básico.

**Algoritmo 4.1 -** Método de Newton-Raphson básico associado ao *PFC*

**Entrada:** estimativa inicial $\boldsymbol{x}^{(0)} = \left[\left[\boldsymbol{\theta}^{(0)}\right]^T \ \left[\boldsymbol{V}^{(0)}\right]^T\right]^T$, dados de geração (barras *PV*) e de carga (*PQ*), matriz $\boldsymbol{Y}_{bus}$, número máximo de iterações, $k_{max}$, tolerância para convergência $\varepsilon$

**Saída:** estados $\boldsymbol{V}$ e $\boldsymbol{\theta}$

1) Iniciar com $k = 0$ e calcular os *mismatches* $\Delta\boldsymbol{P}^{(0)}$ e $\Delta\boldsymbol{Q}^{(0)}$ através de (4.4) e (4.5)
2) Calcular os elementos das submatrizes $\boldsymbol{A}_1^{(k)}, \boldsymbol{B}^{(k)}, \boldsymbol{C}^{(k)}$ e $\boldsymbol{D}^{(k)}$ da matriz jacobiana, usando as expressões das equações (4.13) a (4.20)
3) Resolver o sistema linear (4.8) para os desvios de estado $\Delta\boldsymbol{\theta}^{(k)}$ e $\Delta\boldsymbol{V}^{(k)}$
4) Atualizar os valores dos estados $\boldsymbol{V}^{(k+1)} = \boldsymbol{V}^{(k)} + \Delta\boldsymbol{V}^{(k)}$ e $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}^{(k)}$
5) Fazer $k = k + 1$
6) Calcular os *mismatches* $\Delta\boldsymbol{P}^{(k)}$ *e* $\Delta\boldsymbol{Q}^{(k)}$ e $erro = \max\{||\Delta\boldsymbol{P}^{(k)}||_\infty, \ ||\Delta\boldsymbol{Q}^{(k)}||_\infty\}$
7) Se $erro < \epsilon$, o processo convergiu: ir para o passo 9; caso contrário, continuar no passo 8
8) Verificar se $k < k_{max}$ : em caso positivo, retornar para o passo 2; senão, encerrar o processo, pois o processo não convergiu para o número de iterações e tolerância fixados
9) Os estados calculados são: $\boldsymbol{V} = \boldsymbol{V}^{(k)}$ e $\boldsymbol{\theta} = \boldsymbol{\theta}^{(k)}$, fim

No algoritmo básico para cálculo dos estados, não está sendo levado em conta os limites de reativo, restrição que em análise de fluxo de carga deve ser considerada. Estes são devidamente inseridos nas aplicações que serão estudadas mais à frente. A ideia na ocorrência desta situação é tratar o limite fixando-se $Q_i$ no seu limite violado e transformar a barra do tipo *PV* para *PQ*. Neste caso, na iteração em andamento, haverá aumento do número de equações e de estados, além da dimensão da matriz jacobiana $\boldsymbol{J}^{(k)}$.

Dos passos do Algoritmo 4.1, sem dúvida o que demanda maior carga computacional é o 3, associado à solução do sistema linear envolvendo o cálculo dos

desvios dos estados, a partir dos *mismatches* e da matriz jacobiana. Em geral, a matriz jacobiana é esparsa, o que sugere o uso de eficientes técnicas de esparsidade para resolução do problema. Além disso, qualquer ganho computacional é benéfico, principalmente, quando o alvo é sistemas de grande porte.

Conforme apresentado nos capítulos anteriores, métodos iterativos para resolução de sistemas lineares constituem eficientes ferramentas, se esse subproblema para o *PFC* for adequadamente tratado. Então, nos próximos capítulos, avalia-se o desempenho de alguns métodos numéricos ao *PFC*, confrontando-os com a técnica tradicional direta, baseada em eliminação gaussiana.

## 4.3 SUBPROBLEMA LINEAR ASSOCIADO AO PROBLEMA DE FLUXO DE CARGA

A resolução de sistemas lineares por métodos iterativos, para adequado desempenho requer o uso de pré-condicionadores e de apropriado esquema de ordenamento, conforme discutido no Capítulo 3. Em função disso, propõe-se resolver o subproblema linear do passo 3 do Algoritmo 4.1 seguindo essas inclusões, considerando-se também o uso de pré-condicionadores, obtidos a partir de fatoração de uma matriz $\boldsymbol{T}$. Com base na metodologia de dominância proposta no Capítulo 2, propõe-se uma estrutura de $\boldsymbol{T}$ baseada na estrutura de $\boldsymbol{J}^{(0)}$ (jacobiana da iteração inicial).

Assume-se que, antes de efetuar a fatoração *ILUTP* de $\boldsymbol{T}$, esta matriz seja reordenada, utilizando-se as mesmas matrizes $\boldsymbol{R}_L$ e $\boldsymbol{R}_R$, de transformação de linhas e colunas, que proporcionam reordenamento otimizado de $\boldsymbol{J}^{(0)}$, visando redução de *fill-in*. O processo de reordenamento consiste nas seguintes operações sobre o sistema linear (4.8), na iteração inicial:

$$\boldsymbol{R}_L\boldsymbol{J}^{(0)}\boldsymbol{R}_R\Delta\boldsymbol{y} = -\boldsymbol{R}_L\boldsymbol{b} \tag{4.21}$$

$$\hat{\boldsymbol{J}}^{(0)}\Delta\boldsymbol{y} = -\hat{\boldsymbol{b}} \quad \text{e} \quad \Delta\boldsymbol{x} = \boldsymbol{R}_R\Delta\boldsymbol{y} \tag{4.22}$$

em que $\hat{\boldsymbol{J}}^{(0)} = \boldsymbol{R}_L \boldsymbol{J}^{(0)} \boldsymbol{R}_R$ e $\hat{\boldsymbol{b}} = \boldsymbol{R}_L \boldsymbol{b}$.

Portanto, a matriz $\boldsymbol{T}$ reordenada transforma-se em:

$$\hat{\boldsymbol{T}} = \boldsymbol{R}_L \boldsymbol{T} \boldsymbol{R}_R \tag{4.23}$$

cujos fatores resultantes da fatoração $ILUTP$, $\hat{\boldsymbol{L}}$ e $\hat{\boldsymbol{U}}$, geram

$$\hat{\boldsymbol{T}} = \hat{\boldsymbol{L}} \hat{\boldsymbol{U}} \tag{4.24}$$

Como o objetivo é resolver o sistema linear reordenado em (4.21), efetua-se o pré-condicionamento desse sistema com os fatores $\hat{\boldsymbol{L}}$ e $\hat{\boldsymbol{U}}$ obtidos em (4.24), gerando o novo sistema linear na incógnita $\Delta \boldsymbol{z}$:

$$\hat{\boldsymbol{L}}^{-1} \hat{\boldsymbol{J}}^{(0)} \hat{\boldsymbol{U}}^{-1} \Delta \boldsymbol{z} = -\hat{\boldsymbol{L}}^{-1} \hat{\boldsymbol{b}} \tag{4.25}$$

$$\tilde{\boldsymbol{J}}^{(0)} \Delta \boldsymbol{z} = -\tilde{\boldsymbol{b}} \quad \text{e} \quad \Delta \boldsymbol{y} = \hat{\boldsymbol{U}}^{-1} \Delta \boldsymbol{z} \tag{4.26}$$

em que $\tilde{\boldsymbol{J}}^{(0)} = \hat{\boldsymbol{L}}^{-1} \hat{\boldsymbol{J}}^{(0)} \hat{\boldsymbol{U}}^{-1}$ é uma matriz reordenada e pré-condicionada e $\tilde{\boldsymbol{b}} = \hat{\boldsymbol{L}}^{-1} \hat{\boldsymbol{b}}$.

Note-se que $\Delta \boldsymbol{y}$ fica determinado por método iterativo, a partir da equação (4.26) e o desvio do vetor de estados $\Delta \boldsymbol{x}$ é obtido apenas reordenando os elementos de $\Delta \boldsymbol{y}$, conforme definido na equação (4.22).

As matrizes de transformação $\boldsymbol{R}_L$ e $\boldsymbol{R}_R$ são determinadas a partir das colunas de uma matriz identidade, conforme o método de reordenamento adotado [69, 70]. Nesta tese, adotaram-se os métodos de reordenamento *RCM* e *AMD*. Desta forma, para cada uma destas técnicas essas matrizes foram calculadas.

A partir da segunda iteração do algoritmo de Newton-Raphson, efetua-se procedimento similar ao realizado na equação (4.26), porém considerando nova matriz jacobiana, mas utilizando os *mesmos fatores* $\hat{\boldsymbol{L}}$ *e* $\hat{\boldsymbol{U}}$ *da iteração inicial*, os quais são

mantidos constantes durante todas as demais iterações. Este procedimento, evita carga computacional da fatoração $ILUTP$ sem causar prejuízo significativo ao processo iterativo no Algoritmo 4.1.

O Algoritmo 4.2 ilustra os principais passos na proposta desta tese para se usar um método iterativo na resolução do PFC.

**Algoritmo 4.2 –** Resolução do *PFC* via método iterativo de sistema linear

---

**Entrada:** estimativa inicial $\boldsymbol{x}^{(0)} = \left[\left[\boldsymbol{\theta}^{(0)}\right]^T \ \left[\boldsymbol{V}^{(0)}\right]^T\right]^T$, dados de geração (barras *PV*) e de carga (*PQ*), matriz $\boldsymbol{Y}_{bus}$, número máximo de iterações, $k_{max}$, tolerância para convergência $\varepsilon$

**Saída:** estados $\boldsymbol{V}$ e $\boldsymbol{\theta}$

---

1) Iniciar com $k = 0$ e calcular os *mismatches* $\Delta\boldsymbol{P}^{(0)}$ e $\Delta\boldsymbol{Q}^{(0)}$ através de (4.4) e (4.5); a matriz jacobiana $\boldsymbol{J}^{(0)}$, usando as expressões (4.13) a (4.20) e reordená-la, obtendo as matrizes de transformação $\boldsymbol{R}_L$ e $\boldsymbol{R}_R$, utilizando método apropriado, de acordo com (4.21); reordenar também $\Delta\boldsymbol{P}^{(0)}$ e $\Delta\boldsymbol{Q}^{(0)}$
2) Se $k = 0$, efetuar os cálculos dos fatores $\hat{\boldsymbol{L}}$ e $\hat{\boldsymbol{U}}$ do pré-condicionador por meio de (4.23) e (4.24) e ir para o passo 4; senão, continuar no passo 3
3) Calcular os elementos das submatrizes $\boldsymbol{A}_1^{(k)}, \boldsymbol{B}^{(k)}, \boldsymbol{C}^{(k)}$ e $\boldsymbol{D}^{(k)}$ da matriz jacobiana $\boldsymbol{J}^{(k)}$, reordená-la usando as matrizes de transformação $\boldsymbol{R}_L$ e $\boldsymbol{R}_R$ do passo 1
4) Resolver o sistema linear (4.26) por método iterativo para $\boldsymbol{y}$ e com este valor obter $\boldsymbol{x}$ usando $\boldsymbol{R}_R$ em (4.22), para determinar $\Delta\boldsymbol{\theta}^{(k)}$ e $\Delta\boldsymbol{V}^{(k)}$
5) Atualizar os valores dos estados $\boldsymbol{V}^{(k+1)} = \boldsymbol{V}^{(k)} + \Delta\boldsymbol{V}^{(k)}$ e $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}^{(k)}$
6) Fazer $k = k + 1$
7) Calcular os *mismatches* $\Delta\boldsymbol{P}^{(k)}$ *e* $\Delta\boldsymbol{Q}^{(k)}$ e $erro = \max\{||\Delta\boldsymbol{P}^{(k)}||_\infty, \ ||\Delta\boldsymbol{Q}^{(k)}||_\infty\}$
8) Se $erro < \varepsilon$, o processo convergiu: ir para o passo 10; senão, continuar no passo 9
9) Verificar se $k < k_{max}$: em caso positivo, efetuar o reordenamento em $\Delta\boldsymbol{P}^{(k)}$ e $\Delta\boldsymbol{Q}^{(k)}$ e retornar para o passo 3; senão, encerrar o processo, pois o processo não convergiu para o número de iterações e tolerância fixados
10) Os estados calculados são: $\boldsymbol{V} = \boldsymbol{V}^{(k)}$ e $\boldsymbol{\theta} = \boldsymbol{\theta}^{(k)}$, fim.

---

## 4.4 DETERMINAÇÃO DO PONTO DE MÁXIMO CARREGAMENTO

O efeito do carregamento (estressamento) do sistema pode ser avaliado por uma técnica que consiste no aumento do parâmetro da continuação $p$ nas equações (4.4) e (4.5), e a observação do comportamento das curvas P-V em barras de carga (gráfico da potência ativa *versus* magnitude de tensão) [71, 40]. Este procedimento requer o incremento de $p$ até um valor máximo, denominado *ponto de máximo carregamento* ou *ponto de colapso de tensão*. Os incrementos de $p$ ocorrem em passos discretos e não necessariamente iguais. Para cada valor deste parâmetro, corresponde à resolução de um *PFC*. Portanto, por esta técnica, é possível calcular o valor de $p$ tão próximo possível do seu valor máximo. Porém, paga-se um preço pelo mal condicionamento da matriz jacobiana à medida que $p$ tende ao seu valor máximo. Como consequência, o número de iterações de cada *PFC* aumenta, ficando bastante elevado próximo ao ponto de máximo carregamento.

Exatamente no ponto de colapso de tensão, a matriz jacobiana do *PFC* torna-se singular. Para calcular a tensão e a potência neste ponto crítico, o problema é formulado de outra forma de modo a evitar a singularidade do sistema de equações. A técnica utilizada nesta situação é chamada método da continuação [72, 66]. Com o tipo de estudo baseado na determinação do ponto de máximo carregamento, é possível identificar os elos de transmissão e as cargas mais críticas do sistema. No entanto, esta abordagem está fora do escopo desta tese.

Qualquer que seja o processo de cálculo adotado para se determinar o ponto de máximo carregamento, *precisa-se resolver, iterativamente, problemas não-lineares*, como vários *PFC*s, quando a estratégia adotada é a por cálculo de curvas P-V. É o processo correspondente a aplicar o Algoritmo 4.2 repetidas vezes.

Próximo ao ponto de máximo carregamento, em que a matriz jacobiana fica mal condicionada, o problema não-linear tende a divergir ou a requerer elevado número de iterações em comparação ao caso base ($p = 0$). Portanto, os métodos iterativos lineares

neste tipo de aplicação são aplicáveis também, tornando-se vantajosos. Principalmente, se o cálculo de pré-condicionadores for verificado apenas na iteração inicial, como proposto para os casos bases de fluxo de carga.

Portanto, investigar a eficiência dos métodos iterativos frente ao método direto para este tipo de problema reforça os objetivos da metodologia proposta neste trabalho.

No Capítulo 7, são apresentados experimentos e resultados que ilustram todos os aspectos teóricos discutidos neste capítulo. Porém, antes, no Capítulo 6, são apresentados experimentos e resultados mais gerais visando se determinar a melhor opção de pré-condicionamento e de método iterativo.

# Capítulo 5

# Sistemas-teste

## 5.1 INTRODUÇÃO

Nos capítulos subsequentes, são realizadas simulações com a finalidade de verificar o desempenho computacional da metodologia proposta neste trabalho. Todos os testes são dedicados à aplicação denominada na literatura como fluxo de carga em sistemas elétricos de potência. É uma aplicação que envolve a resolução iterativa de equações não-lineares. Consequentemente, requer a resolução de subproblema que caracteriza um sistema linear específico. Com a finalidade de prover dados e informações para que se realizem os experimentos nesta tese, neste capítulo, são apresentados detalhes dos sistemas-teste que são alvo de modelagem visando estudos de métodos iterativos para sistemas lineares.

O objetivo no uso dos sistemas-teste foi utilizar informações de *softwares* acessíveis e que admitissem agregações a eles de novas metodologias. Os algoritmos implementados neste trabalho, ou foram resultados de propostas de novos *scripts* ou foram objeto de uso de rotinas já existentes. Neste sentido, utilizou-se o aplicativo MATPOWER [73, 11], escrito em MATLAB [74]. Uma série de dados relativos aos sistemas-teste integram o conjunto de arquivos do pacote computacional. Alguns deles amplamente estudados e outros introduzidos mais recentemente, conforme atualizações disponibilizadas pelos autores do aplicativo. Portanto, com este intuito, priorizou-se o uso dos sistemas-teste com dados já implementados seguindo o padrão desse *software*.

O MATPOWER é descrito em [73], podendo ser acessado em [11]. Atualmente, o aplicativo é utilizado em pesquisas e estudos educacionais para simulações de fluxo de potência *CA* e fluxo de potência ótimo [75]. Consiste de um conjunto de arquivos-M, com *scripts* em MATLAB. Seu desenvolvimento foi decorrente das exigências computacionais da plataforma PowerWeb [75] relatadas em [11]. O *software* foi disponibilizado em 1997, via Internet, como um código aberto de um pacote de

simulação, agora distribuído sob Licença Pública Geral do GNU denominada GNU GLP [76] em [11]. Por sua característica de *software* livre é compatível com outras plataformas, como Octave [11].

Considerando o padrão de entrada de dados no formato do MATPOWER, foram estudados quatro sistemas-teste, conforme resumo apresentado na Tabela 5.1.

## 5.2 SÍNTESE DAS CARACTERÍSTICAS DOS SISTEMAS-TESTE

A Tabela 5.1 representa uma síntese dos sistemas-teste, constando o número de barras de carga, de geração e dimensão da matriz jacobiana gerada durante a formulação do problema de fluxo de carga. Os sistemas-teste serão aqui designados como: IEEE 118 barras, IEEE 300 barras, MATPOWER 3375 barras e Brasil Sul-Sudeste.

**Tabela 5.1** – Dados dos sistemas-teste

| mnemônico | sistema elétrico | número de barras | | ordem da matriz |
|---|---|---|---|---|
| | | carga | geração | jacobiana (*n*) |
| `case118IEEE` | IEEE 118 barras | 99 | 19 | 236 |
| `case300IEEE` | IEEE 300 barras | 238 | 62 | 600 |
| `case3375wp` | MATPOWER 3375 barras | 2982 | 392 | 6355 |
| `case_sis50g` | Brasil Sul-Sudeste | 319 | 99 | 1132 |

### 5.2.1 Sistema IEEE 118 barras [77]

O sistema IEEE 118 barras tem o seu diagrama unifilar ilustrado na Figura 5.1. Constitui um equivalente do sistema elétrico de potência americano (Midwestern US) relativo à rede de 1962, amplamente utilizado em testes envolvendo análise de fluxo de carga [78, 79, 80]. Este sistema contém 19 geradores, 35 compensadores síncronos, 117 linhas, 9 transformadores e 91 pontos de carga.

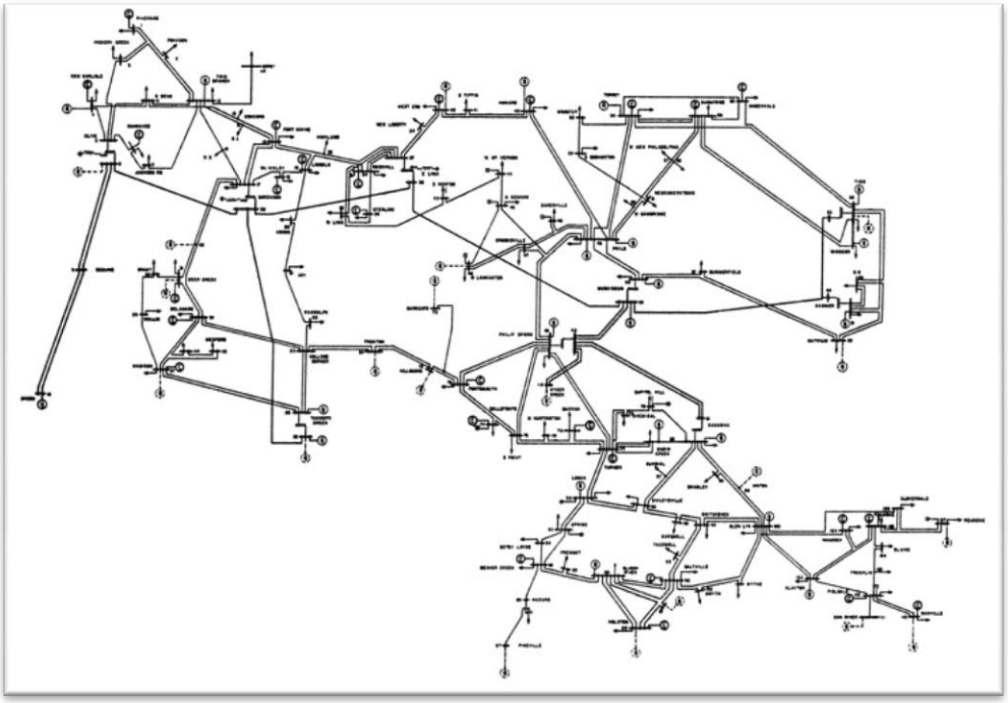


**Figura 5.1** – Diagrama unifilar do sistema IEEE 118 barras [81]

A matriz jacobiana relativa ao problema de fluxo de carga desse sistema tem dimensão 236. Os dados adicionais sobre o sistema podem ser obtidos de [77, 82, 65].

### 5.2.2 Sistema IEEE 300 barras [77]

O diagrama unifilar do sistema IEEE 300 barras é ilustrado na Figura 5.2 [42]. Este sistema consiste de três subsistemas assim constituídos: subsistema 1, com 26 geradores síncronos; subsistema 2, com 21 geradores síncronos e um elo *CC*; e, subsistema 3, com 15 geradores síncronos. No MATPOWER, uma vez que somente a rede *CA* pode ser implementada, em suas barras terminais *CA*, o elo *CC* é convertido em geração/carga equivalente.

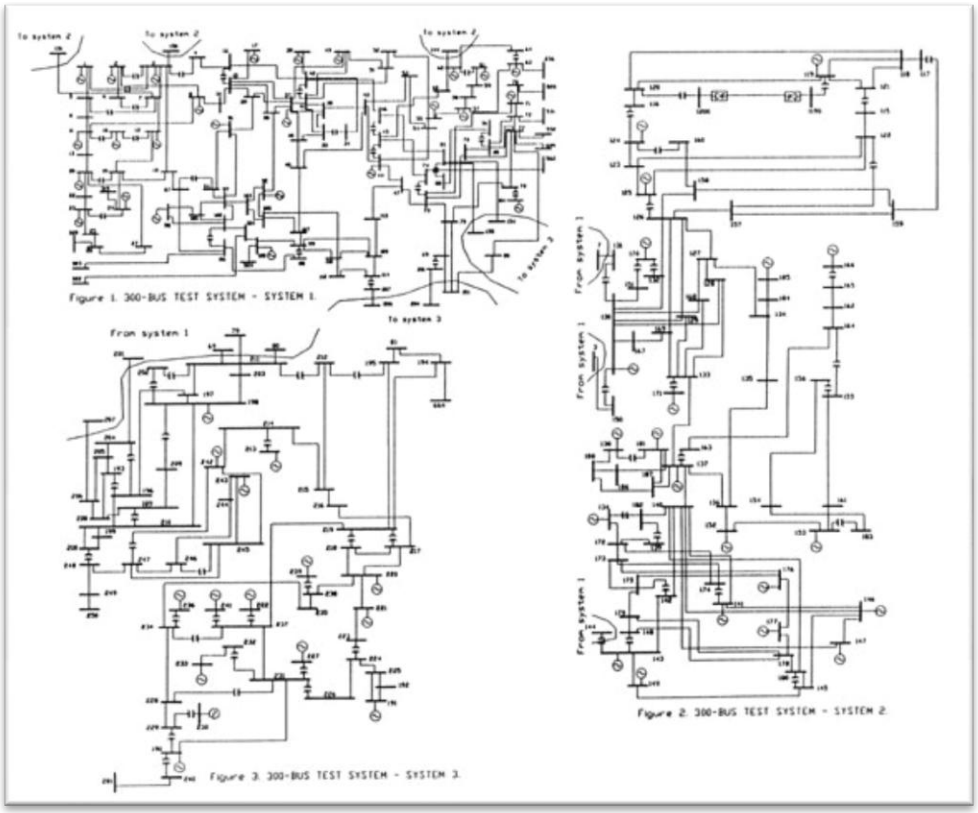


**Fig. 5.2** - Diagrama unifilar do sistema IEEE 300 barras [43]

Os detalhes de dados estáticos e dinâmicos do sistema IEEE 300 barras estão disponíveis em [77]. A matriz jacobiana calculada para este sistema exemplifica o caso de uma matriz indefinida, ou seja, com alguns autovalores tendo parte real negativa e outros com parte real positiva.

### **5.2.3** Sistema Brasil Sul-Sudeste

Este sistema constitui um equivalente do sistema elétrico brasileiro que congrega as regiões Sul, Sudeste e Centro-Oeste, acessível como parte integrante que acompanha o aplicativo comercial para análise dinâmica a pequenos sinais PacDyn [83]. Este aplicativo foi desenvolvido pelo Centro de Pesquisas de Energia Elétrica (CEPEL). A solução do problema de fluxo de carga no PacDyn é utilizada como ponto de operação para linearização de equações, e como resultado, geração de representação de sistema dinâmico na forma, ou de sistema descritor ou de estados. No entanto, o aplicativo

específico, também desenvolvido pelo CEPEL, para análise estática de redes elétricas é o ANAREDE [84].

Os dados são disponibilizados em arquivo padrão do ANAREDE e referenciado como dados do sistema equivalente brasileiro, com 50 geradores. Estas informações foram, então, convertidas neste trabalho para atender ao formato padrão do MATPOWER.

Trata-se de dados de rede relativos à carga pesada do ano de 1987. Tendo em vista que o MATPOWER não contempla a inclusão de elos *CC*, o sistema em 50 Hz associado à usina de Itaipu no sistema Brasil Sul-Sudeste foi considerado como uma carga equivalente conectada ao sistema *CA*, em 60 Hz.

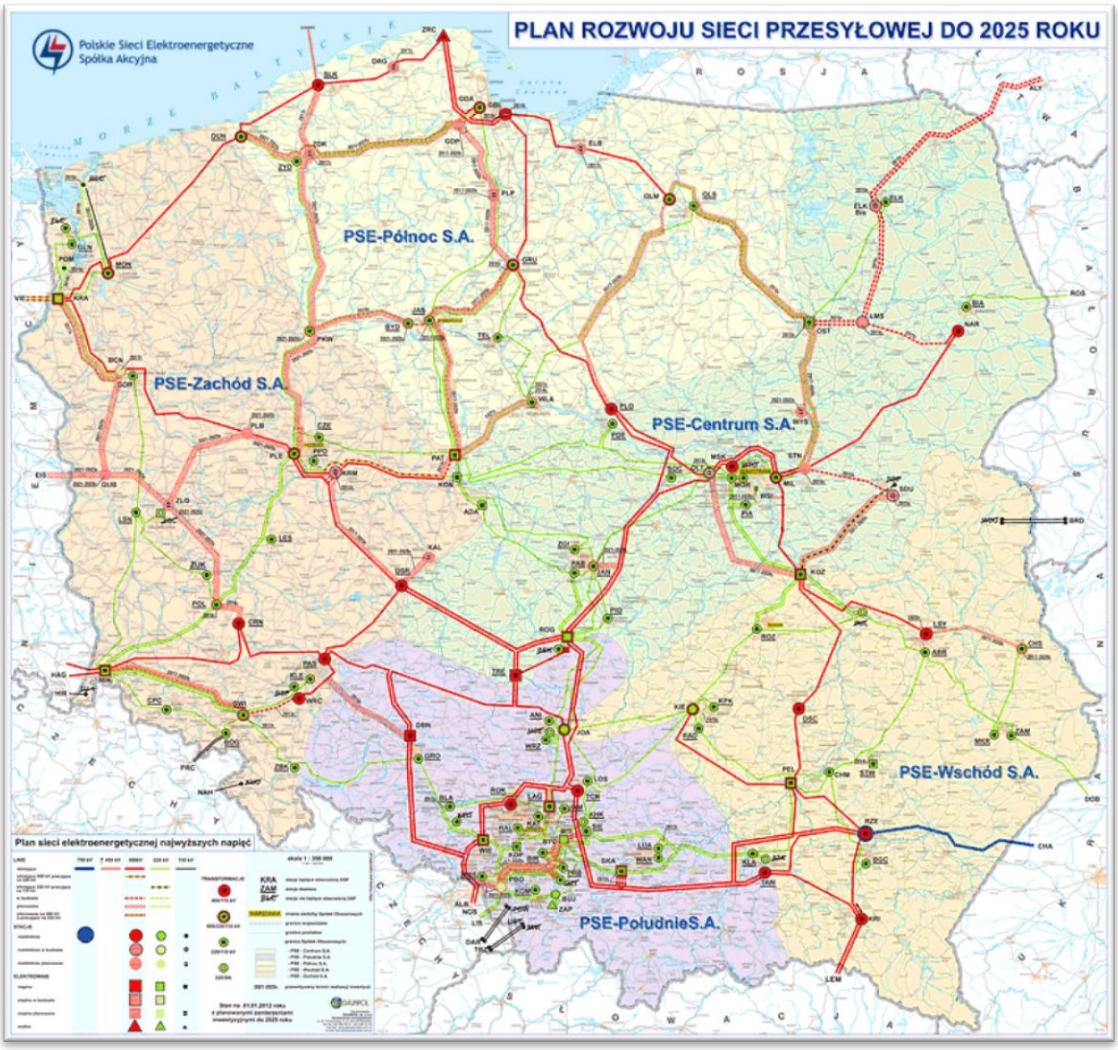
O sistema elétrico Brasil Sul-Sudeste tem 319 barras de carga, 99 barras de geração, proporcionando uma matriz jacobiana com 1132 elementos. Deve-se enfatizar que a realidade atual das dimensões do sistema interligado brasileiro contempla a conexão de todas as regiões do país [85]. Portanto, com um número de barras muito maior do que as verificadas no sistema-teste em apreço. Mas, o banco de dados associado, da mesma forma que o sistema anterior, precisaria ser convertido para o formato padrão MATPOWER. Porém, tendo em vista a restrição de modelagem do *software* adotado neste trabalho, a conversão do referido banco de dados, como o do Sistema Elétrico Brasileiro atual foge ao escopo deste trabalho. Sobretudo, devido ao quantitativo de informações que teriam de ser convertidas e porque são disponibilizadas em versões inacessíveis do aplicativo ANAREDE.

### 5.2.4 Sistema MATPOWER 3375 barras

Trata-se de sistema-teste, cujos dados acompanham as informações disponíveis do MATPOWER [73]. Este sistema tem 3375 barras. Mas, nas simulações desta tese, a barra #10287 foi removida dos dados originais, porque a mesma é isolada do sistema, não tendo qualquer significado físico. A sua presença no conjunto de dados gera elementos nulos na matriz jacobiana que a torna bastante mal condicionada para uso de métodos iterativos lineares. Logo, nas simulações, foi considerado sistema com 2982

barras de carga e 392 de geração. A jacobiana do sistema tem ordem 6355. Fisicamente, o sistema é uma representação do sistema elétrico da Polônia, com carregamento base relativo ao pico de carga do inverno de 2007-2008, e que inclui equivalentes de interconexão com Alemanha, República Checa e Eslováquia.

A Figura 5.3 apresenta um mapa físico da região, ilustrando interconexões do sistema elétrico [86]. Os níveis de tensão nominal do sistema variam de acordo com as distâncias a serem cobertas pela transmissão de potência, sendo: 220 kV, 400 kV e 750 kV (extra-alta tensão) no caso de longas distâncias de transmissão; 110 kV (alta-tensão) no caso de distâncias de transmissão não superior a dezenas de km; 10 kV e 30 kV (média tensão) usadas para linhas de distribuição local. O sistema de transmissão à época era constituído de: 242 linhas, totalizando 13.396 km incluindo, 100 subestações de extra alta tensão (*EAT*) e uma conexão submarina de 450 kV entre Polônia e Suécia.




| *LEGENDA* | | | |
|---|---|---|---|
| | linha de 750kV (azul) | | linha de 220kV (verde) |
| | linha de ± 450kV (laranja) | | linha de 110kV (preta) |
| | linha de 400kV (vermelha) | | |

**Figura 5.3** – Mapa físico com esquema da rede elétrica da Polônia de 2007-2008 [86]

Por ser o sistema que propicia maiores dimensões de matrizes e que tem bastante interconexões, formando um sistema com diversas malhas, o mesmo será alvo da maioria das simulações que se seguem, tanto no Capítulo 6, como no Capítulo 7.

# Capítulo 6

# Estudo de Caso: Sistema Linear Associado ao Problema de Fluxo de Carga

## 6.1 INTRODUÇÃO

Neste capítulo, será realizado um estudo de caso para comprovar a eficácia da metodologia proposta considerando o uso de métodos iterativos lineares. O sistema de equações linear adotado, quando denominado sistema-teste, é o *resultante da primeira iteração do método de Newton-Raphson* aplicado ao problema de fluxo de carga para o sistema elétrico MATPOWER 3375 barras (ver Capítulo 5). O sistema em questão também será considerado no capitulo seguinte, em que o problema de fluxo de carga será resolvido completamente, via método de Newton-Raphson com técnicas de resolução de sistemas lineares por métodos: direto e iterativos. A escolha deste sistema como base de estudo se justifica pelo fato de ser um sistema interligado relativamente complexo e de ter dados acessíveis ao público, no formato Matlab, portanto, no formato do aplicativo utilizado nesta tese [74].

O problema de fluxo de carga foi escolhido como estudo de caso, pois além do interesse prático intrínseco, as matrizes jacobinas típicas desse tipo de problema costumam ter características desafiantes ao se lidar com métodos iterativos lineares. Como por exemplo, normalmente, as matrizes jacobianas são indefinidas, mal condicionadas e, muitas vezes, ficam próximas da singularidade, especialmente quando os sistemas elétricos são de maior porte e operam próximos ao carregamento máximo [22]. Estas características dificultam o processo iterativo de solução levando a uma convergência lenta, ou até mesmo à falha do processo iterativo, justificando a necessidade de se considerar técnicas numéricas que atenuem ou eliminem estas dificuldades [26, 18].

Lembrando que os métodos iterativos resultantes da abordagem geral proposta nos capítulos anteriores consistem em uma combinação adequada do procedimento

reordenamento+pré-condicionamento baseado em desacoplamento+método linear clássico. Além disso, que cada um dos subalgoritmos possui parâmetros de ajuste. Daí, existem várias combinações que precisam ser testadas para se determinar qual a melhor configuração final ao problema linear a ser resolvido. A seguir serão apresentados os passos para escolha das técnicas e ajustes dos parâmetros apropriados para o sistema linear resultante do problema de fluxo de carga do sistema MATPOWER 3375 barras.

## 6.2 CONFIGURAÇÕES PARA AS SIMULAÇÕES NUMÉRICAS

Todas as simulações dos capítulos 6 e 7 são realizadas no ambiente computacional que utiliza o *software* MATLAB versão 7.9.0.529 (R2009b) em um processador Intel (R) Core™ 2Duo CPU T5450 @1.66GHz, com 2.00 GB de memória RAM e sistema operacional de 32 bits. A precisão relativa da máquina é de $2.2204 \times 10^{-16}$. Todas as soluções obtidas via método direto (eliminação gaussiana) e iterativos utilizam *scripts* do próprio MATLAB.

Tanto no solucionador proposto, como também para os demais avaliados, foram usados os mesmos arquivos de dados, cenários e pontos de operação.

## 6.3 DESACOPLAMENTO NA MATRIZ JACOBIANA

A matriz jacobiana $\boldsymbol{A}$ do sistema-teste obtido na primeira iteração do método de Newton-Raphson aplicado ao problema de fluxo de carga para o sistema-teste possui dimensão $n = 6355$ e $n_{nz} = 40704$ (elementos não-nulos), onde as estruturas típicas ilustrando a presença de elementos não-nulos destas matrizes podem ser visualizadas na Figura 6.1.a.

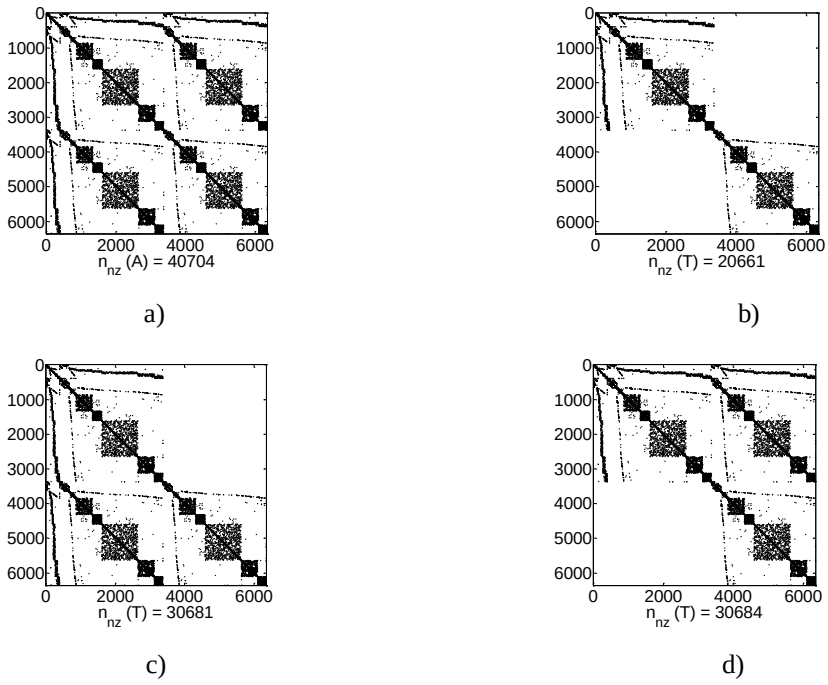



**Figura 6.1** – Estrutura com presença de elementos não-nulos de matrizes:
a) $\boldsymbol{A}$ b) $\boldsymbol{T}_1$ c) $\boldsymbol{T}_2$ d) $\boldsymbol{T}_3$

Conforme apresentado no capítulo 3, estuda-se estrutura de matrizes candidatas a pré-condicionadores. Além da estrutura modificada da própria matriz $\boldsymbol{A}$ via fatoração *ILUTP*, serão consideradas três versões da matriz jacobina truncada $\boldsymbol{T}_i,\ i = 1,2,3$. Para criar as matrizes de pré-condicionador $\boldsymbol{T}$, o primeiro passo consiste na divisão em submatrizes $\boldsymbol{A} = [\boldsymbol{A}_1\ \boldsymbol{B}\ ;\boldsymbol{C}\ \ \boldsymbol{D}]$, ou seja, é necessário determinar o tamanho das submatrizes $\boldsymbol{A}_1, \boldsymbol{B}, \boldsymbol{C}$ e $\boldsymbol{D}$. Para isto, utiliza-se a métrica de dominância que proporciona uma avaliação da coesão do acoplamento entre blocos da matriz jacobiana através das equações (3.7) a (3.19). Os resultados obtidos, quando da divisão em submatrizes, são ilustrados na Figura 6.2 onde os índices de dominância para $\boldsymbol{A}_1, \boldsymbol{B}, \boldsymbol{C}$ e $\boldsymbol{D}$ estão plotadas nos eixos de dominância de colunas (abscissa) e linhas (ordenada).

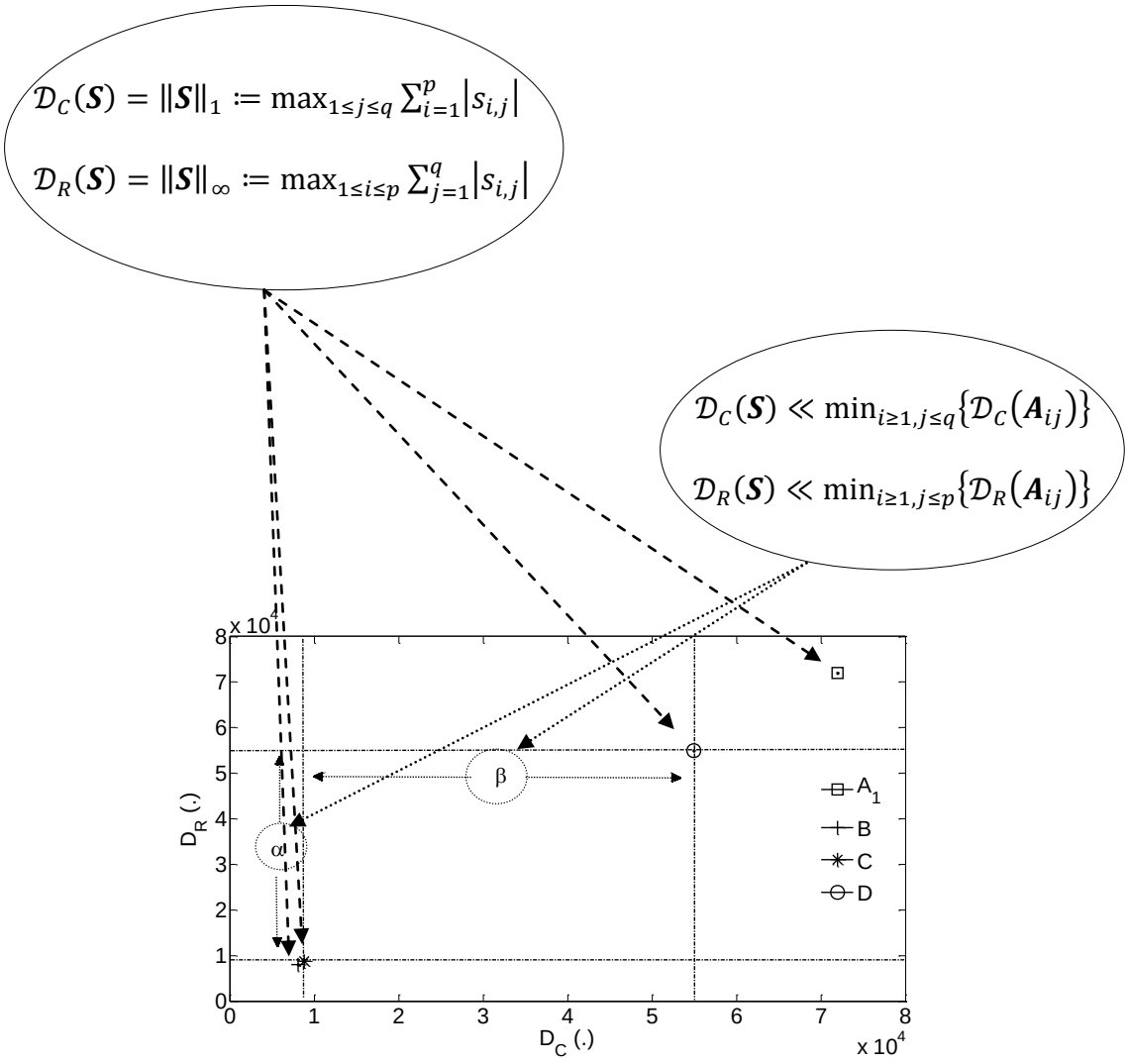



**Figura 6.2** – Valores dos índices de dominância nos eixos $\mathcal{D}_C(\cdot)$ e $\mathcal{D}_R(\cdot)$

Na Figura 6.2, são identificados dois fatores, $\alpha$ e $\beta$ que indicam as diferenças entre valores dos índices $\mathcal{D}_R(\cdot)$ e $\mathcal{D}_C(\cdot)$, respectivamente, e estão associados às submatrizes de valores de dominâncias mais próximas entre si. Tanto $\alpha$ quanto $\beta$ mostram que as diferenças entre os valores de $\mathcal{D}_R(\boldsymbol{D})$ e $\mathcal{D}_R(\boldsymbol{C})$ e entre $\mathcal{D}_C(\boldsymbol{D})$ e $\mathcal{D}_C(\boldsymbol{C})$ são de aproximadamente 5,5 vezes. Quanto maior este valor em relação à unidade, maior o potencial de desacoplamento. Se estas diferenças fossem computadas com relação à $\boldsymbol{A}_1$, os valores dos índices desta matriz seriam de cerca de sete vezes os índices de $\boldsymbol{C}$.

Com estes resultados, conclui-se que existe dominância dos blocos diagonais $\boldsymbol{A_1}$ e $\boldsymbol{D}$ em relação aos blocos fora da diagonal principal $\boldsymbol{B}$ e $\boldsymbol{C}$.

Como cuidado extra a ser tomado no momento da partição da matriz $\boldsymbol{A}$, deve-se avaliar a invertibilidade das matrizes $\boldsymbol{T_i}$. Para se utilizar $\boldsymbol{T_1}$ como pré-condicionador, requer-se que $\boldsymbol{A_1}$ e $\boldsymbol{D}$ sejam matrizes não-singulares. Se a opção for por usar $\boldsymbol{T_2}$ ou $\boldsymbol{T_3}$, $\boldsymbol{A_1}$ ou $\boldsymbol{D}$ deverá ser não-singular, respectivamente. Qualquer que seja o caso, a matriz de pré-condicionamento resultará mais esparsa que $\boldsymbol{A}$.

Na Tabela 6.1, observa-se que os valores percentuais de reduções de elementos não-nulos $(n_{nz})$ ilustrados na Figura 6.1 são significativos. As reduções de $n_{nz}$ para cada opção de $\boldsymbol{T}$ foram calculadas com base na dimensão $n$ e $n_{nz}$ de $\boldsymbol{A}$. Na divisão em submatrizes, foi estabelecido para dimensão de $\boldsymbol{A_1}$ o valor de $n_1 = 3373$, obtendo em consequência $n_{nz}(\boldsymbol{A_1}) = 11499$, $n_{nz}(\boldsymbol{B}) = 10023$, $n_{nz}(\boldsymbol{C}) = 10020$ e $n_{nz}(\boldsymbol{D}) = 9162$.

**Tabela 6.1** – Reduções percentuais de $n_{nz}$ de $\boldsymbol{T_i}$ em relação à matriz $\boldsymbol{A}$

| *Matriz* | $n_{nz}$ | *Redução de* $n_{nz}$ |
|---|---|---|
| $\boldsymbol{T_1}$ | 20661 | 49.24% |
| $\boldsymbol{T_2}$ | 30681 | 24.63% |
| $\boldsymbol{T_3}$ | 30684 | 24.62% |

O $n_{nz}$ da matriz $\boldsymbol{T_1}$, neste exemplo, tem cerca de 50% dos elementos da matriz jacobiana original. Esta opção de desacoplamento para fins de pré-condicionamento foi adotada com sucesso em [1], onde os autores relatam experimentos para cálculos da solução do problema de fluxo de carga com sistemas com até cerca de um milhão de variáveis. Estes sistemas-teste de dimensões gigantes foram gerados artificialmente. Neste estudo, a exceção de experimentos mais pontuais realizados no Capítulo 7, optou-se por trabalhar com sistemas de porte modestos, se comparados aos de [1], mas que refletissem as interconexões efetivamente existentes em sistemas elétricos reais. Desta

forma, evitou-se priorizar qualidades de um sistema radial ou mais malhado, contribuindo para melhor desempenho de um ou outro pré-condicionador.

A proximidade dos índices de dominância das submatrizes $\boldsymbol{B}$ e $\boldsymbol{C}$, observada na Figura 6.2, indicariam, a princípio, a escolha de $\boldsymbol{T}_1$, como melhor alternativa para um pré-condicionador, justificado pelo fato de possuir menos elementos não-nulos e porque a sua matriz inversa pode ser obtida a partir dos blocos diagonais desacoplados $\boldsymbol{A}_1$ e $\boldsymbol{D}$. No entanto, somente pelos valores calculados de índices de dominância não é possível afirmar quanto à maior eficiência do desacoplamento completo das duas malhas. Mas, apenas que, considerar desacoplamento é possível. Esta razão justifica a busca por investigar as três estruturas desacopladas da Figura 6.1 e os respectivos números de condicionamento resultantes de $\boldsymbol{T}^{-1}\boldsymbol{A}$. Por isso, propõe-se neste estudo estender as investigações para as opções que representam o desacoplamento parcial, de acordo com as estruturas definidas por $\boldsymbol{T}_2$ e $\boldsymbol{T}_3$. Embora estas matrizes sejam construídas a um custo adicional de cerca de 50% nos elementos não-nulos em relação à $\boldsymbol{T}_1$, a proposta é mostrar que, apesar deste acréscimo, é possível construir um solucionador iterativo que conduza a um desempenho computacional superior àquele utilizando $\boldsymbol{T}_1$, bem como ser capaz de superar o método direto.

A seguir são investigados o efeito das técnicas de pré-condicionamento, fatoração e reordenamento sobre alguns métodos iterativos clássicos como GMRES, CGS, BiCG e BiCGStab.

## 6.4. EFEITO DO PRÉ-CONDICIONAMENTO À ESQUERDA DA MATRIZ *A*

Nesta seção, é feito um estudo dos números de condicionamento da matriz de coeficientes do sistema linear pré-condicionado e não pré-condicionado. O número de condicionamento das matrizes resultantes de processo de pré-condicionamento à esquerda de $\boldsymbol{A}$ para cada opção de truncamento $\boldsymbol{T}$ é denotado por $\kappa(\boldsymbol{T}_i^{-1}\boldsymbol{A})$.

Utilizou-se duas formas de cálculo distintas para o cálculo do valor do número de condicionamento. Uma delas usou recurso do programa Matlab através do código *condest* [87, 88]. Esta abordagem baseia-se no estimador de número de

condicionamento a partir da norma-1 da matriz, o qual é calculado sem a necessidade da inversão explícita da matriz. Na outra forma, clássica, baseia-se na definição de norma espectral, por exemplo, para uma matriz $\boldsymbol{A}$, onde o número de condição é igual a $\|\boldsymbol{A}\|_2\|\boldsymbol{A}^{-1}\|_2$ [15], o que é equivalente à relação $\sigma_{max}/\sigma_{min}$, sendo $\sigma_{max}$ e $\sigma_{min}$ o máximo e mínimo valor singular de $\boldsymbol{A}$, respectivamente. A Tabela 6.2 ilustra os valores dos números de condicionamento da matriz resultante de $\boldsymbol{A}$ na condição sem pré-condicionamento e quando se utiliza matrizes $\boldsymbol{T}$ (conforme estruturas em (3.19)) como pré-condicionadoras à esquerda.

**Tabela 6.2** – Número de condicionamento $\kappa(\cdot)$

| *Forma de cálculo* | $\kappa(\boldsymbol{A})$ | $\kappa(\boldsymbol{T}_1^{-1}\boldsymbol{A})$ | $\kappa(\boldsymbol{T}_2^{-1}\boldsymbol{A})$ | $\kappa(\boldsymbol{T}_3^{-1}\boldsymbol{A})$ |
|---|---|---|---|---|
| $condest$ | $4.94\times10^6$ | $1.09\times10^4$ | $1.11\times10^4$ | $1.45\times10^3$ |
| $\sigma_{max}/\sigma_{min}$ | $4.59\times10^3$ | 37.65 | 36.30 | 11.61 |

Na Tabela 6.2, os resultados obtidos pelas duas formas de cálculo apresentam diferenças significativas nos valores absolutos do número de condicionamento. A justificativa se deve ao fato dos cálculos dos números de condição seguirem metodologias distintas. Enquanto a determinação do indicador *condest* utiliza norma-1, o cálculo clássico baseia-se na relação dos valores extremos de valores singulares. No entanto, é possível proceder à análise de tendências de melhor condicionamento, fazendo-se uma comparação dos valores absolutos, a partir dos cálculos obtidos por um dos métodos.

Excluído, portanto, o valor sem pré-condicionamento, as opções de desacoplamento total $(\boldsymbol{T}_1)$ e parcial $(\boldsymbol{T}_2)$ apresentam os maiores valores de número de condição. Logo, com base nesta abordagem, $\boldsymbol{T}_3$ mesmo sendo menos esparsa que $\boldsymbol{T}_1$, mas por contribuir para estabilidade numérica de $\boldsymbol{A}$, candidata-se como *pré-condicionador mais adequado* a ser aplicado a métodos iterativos em problemas de fluxo de carga.

### 6.4.1 Efeitos do pré-condicionamento à esquerda nos métodos iterativos lineares

Nesta seção, verifica-se o efeito do pré-condicionamento no desempenho dos métodos iterativos lineares. A quantidade de iterações foi utilizada como um referencial para avaliação da carga computacional visando alcançar a solução do sistema linear. Na Tabela 6.3 são apresentados os números de iterações requeridos pelos métodos GMRES, BiCGStab, BiCG e CGS ao se utilizar o sistema pré-condicionado, apenas à esquerda, considerando as matrizes $\boldsymbol{T}_i$. Foi adotado como critério de parada para aceitação da solução do sistema linear o resíduo relativo $\|\boldsymbol{Ax} - \boldsymbol{b}\|/\|\boldsymbol{b}\|$. A tolerância para convergência foi fixada no valor $10^{-6}$ e admitidas até 42 iterações para cada método. No caso do método GMRES, estas iterações são interpretadas de forma diferente, pois se considera a versão do método com reinicialização. Neste caso, o número de reinicializações máximas (laço interno do método) adotado foi seis, sendo admitidas até sete iterações externas (laço externo), totalizando assim, também quarenta e duas iterações, como estabelecido para os demais métodos (BiCGStab, BiCG e CGS). A solução inicial $\boldsymbol{x}^{(o)} = 0$ foi assumida em todas as simulações. Na Tabela 6.3, para o método GMRES, convencionou-se a notação k/m para indicação do número de reinicializações (m) e do número de iterações externas (k). Quanto ao método BiCGStab, a iteração em número não inteiro é devida ao fato que neste método, conta-se por metade de iteração.

**Tabela 6.3** – Número de iterações e tempos médios de CPU(*s*) para solução do sistema-teste pré-condicionado à esquerda.

| *Método* | *Iterações* | | | *Tempo* | | |
|---|---|---|---|---|---|---|
| | $\boldsymbol{T}_1$ | $\boldsymbol{T}_2$ | $\boldsymbol{T}_3$ | $\boldsymbol{T}_1$ | $\boldsymbol{T}_2$ | $\boldsymbol{T}_3$ |
| GMRES | 2/5 | 1/6 | 1/6 | 0.6344 | 0.3449 | 0.3716 |
| BiCGStab | 7.5 | 3 | 3 | 0.5885 | 0.3087 | 0.3080 |
| BiCG | 12 | 6 | 7 | 0.9035 | 0.6242 | 0.7158 |
| CGS | 8 | 4 | 3 | 0.6052 | 0.4061 | 0.3022 |

Quando o sistema-teste foi simulado utilizando-se o método direto, o tempo médio de *CPU* obtido foi de 0.0533s que em termos comparativos é muito inferior aos tempos obtidos segundo os resultados da Tabela 6.3. Conclui-se que, os métodos iterativos com o tipo de pré-condicionamento adotado não foram competitivos suficiente para superar o método de solução direta, mesmo nos casos em que uma solução satisfatória tenha sido obtida com poucas iterações. Resultados similares foram também obtidos, quando se utilizou as matrizes pré-condicionadoras à direita.

Diante dos resultados com pré-condicionadores somente pré-multiplicando (pós-multiplicando) a matriz $\boldsymbol{A}$, outras estratégias foram investigadas. Uma delas foi a que contempla também o reordenamento da matriz $\boldsymbol{A}$**.**

## 6.5 REORDENAMENTO

Nesta seção, avalia-se os resultados dos experimentos numéricos dedicados a verificação do desempenho dos métodos em estudo, ao se usar conjuntamente técnicas de reordenamento e pré-condicionamento. Para tal, verifica-se a eficiência computacional e a robustez considerando-se quatro configurações de pré-condicionador derivados de $\boldsymbol{A}, \boldsymbol{T}_1, \boldsymbol{T}_2$ e $\boldsymbol{T}_3$. No teste de convergência dos métodos, a tolerância ao erro foi fixada em $10^{-6}$.

### 6.5.1 Características principais dos reordenamentos *AMD* e *RCM*

Considerando-se que a influência da técnica de reordenamento para a resolução de sistema linear iterativo é favorável em termos de eficiência computacional, os métodos *RCM* e *AMD* serão avaliados. A Figura 6.3 ilustra a estrutura da matriz $\boldsymbol{A}$ onde $n = 6355$ e sua estrutura reordenada por *RCM* e *AMD*.

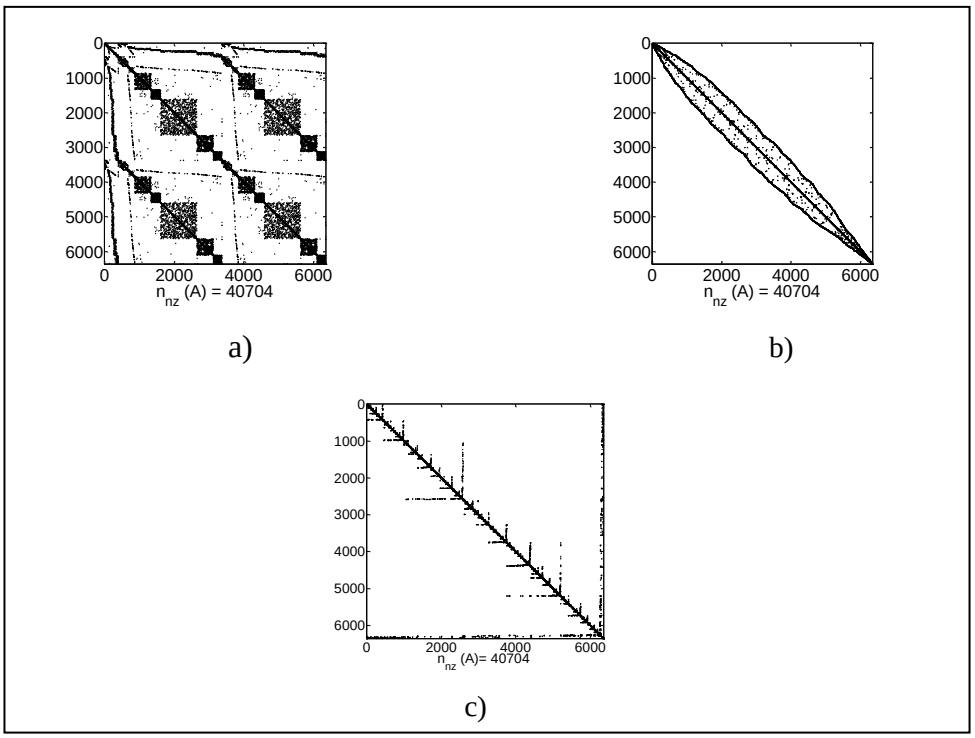



**Figura 6.3** – Exemplo de estruturas da matriz:
a) $\boldsymbol{A}$ b) $\boldsymbol{A}$ após reordenamento *RCM* c) $\boldsymbol{A}$ após reordenamento *AMD*

Na Figura 6.3, observam-se efeitos distintos dos reordenadores. Em todos os casos, o número de elementos não-nulos é o mesmo da matriz original.

### 6.5.2 Efeitos do reordenamento e pré-condicionamento incompleto

6.5.2.1 Análise pelo número de condicionamento

A partir de simulações realizadas sob duas condições: sem e com aplicação dos reordenamentos (*RCM* e *AMD*), efetuam-se os cálculos do número de condicionamento da matriz do sistema pré-condicionado. Na Tabela 6.4, os resultados são apresentados considerando pré-condicionamento $ILUTP$. Quatro opções de pré-condicionadores foram testados. Os dados contidos na coluna de $ILU(\boldsymbol{A})$ desta tabela referem-se ao número de condicionamento no qual foram usados os fatores $\boldsymbol{L}$ e $\boldsymbol{U}$ resultantes da fatoração $ILUTP$ da matriz de coeficientes, ambos aplicados como pré-condicionador à esquerda/direita de

$\boldsymbol{A}$, como definido em (3.3). Aqueles contidos em $ILU(\boldsymbol{T}_1), ILU(\boldsymbol{T}_2), ILU(\boldsymbol{T}_3)$, correspondem aos números de condicionamento obtidos, mas quando são adotados fatores $ILU$ da decomposição baseada em (3.21). Na linha correspondente a opção sem reordenamento, os pré-condicionadores são obtidos diretamente de $\boldsymbol{A}$, $\boldsymbol{T}_1, \boldsymbol{T}_2$ e $\boldsymbol{T}_3$. Nas demais linhas, quando há reordenamento, este processo é realizado com base na modificação de linhas e colunas de $\boldsymbol{A}$ de acordo com (3.21). As matrizes de transformação $\boldsymbol{R}_L$ e $\boldsymbol{R}_R$ obtidas deste reordenamento são então também aplicadas a $\boldsymbol{T}_1, \boldsymbol{T}_2$ e $\boldsymbol{T}_3$.

**Tabela 6.4** – Número de condicionamento de $\boldsymbol{A}$ com e sem pré-condicionador, considerando os fatores $\boldsymbol{L}$ e $\boldsymbol{U}$ obtidos a partir de decomposição $ILUTP$ e o reordenamento de $\boldsymbol{A}$

| *Reord.* | *Nº condição* | $ILU(\boldsymbol{A})$ | $ILU(\boldsymbol{T}_1)$ | $ILU(\boldsymbol{T}_2)$ | $ILU(\boldsymbol{T}_3)$ |
|---|---|---|---|---|---|
| *Sem* | $condest$ | $2.52 \times 10^3$ | $8.92 \times 10^3$ | $2.53 \times 10^3$ | $8.82 \times 10^3$ |
| | $\sigma_{max}/\sigma_{min}$ | 48.60 | 196.19 | 50.70 | 196.47 |
| *RCM* | $condest$ | 597.16 | $7.64 \times 10^4$ | $8.85 \times 10^{10}$ | $2.98 \times 10^5$ |
| | $\sigma_{max}/\sigma_{min}$ | 10.89 | $1.12 \times 10^4$ | $1.09 \times 10^{11}$ | $1.22 \times 10^5$ |
| *AMD* | $condest$ | 56.94 | 71.31 | 67.95 | 65.82 |
| | $\sigma_{max}/\sigma_{min}$ | 3.56 | 2.58 | 5.76 | 3.24 |

A exemplo das análises anteriores, sem a fatoração *ILU*, o cálculo do número de condicionamento baseado na norma-1 (cálculo de *condest*) foi o que resultou em maiores valores absolutos. Os cálculos dos números de condição a partir da relação dos valores singulares máximo e mínimo considerando as opções $ILU(T_i)$ e reordenamento *RCM* indicaram valores extremamente elevados, dando a entender a ineficácia dessa opção de pré-condicionador. Resultado mesmo pior em comparação ao caso sem pré-condicionador e sem reordenamento. No entanto, as opções com reordenamento *AMD* indicaram baixos valores de número de condição, tanto quando o cálculo ocorre com base em norma-1 quanto na forma clássica, com valores singulares. Diante desses resultados, ao se usar os fatores *ILU*, constata-se que a forma de reordenamento *AMD* é a que apresenta forma mais estável numericamente. Esta constatação pode ser justificada

pelo fato do algoritmo para implementar a opção $AMD$ ser um conjunto de rotinas para ordenamento de matriz esparsa para fatoração $ILU$ com pivoteamento diagonal [69] confirmado. Nesta simulação, os *setups* de programação (ajustes para implementação dos métodos iterativos através dos *scripts* em Matlab) foram *udiag*=0; *thresh*=0 e *milu='col'*. Para maiores detalhes sobre $ILUTP$ ver [18].

6.5.2.2 Análise pelo número de elementos não-nulos

Sob a mesma linha investigativa, foram observadas as influências dos métodos de reordenamento, segundo o número de elementos não-nulos de $\boldsymbol{L}$ e $\boldsymbol{U}$ em cada processo de fatoração das matrizes $\boldsymbol{A}$, $\boldsymbol{T}_1$, $\boldsymbol{T}_2$ e $\boldsymbol{T}_3$ ilustrados na Figura 6.4 abaixo.

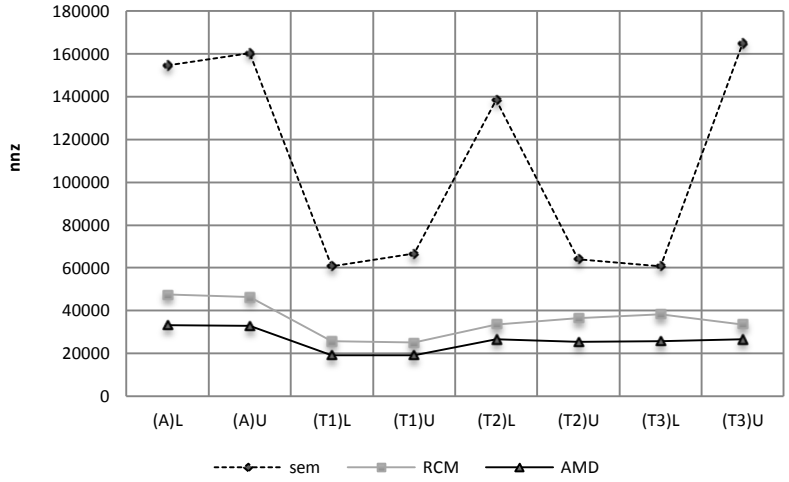



**Figura 6.4** - Números de elementos não-nulos dos fatores $ILU$

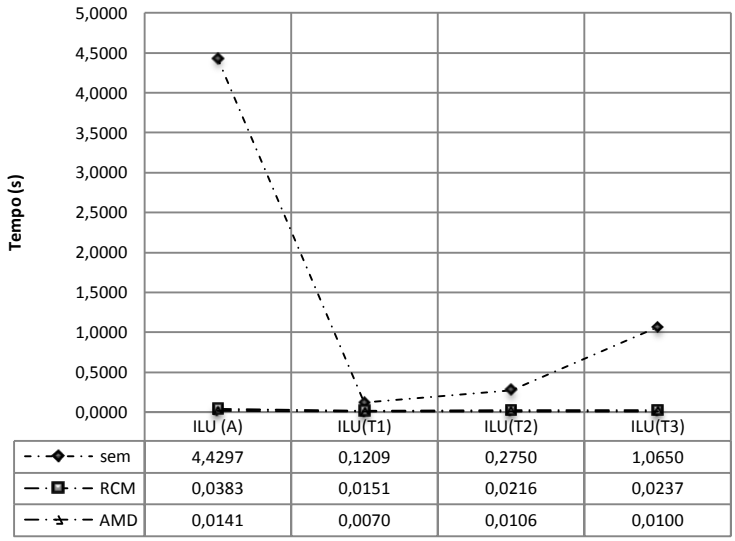
Permaneceu a influência positiva do reordenamento $AMD$ no processo, favorecendo a obtenção dos menores valores de $n_{nz}$ dentre os demais.

6.5.2.3 Análise pelo tempo médio de *CPU*

No procedimento de aferição do tempo de $CPU$, o parâmetro utilizado foi o tempo médio obtido para execução de um laço delimitado pelos códigos `tic` e `toc` do Matlab. Para esta finalidade, os cálculos no trecho de programa em que se aferiu o tempo de $CPU$ foram repetidos 500 vezes e o resultado, dividido pelo número de repetições efetivamente consideradas. Este procedimento atenua a influência de operações

realizadas pelo *hardware* durante a aferição do tempo de *CPU* e que não fazem parte do processo cujo desempenho computacional desejou-se aferir. As cinco primeiras repetições foram desprezadas devido à constatação de instabilidades de resultados de tempos de $CPU$ (valores elevados e inconsistentes) nos cálculos iniciais. Portanto, 495 foram efetivamente levadas em conta para o cálculo da média dos tempos de processamento. Verificou-se que nesses laços iniciais os tempos de *CPU*, principalmente, os dois primeiros, eram consideravelmente superiores à média. Embora tenha sido testada para um único laço, a opção `tcpu` recomendada no aplicativo Matlab para aferição do tempo de *CPU*, verificou-se desfavorável em comparação à estratégia adotada neste trabalho.

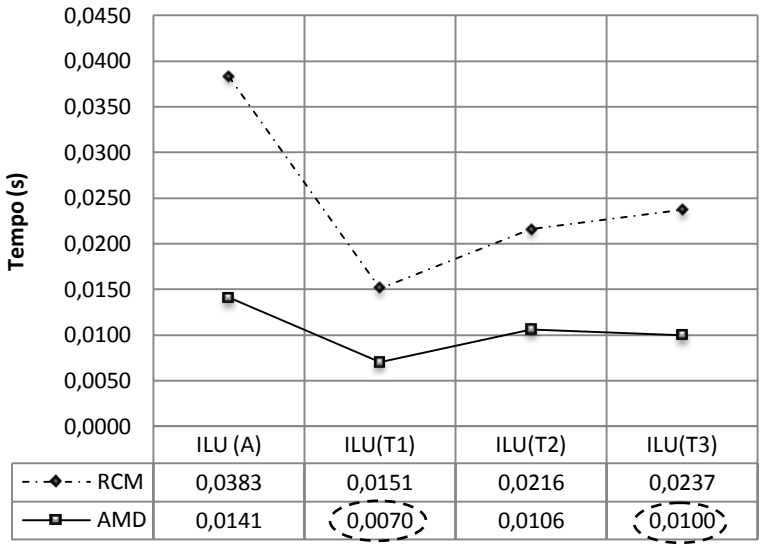
Os resultados dos tempos médios de $CPU$ obtidos para os cálculos dos fatores $ILU$ nas quatro opções avaliadas são ilustrados na Figura 6.5. Tendo em vista as diferenças significativas, na Figura 6.6, detalham-se somente os casos com reordenamento. Mediante intensivos testes e avaliação de resultados, concluiu-se pelo reordenamento de linhas e colunas de $\boldsymbol{A}$.




| | ILU (A) | ILU(T1) | ILU(T2) | ILU(T3) |
|---|---|---|---|---|
| sem | 4,4297 | 0,1209 | 0,2750 | 1,0650 |
| RCM | 0,0383 | 0,0151 | 0,0216 | 0,0237 |
| AMD | 0,0141 | 0,0070 | 0,0106 | 0,0100 |

**Figura 6.5** – Tempos médios de CPU para obtenção de fatores $ILU$ conforme opções de estruturas de $\boldsymbol{T}$

Quando o reordenamento não foi aplicado, vide Figura 6.4, o número de elementos não-nulos dos fatores $ILU$ foi elevado. Isto justifica os também elevados tempos médios de CPU sob a mesma condição, vide Figura 6.5. Por outro lado, o efeito contrário foi observado quando estes fatores foram calculados considerando os reordenamentos. Na Figura 6.6 é destacado o intervalo de 0 a 0,04 s da Figura 6.5, com seleção de interesse pelo comportamento dos dois tipos de reordenamento.




| | ILU (A) | ILU(T1) | ILU(T2) | ILU(T3) |
|---|---|---|---|---|
| RCM | 0,0383 | 0,0151 | 0,0216 | 0,0237 |
| AMD | 0,0141 | 0,0070 | 0,0106 | 0,0100 |

**Figura 6.6** – Tempos médios de *CPU* na obtenção de fatores $ILU$ mediante *AMD* e *RCM*

O melhor resultado observado ocorreu quando do desacoplamento total dos blocos $ILU(\boldsymbol{T}_1)$, o que justifica a menor média dos tempos de *CPU* na Figura 6.6 para a técnica *AMD*. Ainda neste tipo de reordenamento, ficou evidente que a maior quantidade de elementos não-nulos ocorreu para o caso sem desacoplamento de blocos $ILU(\boldsymbol{A})$. Os resultados indicam a influência imposta pelo reordenamento na matriz-teste na melhoria de desempenho de pré-condicionamento $ILUTP$ para os desacoplamentos $\boldsymbol{T}_1$ e $\boldsymbol{T}_3$.

## 6.6 DESEMPENHO DOS MÉTODOS LINEARES COM REORDENAMENTO E PRÉ-CONDICIONAMENTO

Para o procedimento avaliativo, efetuou-se o cálculo da solução do sistema linear considerando-se os pré-condicionadores obtidos a partir de fatores $ILU$, levando-se em conta ou não o ordenamento de linhas e colunas de $\boldsymbol{A}$. A Tabela 6.5 ilustra o número de iterações requerido por cada método (GMRES, BiCGStab, BiCG e CGS) e o tempo de

*CPU* para realização dos cálculos. No tempo de *CPU* não está incluída a carga computacional demandada para a fatoração *ILU* de cada método. Todos os cálculos desenvolvidos para decomposição *ILU* foram realizados com base na variante *ILUTP*, (*script ILU* do Matlab e ajustes requeridos) com *drop tolerance* ($\tau$) fixada em $6.5\times10^{-5}$, `setup.thresh=0, setup.udiag=0 e setup.milu='col'.`

**Tabela 6.5** – Iterações requeridas e tempo computacional em segundos

| *Reord.* | *Método* | *Iterações* | | | | *Tempo* | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\boldsymbol{A}$ | $\boldsymbol{T}_1$ | $\boldsymbol{T}_2$ | $\boldsymbol{T}_3$ | $\boldsymbol{A}$ | $\boldsymbol{T}_1$ | $\boldsymbol{T}_2$ | $\boldsymbol{T}_3$ |
| *Sem* | GMRES | 4/6 | 4/5 | 4/6 | 4/1 | 0.1619 | **0.0976** | 0.1097 | 0.1049 |
| | BiCGStab | - | 7.5 | 3 | 3 | 0.0650 | 0.0660 | **0.0540** | 0.0710 |
| | BiCG | - | 12 | 6 | 7 | 0.2439 | **0.1746** | 0.1863 | 0.2386 |
| | CGS | - | 8 | 4 | 3 | 0.0827 | 0.0685 | **0.0621** | 0.0796 |
| *RCM* | GMRES | 2/6 | 3/1 | 6/5 | 2/3 | 0.0431 | 0.0442 | 0.1176 | **0.0361** |
| | BiCGStab | 5.5 | 8 | 13 | 4.5 | 0.0256 | 0.0312 | 0.0572 | **0.0238** |
| | BiCG | 8 | 13 | 16 | 9 | **0.0553** | 0.0667 | 0.0948 | 0.0603 |
| | CGS | 6 | 8 | 12 | 5 | 0.0274 | 0.0306 | 0.0562 | **0.0231** |
| *AMD* | GMRES | 2/2 | 2/5 | 2/3 | 2/1 | 0.0283 | 0.0338 | 0.0278 | **0.0234** |
| | BiCGStab | 4 | 8.5 | 5 | 3.5 | 0.0170 | 0.0309 | 0.0197 | **0.0149** |
| | BiCG | 7 | 12 | 9 | 8 | **0.0396** | 0.0537 | 0.0427 | 0.0435 |
| | CGS | 5 | 9 | 7 | 4 | 0.0204 | 0.0318 | 0.0269 | **0.0171** |

Observa-se nas indicações em negrito da Tabela 6.5 os melhores tempos obtidos em relação ao método usado com ou sem reordenamento. O melhor resultado obtido com tempo de 0.0149s ocorreu para a condição $ILU(\boldsymbol{T}_3)$ ao se usar o método BiCGStab com reordenamento *AMD*, seguido nas mesmas condições pelo método CGS. Sempre que a estratégia de reordenamento foi utilizada, a exceção do BiCG, todos os demais resultados foram favoráveis à opção de reordenamento com desacoplamento $\boldsymbol{T}_3$.

Comparando os números de condição ilustrados na Tabela 6.4 e os desempenhos dos métodos com base no tempo de processamento e número de iterações para

convergência da Tabela 6.5, conclui-se que o fato de se ter número de condição muito elevado não é indicador adequado para caracterizar mal condicionamento de matriz.

Os resultados da Tabela 6.5 indicam que o reordenamento *AMD* é o mais indicado, propiciando cálculos finais mais rápidos, gerando menos *fill-in* com a fatoração completa *LU* e resultando na melhor qualidade de fatorações *ILU* no mesmo tempo. Comparando-se os pré-condicionadores à esquerda e os baseados em decomposição *ILU*, verificou-se que, para a primeira jacobiana do método de Newton-Raphson, a melhor técnica de pré-condicionamento é a baseada na fatoração *LU* incompleta (*ILU*) da matriz jacobiana, com a opção de pré-condicionador $\boldsymbol{T}_3$**.**

## 6.7 SINTONIA DOS PARÂMETROS DO MÉTODO *ILUTP*

Os experimentos das seções anteriores indicam que a combinação $AMD + ILUTP$ é a mais indicada para a matriz $\boldsymbol{A}$ resultante da primeira iteração do método de Newton-Raphson. Os resultados obtidos com o reordenamento $RCM$ com $ILUTP$ e $ILUTP$ sem reordenamento foram bastante piores. Nesta seção considera-se, então, uma sintonia mais fina dos parâmetros da fatoração $ILUTP$. Os parâmetros internos pré-definidos no *script ILU* (opção por fatoração $ILUTP$) para configuração do solucionador iterativo (método+*AMD*+$ILUTP$) e utilizados em todas as simulações daqui para frente estão contidos na Tabela 6.6.

**Tabela 6.6** - Parâmetros usados na fatoração $ILUTP$ e tolerância

| *Solucionador iterativo* | *Parâmetros* | |
|---|---|---|
| | tolerância para convergência | `setup` |
| método+*AMD*+$ILUTP$ | tol=$10^{-4}$ | `udiag = 0`<br>`thresh = 0`<br>`smilu = 'col'` |

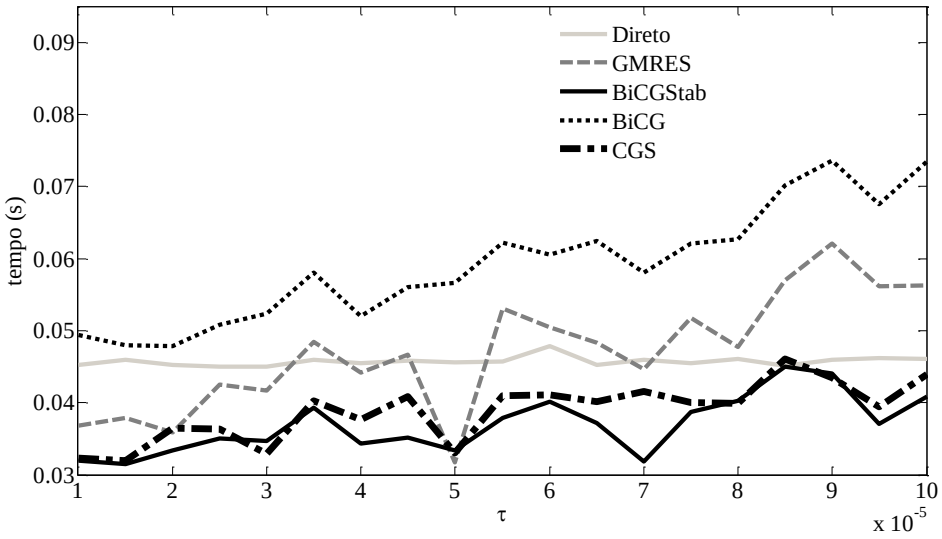
### 6.7.1 *Drop tolerance* $(\tau)$

No estudo preliminar da seção anterior, foi usado um valor fixo para o parâmetro *drop tolerance* $(\tau)$ do algoritmo $ILUTP(\tau)$ (este parâmetro influencia no preenchimento de elementos não-nulos do processo de fatoração $ILU$). Nesta seção, considerando o ordenamento fixo como sendo o $AMD$, são estudados quais valores ou faixa de valores onde o parâmetro *drop tolerance* $(\tau)$ do algoritmo $ILUTP(\tau)$ leva a melhores resultados para o pré-condicionamento da jacobiana associada ao primeiro passo do método de Newton-Raphson.

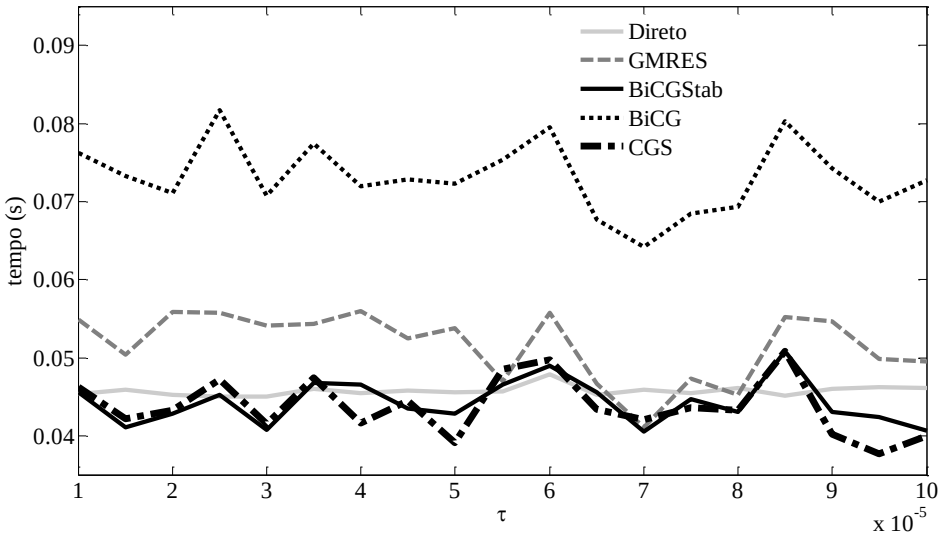
### 6.7.2 Sensibilidade da *drop tolerance* e das matrizes truncadas nos métodos lineares

Nesta subseção, avalia-se a robustez do desempenho de cada método frente a variações no valor da *drop tolerance* $\tau$. Para tal finalidade, nas simulações apresentadas a seguir, a faixa dos valores de $\tau$ segue a recomendação sugerida em [18], entre $10^{-5}$ a $10^{-4}$. Os parâmetros variáveis neste experimento são os fatores $ILU$ de diferentes matrizes, conforme proposto no Capítulo 3. Desta forma, calculam-se fatores $ILU$ de $\boldsymbol{A} = \hat{\boldsymbol{J}}_0$, e das matrizes truncadas $\boldsymbol{T}_1$, $\boldsymbol{T}_2$ e $\boldsymbol{T}_3$. Estas últimas três matrizes são obtidas da utilização das matrizes $\boldsymbol{A}_1$, $\boldsymbol{B}$, $\boldsymbol{C}$ e $\boldsymbol{D}$, reordenadas, conforme definido no Capítulo 5 e utilizando as estruturas propostas em (3.19).

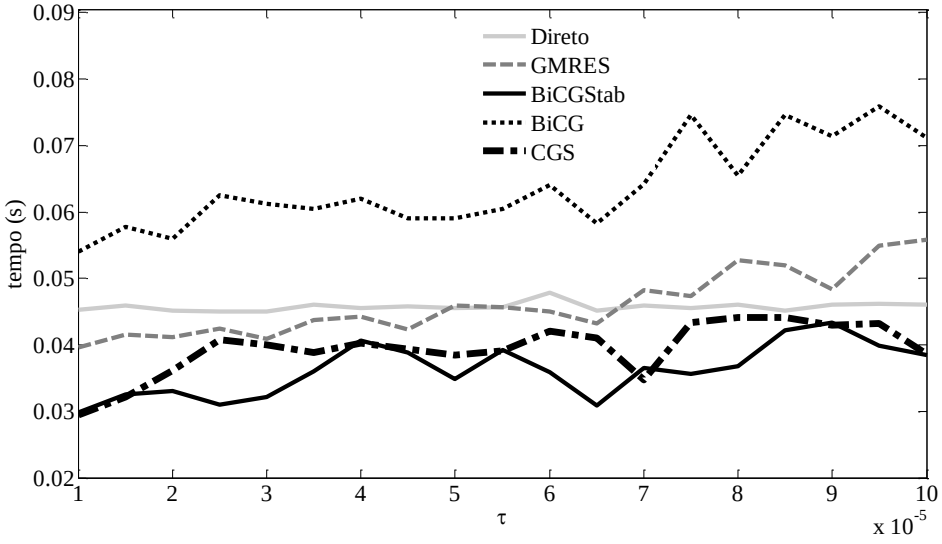
Também são avaliados desempenho de quatro métodos iterativos (GMRES, BiCG, BiCGStab e CGS) e do método direto. A avaliação de eficiência computacional (tempo de $CPU$) é ilustrada na Figura 6.7. Nos dados, já estão incluídas as parcelas de tempo médio de $CPU$ para fatoração $ILU$, bem como a parcela para a solução iterativa do sistema linear.



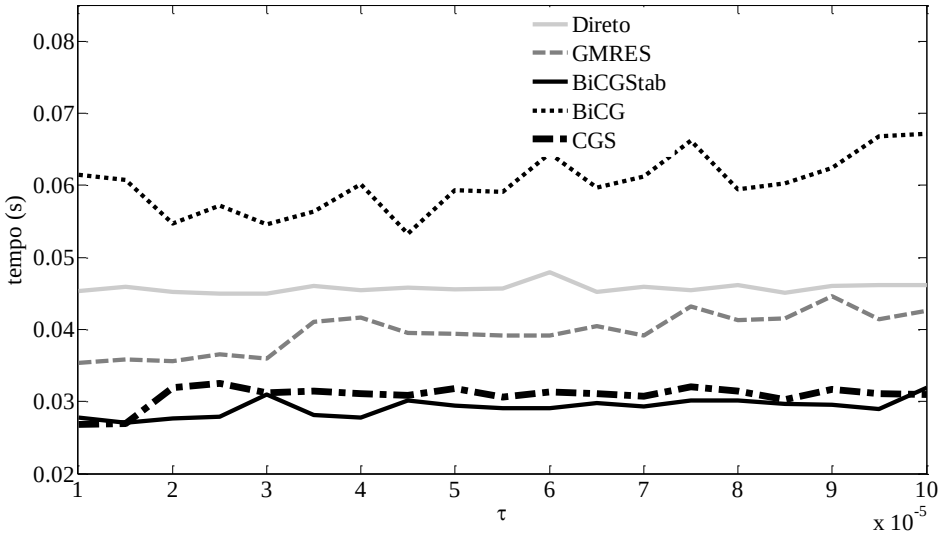

a) Tempo de *CPU* para cálculo dos estados × *drop tolerance*, ao se adotar fatoração $ILU(\boldsymbol{A})$




b) Tempo de *CPU* para cálculo dos estados × *drop tolerance*, ao se adotar fatoração $ILU(\boldsymbol{T_1})$




c) Tempo *CPU* para cálculo dos estados × *drop tolerance*, ao se adotar fatoração $ILU(\boldsymbol{T}_2)$




d) Tempo de *CPU* para cálculo dos estados × *drop tolerance*, ao se adotar fatoração $ILU(\boldsymbol{T}_3)$

**Figura 6.7** – Desempenho dos métodos lineares em função de $\tau$ com pré-condicionador baseado em fatores *ILU* de: a) $\boldsymbol{A}$; b) $\boldsymbol{T}_1$; c) $\boldsymbol{T}_2$; e d) $\boldsymbol{T}_3$

O resultado obtido para o método direto é aquele computado diretamente usando a opção $\boldsymbol{x} = \boldsymbol{A}\backslash\boldsymbol{b}$ no aplicativo Matlab, em que são utilizados $\boldsymbol{A}$ e $\boldsymbol{b}$ com opção de

reordenamento *AMD*. Note-se que na prática apesar de invariante com a *drop tolerance*, nos gráficos, os valores de $\boldsymbol{x}$ para o método direto variam. Este fato, novamente, reforça a situação que os valores computados são dependentes da ocupação do *hardware* no momento do cálculo. Justifica-se, deste modo, aferir o tempo de *CPU* com base em uma média de repetições do mesmo cálculo, conforme já mencionado anteriormente. A este respeito, após inúmeras simulações, verificou-se que as oscilações de valores a cada repetição são mais pronunciadas nos *scripts* do Matlab `gmres.m`, `bicg.m`, `bicgstab.m` e `cgs.m`, utilizados para calcular a solução de determinado método iterativo. Isto se verifica porque nas rotinas `*.m`, são executados vários comandos internamente, enquanto na opção $\boldsymbol{x} = \boldsymbol{A}\backslash\boldsymbol{b}$ há uma única linha de comando.

A visualização dos gráficos da Figura 6.7 permite concluir que dentre as opções de matrizes pré-condicionadoras aplicadas aos métodos testados, a matriz $\boldsymbol{T}_3$ via fatoração $ILU$ aplicada ao método BiCGStab, seguido pelo CGS propicia tempos de *CPU* mais competitivos que os demais para uma ampla faixa de valores de $\tau$. O método BiCG foi o que apresentou o pior desempenho em todas as simulações, inclusive pior que o método direto justificado pelo alto custo computacional para implementação de seu algoritmo. Ao longo das simulações o método GMRES, em média, apresentou sempre desempenho inferior aos métodos BiCGStab e CGS. Este resultado está em desacordo com a preferência de uso pelo método GMRES aplicado na resolução de problemas de fluxo de carga encontrado em vários trabalhos [7, 21, 1].

Comparando-se o desempenho do método direto com os iterativos, observa-se que na situação de fatoração $ILU(\boldsymbol{A})$, os tempos médios dos tempos de $CPU$ são comparáveis. Na situação de $ILU(\boldsymbol{T}_1)$, os resultados para os métodos iterativos são piores em comparação aos verificados, quando se usa $ILU(\boldsymbol{A})$. O desempenho com $ILU(\boldsymbol{T}_2)$ é superior ao obtido com $ILU(\boldsymbol{T}_1)$. Mas, de todas as avaliações, o melhor desempenho ocorreu ao se usar $ILU(\boldsymbol{T}_3)$.

Para uma análise mais apurada do desempenho do BiCGStab nos dados anteriores e dos fatores $ILU(\boldsymbol{T}_i), i = 1,2,3$, frente ao método direto, reuniram-se na

Figura 6.8 os registros dos resultados destes dois métodos na faixa estabelecida para $\tau$.

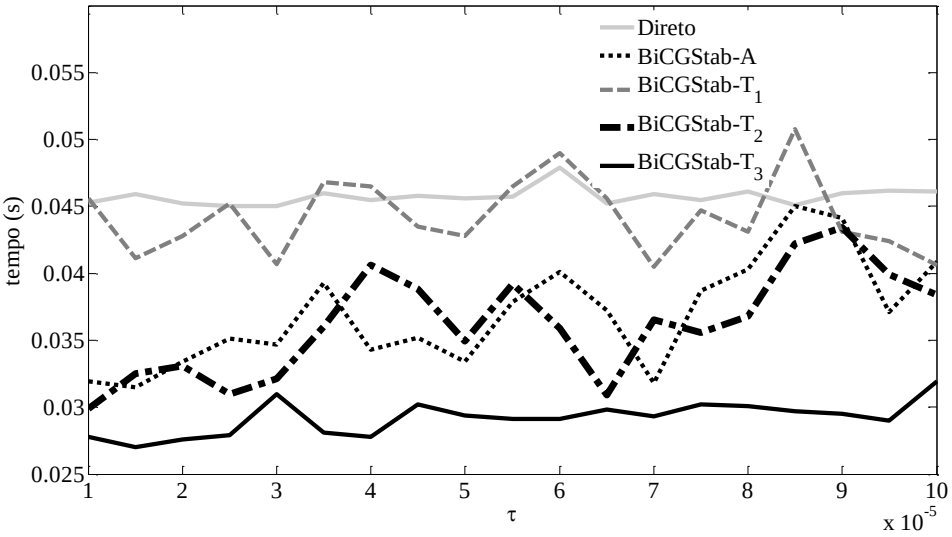



**Figura 6.8**– Desempenho dos métodos: direto e BiCGStab com opções de pré-condicionadores.

Pela Figura 6.8, para a mesma faixa de $\tau$, confirma-se que o desempenho do BiCGStab pré-condicionado por fatores obtidos de $ILU(\boldsymbol{T}_3)$, mostra-se um solucionador iterativo de melhor eficiência computacional. Na faixa de *drop tolerance* avaliada, o tempo médio de $CPU$ foi cerca de 50% inferior ao tempo requerido pelo método direto. A Figura 6.8 também evidencia um desempenho competitivo do BiCGStab frente ao método direto, mesmo quando o pré-condicionador se baseia na decomposição $ILU$ das outras matrizes $\boldsymbol{T}_2$ e $\boldsymbol{T}_3$. Até mesmo para a matriz jacobiana $\boldsymbol{A}$ completa. Vale destacar que nos métodos baseados em desacoplamento encontrados na literatura, em geral se usa a opção $\boldsymbol{T}_1$, ou seja, desacoplamento total (veja, por exemplo, [1]). Pela comparação das Figuras 6.7 e 6.8, fica evidente que o resultado para desacoplamento completo é menos favorável, se comparado ao resultado para o desacoplamento parcial dado por $\boldsymbol{T}_3$.

# Capítulo 7

# Simulações Numéricas para o Problema de Fluxo de Carga

## 7.1 INTRODUÇÃO

Este capítulo apresenta resultados de simulações numéricas para análise da eficiência do método de Newton-Raphson, quando aplicado ao problema de fluxo de carga em função do método linear utilizado a cada iteração. Enfatiza-se o $PFC$ aplicado a um caso base e a situações progressivas de carregamento até próximo do colapso de tensão (máximo carregamento). Nesta última aplicação, a matriz jacobiana fica próxima da condição de singularidade, requerendo mais iterações que o usual para um caso base.

Observe-se que o estudo realizado no Capítulo 6 fornece um indicativo de melhores algoritmos lineares e parâmetros de sintonia. Porém, como os estudos desse capítulo foram dedicados a apenas o subproblema linear da primeira iteração do método de Newton-Raphson, é necessário avaliar o que ocorre nas demais iterações, até a convergência final do processo não-linear. No entanto, um estudo completo de qual o melhor algoritmo linear e melhores parâmetros para todas as iterações é computacionalmente muito custoso e inviável na prática, principalmente quando a ordem do sistema é excessivamente grande. Assim, para simplificar, considerou-se a aplicação do mesmo método linear para cada um dos subproblemas lineares de todas as iterações do método de Newton Raphson, conforme ilustrado na Figura 4.1.

## 7.2 AJUSTES DE PARÂMETROS PARA A FATORAÇÃO

Uma vez que para a primeira jacobiana do método de Newton-Raphson a melhor técnica de fatoração da matriz jacobiana foi a fatoração $ILU$, com a opção de algoritmo $ILUTP$, considerar-se-á apenas esta técnica nos próximos cálculos.

Para cada iteração *k* do método iterativo de Newton-Raphson, foram utilizados os fatores *ILU* calculados na iteração inicial (*k* = 0, no Algoritmo 5.2). Consistente com os

resultados para o sistema linear do capítulo anterior, após várias simulações, confirmou-se que a técnica *AMD* é a mais indicada para os sistemas lineares de cada iteração do método de Newton-Raphson. Os resultados obtidos com o reordenamento *RCM* e sem reordenamento foram bastante piores. Por isso, somente resultados com pré-condicionamento e com ordenamento *AMD* são apresentados no cálculo dos estados do *PFC*.

Os parâmetros internos pré-definidos no *script ILUTP* (fatoração *ILU*) para configuração do solucionador iterativo (método+*AMD*+*ILUTP*) e utilizados em todas as simulações são os mesmos definidos na Tabela 6.6. Para efeito do cálculo de pré-condicionador, foi utilizada somente a estrutura de matriz $\boldsymbol{T}_3$.

## 7.3 FAIXA DA *DROP TOLERANCE* UTILIZADA NO MÉTODO NEWTON-RAPHSON

Visando escolher o melhor valor para a *drop tolerance* ($\tau$), foram realizados experimentos para solução do *PFC*. Com o objetivo de se verificar a tendência dos valores $\tau$ na resolução do problema, foi realizada uma amostragem de testes com oito realizações (oito repetições de mesmos testes) com o BiCGStab e com pré-condicionamento $ILU(\boldsymbol{T}_3)$ (BiCGStab+$ILU(\boldsymbol{T}_3)$). Todas as simulações foram feitas para a faixa estabelecida da *drop tolerance* ($\tau$), conforme ilustrado na Figura 7.1. Neste teste, foi considerado apenas o caso base relativo ao sistema MATPOWER 3375 barras. A convergência do algoritmo para cálculo dos estados ocorreu com quatro iterações, tanto usando o método direto quanto os iterativos.

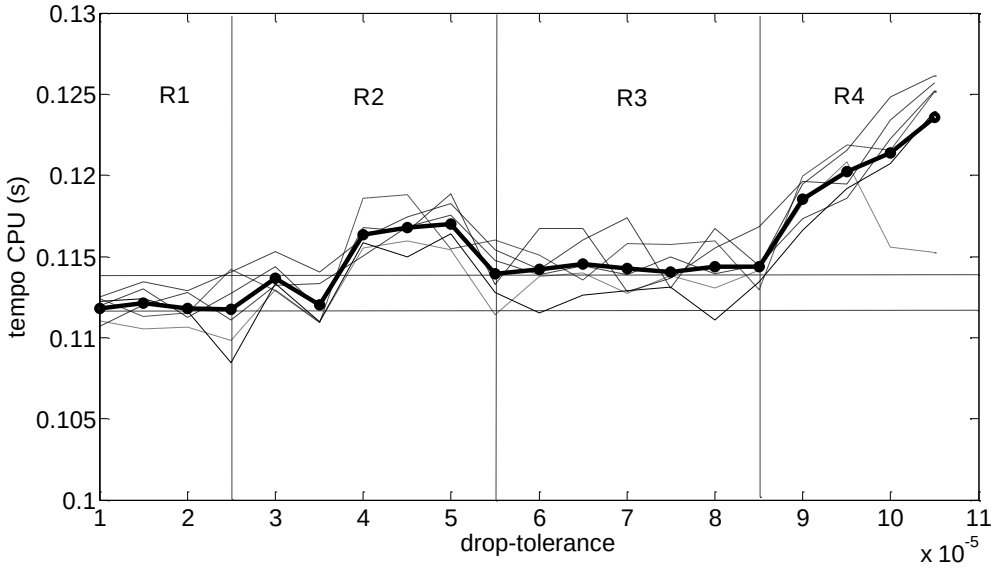


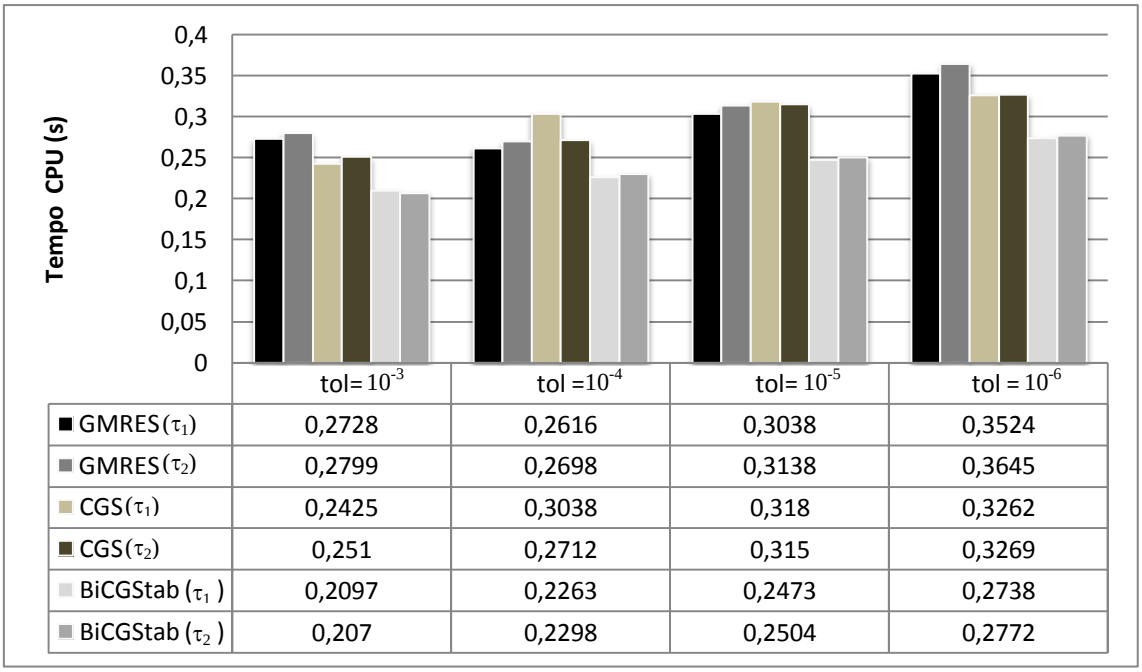

**Figura 7.1** – Desempenho em oito realizações do BiCGStab+$ILU(\boldsymbol{T}_3)$, em função de variações de $\tau$

A partir do gráfico da Figura 7.1, verifica-se que o comportamento do tempo médio de $CPU$ (linha negrito sólido), é possível distinguir quatro regiões identificadas por linhas tracejadas verticais. Destas, as identificadas como R1 e R3 são aquelas em que o comportamento do tempo médio de $CPU$ apresenta-se mais estável. Associando esta estabilidade às correspondentes faixas de valores de $\tau$, obtém-se, $(1 \text{ a } 2.5) \times 10^{-5}$ (R1) e $(5.5 \text{ a } 8.5) \times 10^{-5}$ (R3). Para a escolha de qual valor de $\tau$ é mais favorável em cada uma destas faixas, considera-se aquele valor que corresponder ao menor tempo médio de *CPU*. Assim, identificam-se os valores para $(\tau)$ de $2.5 \times 10^{-5}$ (R1) e $5.5 \times 10^{-5}$ (R3).

Investigou-se também o efeito do parâmetro *tolerância ao erro, tol*, dos métodos iterativos lineares. Uma vez que o valor da tolerância dos algoritmos lineares pode afetar o desempenho do algoritmo completo de Newton-Raphson, não há garantia de que o método BiCGStab continue como a melhor opção. Desta forma, realizou-se experimentos numéricos envolvendo mil repetições na solução do problema completo do fluxo de carga para os métodos iterativos GMRES, CGS e BiCGStab pré-condicionados com $ILU(\boldsymbol{T}_3)$, além do método direto. Na Figura 7.2, apresentam-se os tempos médios

de $CPU$ em função da tolerância, como também para os dois melhores valores de $\tau$ obtidos anteriormente, $\tau_1 = 2.5 \times 10^{-5}$ e $\tau_2 = 5.5 \times 10^{-5}$. As legendas GMRES $(\tau_1)$ e GMRES $(\tau_2)$ na figura significam que foram usadas, respectivamente, as *drop tolerances* $\tau_1$ e $\tau_2$ no processo de fatoração *ILU* para cálculo dos pré-condicionadores, então utilizados no método GMRES. Notação similar foi adotada na legenda para os demais métodos.

Para praticamente todos os valores de tolerância ao erro do método linear, confirmou-se que o BiCGStab+$ILU(\boldsymbol{T}_3)$ e $\tau = 2.5 \times 10^{-5}$ apresenta menor carga computacional.




| | tol= $10^{-3}$ | tol =$10^{-4}$ | tol= $10^{-5}$ | tol = $10^{-6}$ |
|---|---|---|---|---|
| GMRES($\tau_1$) | 0,2728 | 0,2616 | 0,3038 | 0,3524 |
| GMRES($\tau_2$) | 0,2799 | 0,2698 | 0,3138 | 0,3645 |
| CGS($\tau_1$) | 0,2425 | 0,3038 | 0,318 | 0,3262 |
| CGS($\tau_2$) | 0,251 | 0,2712 | 0,315 | 0,3269 |
| BiCGStab ($\tau_1$ ) | 0,2097 | 0,2263 | 0,2473 | 0,2738 |
| BiCGStab ($\tau_2$ ) | 0,207 | 0,2298 | 0,2504 | 0,2772 |

**Figura 7.2** – Tempo de *CPU* requerido pelo método de Newton-Raphson mediante variações de parâmeto dos métodos lineares: tolerâncias $(tol)$, *drop tolerance* $(\tau)$

Na Figura 7.2, verifica-se que o tempo de $CPU$ do método GMRES com $tol = 10^{-3}$ foi superior ao tempo requerido para $tol = 10^{-4}$, quebrando-se a tendência de quanto menor a tolerância, menor o tempo consumido. Para o método BiCGStab, com

$tol = 10^{-3}$, houve quebra no padrão de desempenho superior (tempo menor) para $\tau = 2.5 \times 10^{-5}$ em relação à $\tau = 5.5 \times 10^{-5}$.

Resumindo, nesta seção foram apresentados resultados de vários experimentos numéricos com o método de Newton-Raphson aplicado ao problema de fluxo de carga variando-se alguns dos seguintes parâmetros: método iterativo linear, tolerância ao erro do método linear, método de ordenamento, *drop tolerance* do método $ILUTP$, matriz jacobiana da primeira iteração truncada para condicionamento. Pode-se concluir que a fim de se ter robustez numérica e melhor desempenho, dentre os métodos iterativos lineares, a melhor configuração para ser utilizada no método de Newton-Raphson para o problema de fluxo de carga é o método BiCGStab. As melhores opções foram: erro de tolerância $tol = 10^{-4}$, pré-condicionamento baseado em $ILUTP$ com *drop tolerance* $\tau = 2.5 \times 10^{-5}$, matriz de pré-condicionador na forma $\boldsymbol{T}_3$. Neste tipo de problema, a combinação BiCGStab+$ILU(\boldsymbol{T}_3)$ se mostrou de melhor desempenho em comparação com o que se costuma recomendar na literatura, a saber, utilizar o método GMRES e nos trabalhos onde se considera truncamento da matriz do pré-condicionador, usar a matriz jacobiana com desacoplamento na forma $\boldsymbol{T}_1$ [1].

## 7.4 AVALIAÇÃO DO SOLUCIONADOR PROPOSTO PARA DIVERSOS SISTEMAS ELÉTRICOS DE POTÊNCIA

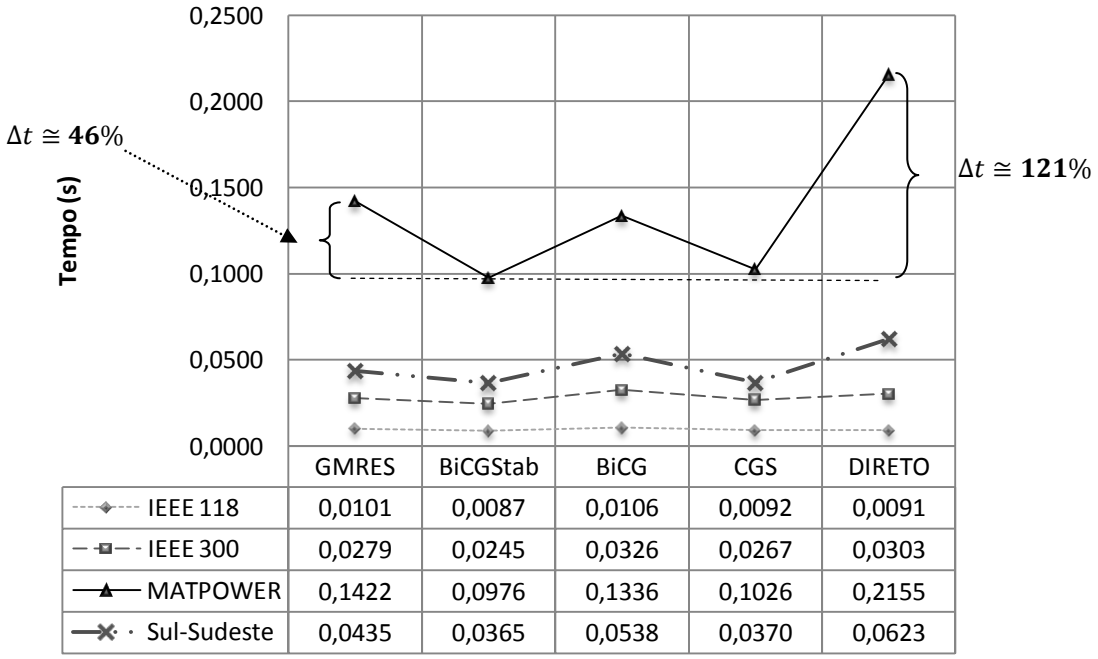
O estudo realizado nas seções anteriores indica que para solução do problema de fluxo de carga, o método proposto consistindo na combinação adequada de BiCGStab com pré-condicionador baseado na fatoração $LU$ incompleta $ILU(\boldsymbol{T}_3)$ e ordenamento $AMD$ é a opção mais atrativa. No entanto, este estudo foi realizado apenas para um sistema elétrico no seu caso base. Nas subseções a seguir, avalia-se a viabilidade e desempenho do método para aplicação em sistemas elétricos de diversas dimensões

### 7.4.1 Dependência do método de solução com o sistema elétrico

As características de conexão da rede altera a estrutura das equações do sistema (características de acoplamento entre subsistemas, esparsidade da matriz jacobiana, etc.). Visando determinar o método iterativo linear mais apropriado e melhor pré-

condicionador, a princípio, para cada sistema elétrico real para o qual se deseja resolver o problema de fluxo de carga, deve-se repetir todo o estudo realizado nas seções anteriores. A fim de avaliar se o resultado básico de que o método de Newton-Raphson com BiCGStab+$ILU(\boldsymbol{T}_3)$ também se verifica para outros sistemas reais, a seguir realizam-se experimentos onde são considerados, além do sistema MATPOWER 3375 barras, os seguintes sistemas elétricos: IEEE 118 barras, IEEE 300 barras, e o Brasil Sul-Sudeste (equivalente de sistema elétrico brasileiro).

O estudo considera a resolução de um caso base de *PFC* utilizando método iterativo linear. A exemplo das simulações anteriores, foi utilizada a opção de reordenamento *AMD*, pré-condicionador obtido a partir de $ILU(\boldsymbol{T}_3)$. Os parâmetros adotados foram: $tol = 10^{-4}$, e *drop tolerance* $\tau = 2.5 \times 10^{-5}$. Os resultados deste experimento em termos de tempo de $CPU$ são ilustrados na Figura 7.3.




| | GMRES | BiCGStab | BiCG | CGS | DIRETO |
|---|---|---|---|---|---|
| IEEE 118 | 0,0101 | 0,0087 | 0,0106 | 0,0092 | 0,0091 |
| IEEE 300 | 0,0279 | 0,0245 | 0,0326 | 0,0267 | 0,0303 |
| MATPOWER | 0,1422 | 0,0976 | 0,1336 | 0,1026 | 0,2155 |
| Sul-Sudeste | 0,0435 | 0,0365 | 0,0538 | 0,0370 | 0,0623 |

**Figura 7.3** – Tempo médio de *CPU* após mil repetições do processo de Newton-Raphson, de acordo com o método linear e o sistema elétrico

Na Figura 7.3 os sistemas-testes estão representados conforme simbologia identificadas na legenda a exemplo de —▲— representar o sistema MATPOWER. As diferenças percentuais (ganhos) de tempo do BiCGStab em relação ao GMRES e ao método direto também estão destacadas.

Observa-se ainda na Figura 7.3, que independentemente do sistema elétrico considerado, o desempenho do método Newton-Raphson com BiCGStab+$ILU(\boldsymbol{T}_3)$ é sempre superior aos dos demais métodos, incluindo mesmo o tradicional método direto. Observa-se ainda que para o sistema MATPOWER 3375 barras, o problema de fluxo de carga resultou ser consideravelmente mais custoso do que os outros sistemas (IEEE 118 barras, IEEE 300 barras, e o Brasil Sul-Sudeste. Para este sistema mais custoso, o melhor desempenho foi obtido para BiCGStab e o pior para o método direto. Este resultado é um indicativo de que a metodologia proposta seja competitiva e até mesmo mais vantajosa do que o uso do método direto à medida que o sistema se torna mais complexo.

### 7.4.2 Efeito da dimensão do sistema elétrico

Conforme visto na subseção anterior, para problemas de fluxo de carga de custo computacional relativamente baixo, o método de Newton-Raphson apresenta desempenho similar, independente do método linear utilizado para resolver o subproblema linear de cada iteração. A combinação BiCGStab+$ILU(\boldsymbol{T}_3)$ mostrou-se mais vantajosa à medida que o porte do sistema cresce. Para confirmar esta constatação, avalia-se se estas propriedades se verificam quando se considera sistemas elétricos de maiores dimensões. Por isso, estudos similares foram realizados em mais outros três sistemas-teste, todos fictícios, gerados a partir de expansões e interconexões do sistema MATPOWER 3375 barras. Este sistema original, com 3374 barras (lembrando que em relação aos dados originais [75] uma das suas barras foi removida propositadamente, pois é isolada nos dados originais) será denominado sis1. Os demais sistemas foram denominados sis2, sis3 e sis4. A barra *swing* nestes três últimos sistemas foi excluída. Todos estes sistemas fictícios foram criados, porque não se dispunha de dados de sistemas de maior porte no formato do MATPOWER.

Os sistemas foram concebidos a partir de duas interconexões entre sistemas, uma delas sempre no sistema sis1, formando sempre um sistema síncrono. A Figura 7.4 ilustra, por exemplo, esquema de formação para o sistema sis4, concebido portanto, a partir de quatro sistemas sis1. A Tabela 7.1 apresenta dados básicos relativos às dimensões de cada sistema, contendo o número de barras; a quantidade de barras PQ, $n_{\mathrm{PQ}}$ e barras PV, $n_{\mathrm{PV}}$, respectivamente; e a ordem *n* da matriz jacobiana.

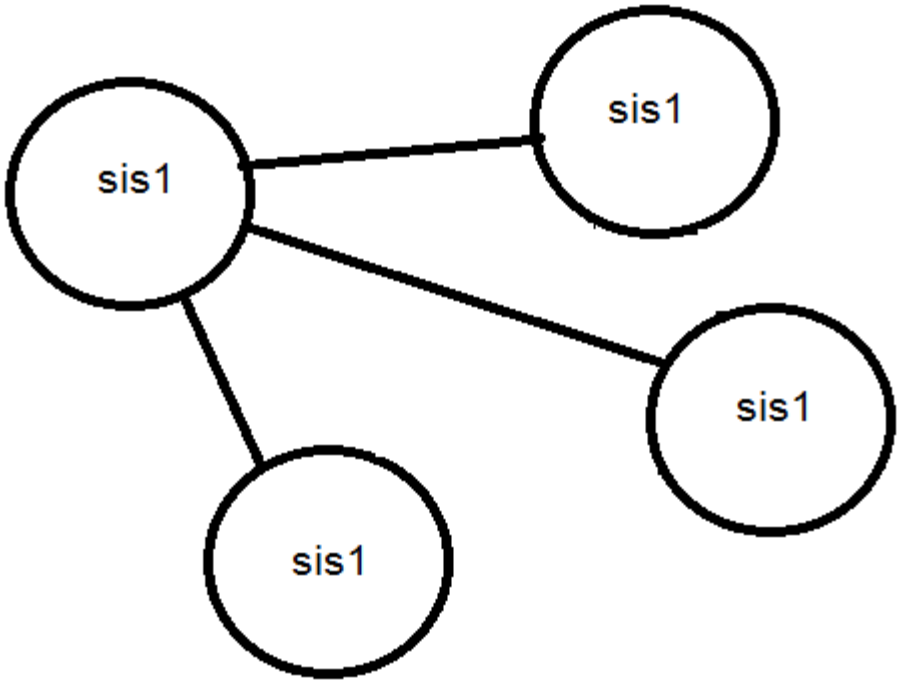


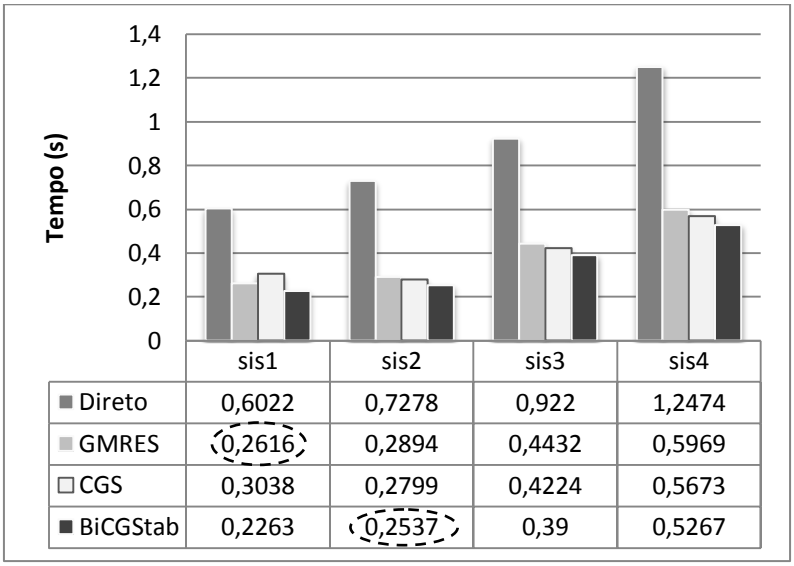

**Figura 7.4** – Sistema elétrico fictício sis4, formado pela interconexão de quatro sistemas sis1

**Tabela 7.1** - Dados básicos dos sistemas

| *Sistemas* | *barras* | $n_{PQ}$ | $n_{PV}$ | *n* |
|---|---|---|---|---|
| sis1 | 3374 | 2982 | 391 | 6355 |
| sis2 | 6747 | 5964 | 782 | 12710 |
| sis3 | 10120 | 8946 | 1173 | 19065 |
| sis4 | 13493 | 11928 | 1564 | 25420 |

Na Tabela 7.1, o maior sistema tem ordem 25420, para um total de 13493 barras. Para este sistema, a matriz $\boldsymbol{C}$ de zeros no pré-condicionador $\boldsymbol{T}_3$ tem dimensão 11928 × 13492.

Calcularam-se os estados dos quatro sistemas por meio do método Newton-Raphson utilizando os métodos: direto e iterativos GMRES, CGS e BiCGStab (todos usando pré-condicionador baseado em $ILU(\boldsymbol{T}_3)$), como realizado anteriormente para o caso base. Para este teste, a *drop tolerance* foi fixada em $2 \times 10^{-5}$ e a tolerância de convergência adotada para os métodos iterativos para sistemas lineares foi $tol = 10^{-4}$. Aceitou-se que houve convergência do processo iterativo de Newton-Raphson somente quando o *mismatch* foi inferior a $10^{-6}$. A Figura 7.5 ilustra os tempos de *CPU* obtidos nas simulações.




| | sis1 | sis2 | sis3 | sis4 |
|---|---|---|---|---|
| Direto | 0,6022 | 0,7278 | 0,922 | 1,2474 |
| GMRES | 0,2616 | 0,2894 | 0,4432 | 0,5969 |
| CGS | 0,3038 | 0,2799 | 0,4224 | 0,5673 |
| BiCGStab | 0,2263 | 0,2537 | 0,39 | 0,5267 |

**Figura 7.5** – Evolução do tempo de *CPU* mediante crescimento da ordem do sistema

Na Figura 7.5, os resultados mostram que o desempenho do método BiCGStab+$ILU(\boldsymbol{T}_3)$ em todas as simulações supera os demais. E em todos os casos, em termos de tempo de CPU, os métodos iterativos são bastante superiores ao método direto. Portanto, demonstra-se que quanto maior o porte do sistema, o método BiCGStab tende a ter o desempenho mais vantajoso que os outros métodos iterativos lineares

estudados. Evidentemente, esta tendência é esperada para sistemas até de maiores dimensões.

## 7.5 INFLUÊNCIA DO CARREGAMENTO

Outro fator que pode afetar as características de convergência associadas ao problema de fluxo de carga é a condição de carga do sistema. Por isso, nesta seção, avalia-se o desempenho e a precisão obtida ao se utilizar métodos iterativos para solucionar o *PFC* em condições estressantes de carregamento.

O carregamento progressivo do sistema foi simulado aumentando-se gradualmente a carga (potência ativa e reativa) e a geração do sistema. Este procedimento é equivalente a incrementar o valor de *p*. Na carga, manteve-se o fator de potência e na geração (barras PV) aumentou-se a geração de potência ativa nas mesmas proporções ao que se fez nas cargas ativas, assim como os limites de reativos das unidades geradoras. Este último procedimento foi implementado a fim de evitar a violação dos limites de potência reativa dos geradores. Neste caso, seria necessário alterar o tipo de barra do gerador atingido de PV para PQ, o que alteraria a dimensão da matriz jacobiana do *PFC*. Esta situação não foi tratada neste trabalho.

As avaliações foram baseadas apenas em estudos no sistema MATPOWER 3375 barras, pois este é o maior dos sistemas reais que se dispõe de dados e o que se considerou que demandaria maior complexidade para o *PFC*.

### 7.5.1 Precisão dos cálculos

Inicialmente, avaliou-se a precisão dos resultados obtidos utilizando métodos iterativos lineares frente àqueles calculados da forma convencional, com o método direto. Com esta finalidade, foram realizadas simulações variando-se o carregamento do sistema, alterando-se progressivamente o parâmetro *p* nas equações (4.4) e (4.5). Esta variação foi possível até 1,4695, ocorrendo divergência para valores superiores.

Na resolução dos sistemas lineares, sempre foi considerada a opção com reordenamento *AMD*. Para os métodos iterativos, implementou-se apenas a opção de solucionador BiCGStab+$ILU(\boldsymbol{T}_3)$.

A Figura 7.6 ilustra o perfil da magnitude de tensão obtido para uma barra de carga do sistema, ao se solucionar os *PFC*s utilizando os métodos direto e iterativo. Em função da precisão de tolerância requerida para o *PFC*, observa-se que ambos os métodos levam a soluções similares. Comportamento semelhante é obtido para as demais barras de carga do sistema. Na Figura 7.7, apresenta-se um gráfico indicando o número de iteração requerido para convergência dos fluxos de carga calculados até o ponto de máximo carregamento. Confirma-se a precisão dos métodos direto e iterativo, obtendo-se o mesmo número de iterações à medida que se aumenta *p*.

A partir do gráfico da Figura 7.7, observa-se que é notório o aumento do número de iterações requerido para convergência do *PFC* próximo ao ponto de colapso de tensão. Nesta situação, a matriz jacobiana fica muito próxima de se tornar singular. Evidentemente, este processo reflete-se no tempo de *CPU*, que é também mensurado para essas condições e ilustrada a sua evolução em função de *p* na Figura 7.8.

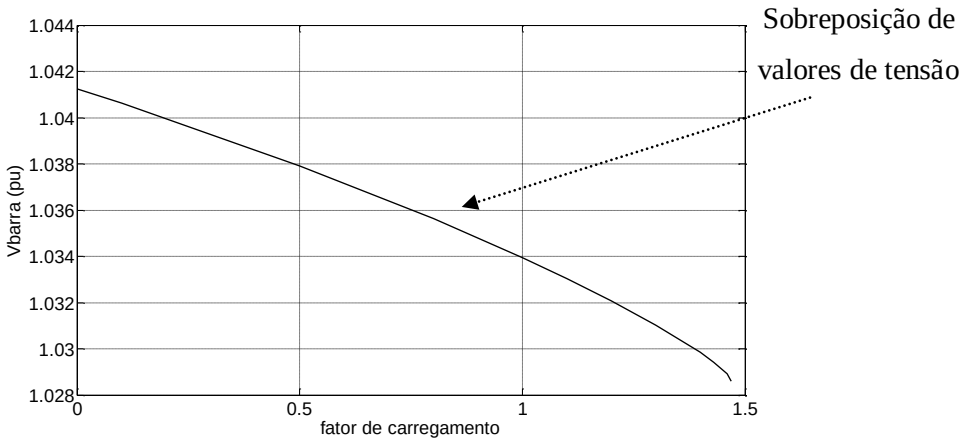



**Figura 7.6** - Perfil de tensão de barra obtida pelos métodos: direto e BiCGStab+$ILU(\boldsymbol{T}_3)$ com evolução da carga

No gráfico da Figura 7.6, visualiza-se o intervalo onde estão registrados os valores de tensão (pu) entre $1{,}028 < V_{barra} < 1{,}042$ correspondentes ao intervalo de aplicação do fator de carregamento, iniciando no caso base (fator de carregamento igual a 0) até o ponto em que o processo de convergência do *PFC* fica inviável ($p$ = 1,4695).

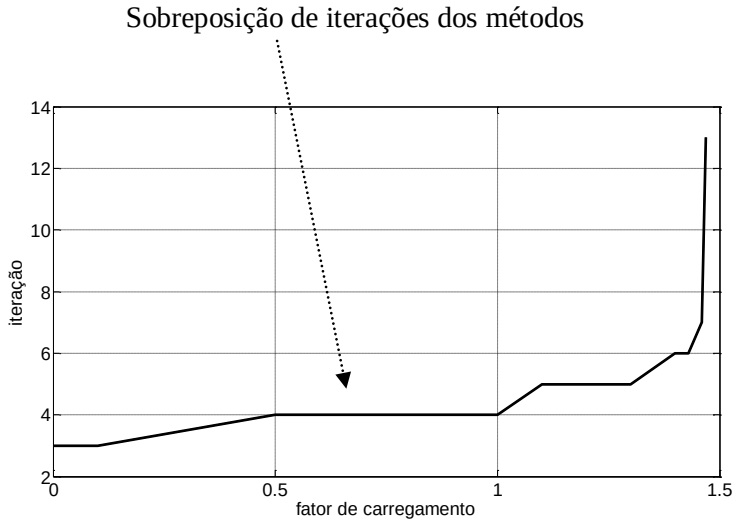



**Figura 7.7** – Perfil do número de iterações dos métodos com evolução da carga

Deve ser enfatizado que em todas as situações, mesmo próximo ao ponto de colapso de tensão, o método iterativo com as características já testadas anteriormente sempre teve desempenho superior ao método direto, atendendo a condições de precisão esperada para convergência e menor tempo de processamento.

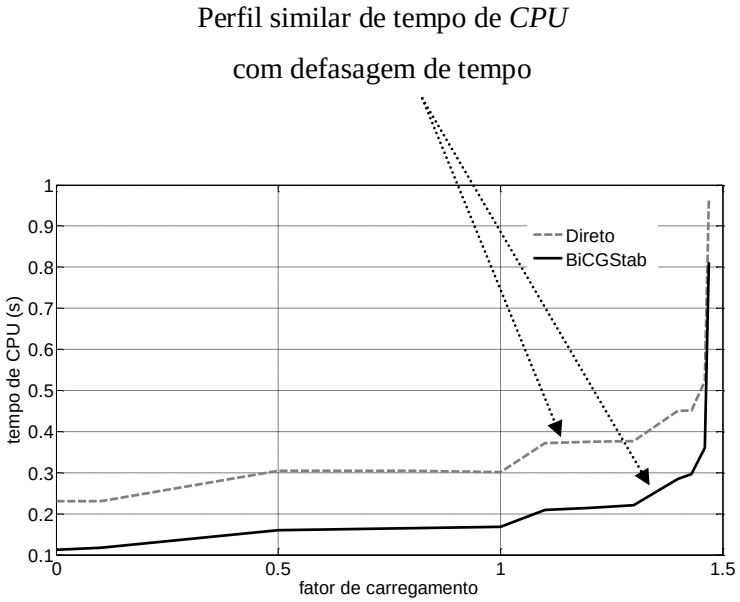



**Figura 7.8** – Comportamento dos métodos direto e BiCGStab+$ILU(\boldsymbol{T}_3)$com evolução da carga

### 7.5.2 – Tempos de processamento em condições extremas de carregamento

Com o objetivo de avaliar o desempenho do método iterativo na condição mais adversa de carregamento – denominada limite de convergência (LC) – aferiu-se o tempo de CPU nessa condição e fez-se a comparação com o caso base – Carga Pesada (CP). Utilizaram-se os mesmos parâmetros para os métodos iterativos lineares já apresentados anteriormente, bem como opção de reordenamento. Ou seja, solucionador+$ILU(\boldsymbol{T}_3)$+*AMD*.

A Figura 7.9 ilustra os resultados observados para o comportamento dos métodos iterativos GMRES, BiCGStab, BiCG, CGS, além do método direto. Nas duas condições, confirmou-se a superioridade do método BiCGStab. A superioridade com relação ao método direto na condição de máximo carregamento é de cerca de 60% e de 27% com relação ao método GMRES.

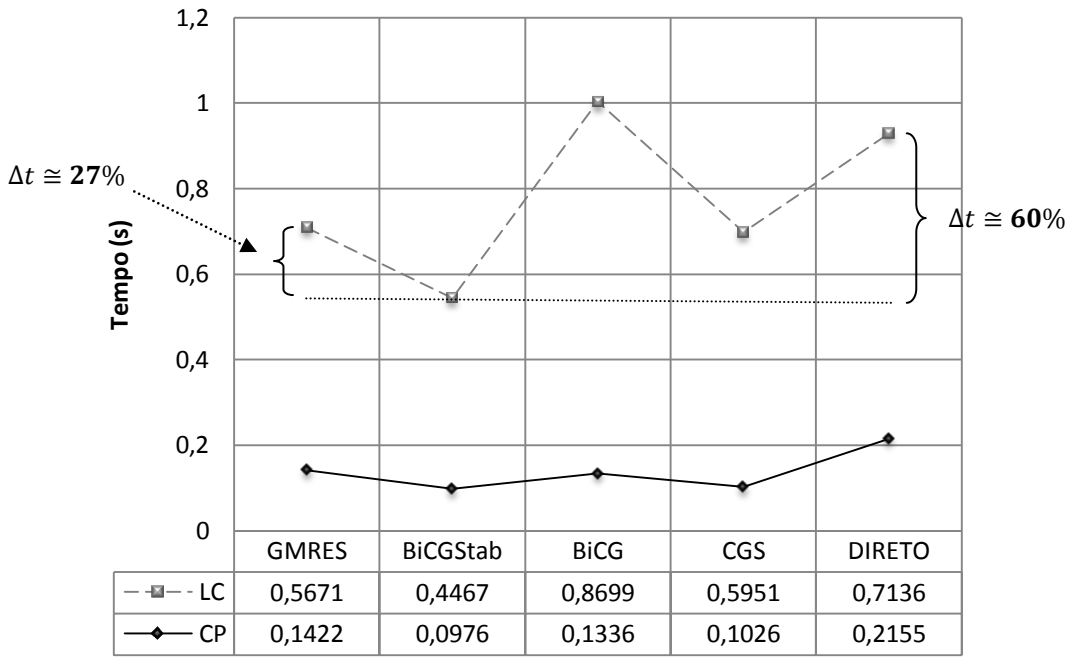



**Figura 7.9** – Desempenho do Newton-Raphson sob carregamentos e métodos distintos

Confirma-se ainda que quanto mais oneroso o problema de fluxo de potência, maior a vantagem do método proposto BiCGStab+$ILU(\boldsymbol{T}_3)$ com relação aos outros métodos lineares e, em particular, ao método direto.

Conclusão: neste capítulo mostrou-se que o método iterativo linear proposto torna o método Newton-Raphson mais eficiente do que os demais métodos lineares (iterativos e direto). As conclusões foram obtidas com base em simulações de sistemas-teste, incluindo situações em que se testou até mesmo pontos de operação muito próximo à singularidade da matriz jacobiana do *PFC* (ponto de colapso de tensão, equivalente ao máximo carregamento do sistema). Os experimentos numéricos indicam que a vantagem se torna mais evidente à medida que a solução do problema não-linear é mais onerosa computacionalmente. Esta vantagem foi verificada para um conjunto de sistemas elétricos reais e para sistemas de grandes dimensões.

# Capítulo 8
# Conclusões e Trabalhos Futuros

## 8.1 CONCLUSÕES GERAIS

Neste trabalho foi realizado um estudo sistemático de técnicas computacionais para solução de sistemas de equações lineares de grande porte e esparsos. Foi proposta uma técnica geral que combina de forma adequada reordenamento, desacoplamento, pré-condicionamento e métodos iterativos lineares clássicos. A proposta de um algoritimo simples para a identificação de subsistemas fracamente acoplados foi uma das contribuições deste trabalho. Cada uma das outras técnicas (reordenamento, pré-condicionamento, etc) é bem conhecida na literatura e não foram propostas modificações em nenhuma delas em particular. A maior contribuição desta tese consiste na proposta de uma arquitetura de combinação adequada destas técnicas básicas.

O algoritmo geral resultante da combinação proposta possui vários (sub-) algoritmos básicos e parâmetros que devem ser sintonizados para cada problema particular. A efetividade da técnica proposta foi verificada por meio de simulações numéricas na resolução do problema de fluxo de carga para o sistema elétrico MATPOWER 3375 barras. Foram apresentadas indicações de como sintonizar os parâmetros e de como o algoritmo linear resultante pode ser utilizado no algoritmo de Newton-Raphson. Outra contribuição foi mostrar que o método BiCGStab, dentro da estrutura de combinação com pré-condicionamento e reordenamento proposto, é recomendável para ser utilizado em problemas de fluxo de carga. Isto porque apresentou desempenho superior ao método GMRES, assim como a todos os outros métodos tradicionais como BiCG, CGS e o método direto. Na literatura do problema de fluxo de carga, para o subproblema linear do algoritmo de Newton-Raphson, a maioria dos trabalhos usam o método GMRES com alguma combinação de condicionamento e reordenamento.

Simulações numéricas para várias situações com carregamento progressivo do sistema até o seu ponto de colapso indicaram a eficácia do método BiCGStab também para essas condições. Os resultados em termos de tempo de processamento demonstrou a superioridade do método iterativo em relação ao direto, mantendo a mesma precisão de resultados. Estas conclusões evidenciam a opção de uso de métodos iterativos, por exemplo, para o cálculo de fluxo de carga continuado, técnica mais apropriada para se determinar a barra crítica e o ponto de máximo carregamento de sistemas elétricos de potência.

Os experimentos numéricos indicaram também que a vantagem do BiCGStab aumenta à medida que o porte do sistema a ser resolvido cresce, seja devido à complexidade de estrutura, seja quanto maior for a ordem do sistema.

## 8.2 TRABALHOS FUTUROS

Como sugestões para investigações futuras, podem ser indicadas as seguintes linhas de ação:

- Analisar o problema que vise propiciar um procedimento de escolha adaptativa do método linear/pré-condicionadores mais apropriados para cada iteração: de fato, neste trabalho, considerou-se a aplicação do mesmo método linear e mesmos fatores pré-condicionadores para todos os subproblemas lineares das iterações do método de Newton Raphson. Uma escolha adaptativa, possivelmente, permitiria um ganho em desempenho.

- Investigar outras configurações para o cálculo dos pré-condicionadores: pequenas mudanças nas combinações da ordem com que as diversas técnicas numéricas são utilizadas podem afetar o desempenho global do algoritmo resultante. Neste trabalho, foram consideradas algumas possibilidades para a matriz de coeficientes para o que de fato, o cálculo de pré-condicionadores fosse realizado. Neste sentido, outras possibilidades podem ser investigadas.

- Realizar testes em sistemas reais de maiores dimensões que as testadas neste trabalho.

- Extender a metodologia para outros problemas estudados para a análise de sistemas elétricos de potência, como fluxo continuado, fluxo de carga ótimo, cálculo de resposta em frequência e simulação de transitórios eletromecânicos e transitórios eletromagnéticos, análise modal, entre outras aplicações.

# Referências Bibliográficas


[1] R. Idema, D. J. Lahaye, C. Vuik and L. V. d. Sluis, "Scalable Newton-Krylov solver for very large power flow problems," *IEEE Transactions on Power Systems,* vol. 27, no. 1, pp. 390 - 396, 2012.

[2] C. Borges, F. D.M. e A. Coutinho, "Utilização de Método Tipo Gradiente Conjugado na Aceleração do Fluxo de Potência em Computação Vetorial," *XIV Seminário Nacional de Produção e Transmissão de Energia Elétrica,* 1997.

[3] A. Monticelli, Fluxo de Carga em Redes de Energia Elétrica, São Paulo: Edgard Blücher Ltda, 1983.

[4] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," *Proceeding ACM 24 th National Conference,* pp. 157-172, 1969.

[5] J. Gilbert, C. Moler and R. Schreiber, "Sparse Matrices in Matlab:Design and Implementation," *SIAM Journal on Matrix Analysis,* vol. 13, no. 1, pp. 333-356, 1992.

[6] M. Pai and H. Dag, "Iterative solver techniques in large scale power system computation," *Proceedings of the 36th IEEE Conference on Decision and Control,* vol. 4, pp. 3861-3866, 1997.

[7] Y. Cheng and C. Shen, "A Jacobian-free Newton-GMRES(m) method with adaptive preconditioner and its application for power flow calculation," *IEEE Transactions on Power Systems,* vol. 21, no. 3, pp. 1096-1103, 2006.

[8] A. Semlyen, "Fundamental concepts of a Krylov subspace power flow methodology," *IEEE Transactions on Power Systems vol.11 nº 3,* pp. 1528-1537, 1996.

[9] M. Pai, "A new preconditioning technique for the GMRES algorithm in power flow and P-V curve calculations," *International Journal of Electrical Power & Energy Systems,* vol. 25, no. 3, pp. 239-245, 2003.

[10] H. Dag and A. Semlyen, "A new preconditioned conjugate gradient power flow," *IEEE Transactions on Power Systems,* vol. 18, no. 4, pp. 1248-1255, 2003.

[11] R. D. Zimmerman, C. E. Murillo-Sanchez and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education," *IEEE Transactions on Power Systems,* vol. 26, no. 1, pp. 12-19, 2011.

[12] R. D.Zimmerman, C. E. Sanchez and D. Gan, "A MATLAB Power System Simulation Package," MATPOWER, 14 dec 2011. [Online]. Available: http://www.pserc.cornell.edu/matpower. [Accessed 23 feb 2013].

[13] H. van der Vorst, Iterative Krylov methods for large linear systems, Cambridge: Cambridge University Press, 2003.

[14] D. Young, Iterative solution of large linear systems, New York, USA: Academic Press, 1971.

[15] G. Golub and C. Loan, Matrix Computations, Baltimore: The Johns Hopkins University Press, 1996.

[16] J. Sheldon, "On the numerical solution of elliptic difference equations," *Mathematical Tables and other Aids to Computation,* vol. 9, no. 51, pp. 101-112, 1955.

[17] C. Paige and M. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM Journal on Numerical Analysis,* vol. 12, no. 4, pp. 617-629, 1975.

[18] Y. Saad, Iterative methods for sparse linear systems, Philadelphia: Society for Industrial and Applied Mathematics, 2003.

[19] F. Galiana, H. Javidi and S. McFee, "On the application of a preconditioned conjugate gradient algorithm to power network analysis," in *Power Industry Computer Application Conference.*, Phoenix, Arizona, 1993.

[20] C. Borges e A. F. Coutinho, "Solução de Fluxo de Potência em Ambiente Vetorial usando o Método do Gradiente Bi-Conjugado Estabilizado," *Anais do 11º Congresso Brasileiro de Automática,vol.1,* pp. 160-166, 1996.

[21] A. Flueck and H. Chiang, "Solving the nonlinear power flow equations with an inexact Newton method using GMRES," *IEEE Transactions on Power Systems,* vol. 13, no. 2, pp. 267-273, 1998.

[22] F. León and A. Semlyen, "Iterative solves in the Newton power flow problem: Preconditioners, inexact solutions and partial Jacobian updates," *IEE Proceedings Generation, Transmission and Distribution,* vol. 149, no. 4, pp. 479-484, 2002.

[23] Y. Saad and M. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific Computing,* vol. 7, no. 3, pp. 856-869, 1986.

[24] G. Recktenwald, "Stopping Criteria for Iterative Solution Methods," Portland State University, Portland, USA, 2012.

[25] G. Golub and J. Ortega, Scientific Computing an Introduction with Parallel Computing, Boston, USA: Academic Press, 1993.

[26] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *Journal of Computation Physics,* vol. 182, pp. 418-477, 2002.

[27] L. Carvalho and S. Gratton, *Avanços em métodos de Krylov para solução de sistemas lineares de grande porte*, São Carlos, Brasil: Sociedade Brasileira de Matemática Aplicada e Computacional, 2009.

[28] R. Barret, M. Berry, T. Chan and e. all, Templates for the solution of linear systems: building blocks for iterative methods, Philadelphia,PA: Society for Industrial and Applied Mathematics, 1994.

[29] S. Khaitan and J. McCalley, "A class of new preconditioners for linear solvers used in power system time-domain simulation," *IEEE Transactions on Power Systems,* vol. 25, no. 4, pp. 1835-1844, 2010.

[30] Y. Chen, C. Shen and J. Wang, "Distributed transient stability simulation of power systems based on a Jacobian-Free Newton-GMRES method," *IEEE Transactions on Power Systems,* vol. 24, no. 1, pp. 146-156, 2009.

[31] C. Paige, M. Rozložník and Z. Strakos, "Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES," *SIAM Journal on Matrix Analysis and Applications,* vol. 28, pp. 264-284, 2006.

[32] J. Drkosová, A. Greenbaum, M. Rozložník and Z. Strakos, "Numerical stability of the GMRES method," *BIT Numerical Mathematics,* vol. 35, pp. 308-330, 1995.

[33] H. van der Vorst, "Bi-CGStab: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM Journal on Scientific Computing*, vol. 13, no. 2, pp. 631-644, 1992.

[34] V. Simoncini and D. B. Szyld, "Recent Computational Developments in Krylov Subspace Methods for Linear Systems," *Numerical Linear Algebra with Applications*, vol. 14, no. 1, pp. 1-59, 2007.

[35] P. Sonneveld, "CGS: a fast Lanczos-type solver for nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, no. 1, pp. 36-52, 1989.

[36] C. Lanczos, "Solution of systems of linear equations by minimized iterations," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 33-53, 1952.

[37] R. Fletcher, "Conjugate gradient methods for indefinite systems," *Numerical Analysis Lecture Notes in Mathematics*, vol. 506, pp. 73-89, 1976.

[38] C. Brezinski, M. Zaglia and H. Sadok, "Breakdowns in the implementation of the Lanczos method for solving linear systems," *Computers and Mathematics with Applications*, vol. 33, pp. 31-44, 1997.

[39] Y. Saad and H. van der Vorst, "Iterative solution of linear systems in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 1, pp. 1-33, 2000.

[40] J. Pessanha, A. Paz and R. Prada, "Aplicação do método GMRES em estudos de estabilidade de sistemas de energia elétrica," *Sba: Controle & Automação Sociedade Brasileira de Automatica*, vol. 23, no. 3, pp. 321-330, 2012.

[41] J. A. Meijerink and H. A. Van der Vorst, "Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Pratical Problems," *Journal of Computational Physics*, vol. 44, no. 1, pp. 134-155, 1981.

[42] T. U. o. Q. Australia, "Test Systems," Faculty of Engineering, Architecture and Information Technology, [Online]. Available: http://itee.uq.edu.au/pss-l/test%20system.htm. [Acesso em 5 jan 2013].

[43] U. o. Washington, "Power systems test case archive," University of Washington,

[Online]. Available: http://www.ee.washington.edu/research/pstca/pf300/pg_tca300fig.htm. [Accessed 23 ap 2013].

[44] M. Benzi, Szyld, D.B and A. Duin, "Orderings for incomplete factorization preconditioning of nonsymmetric problems," *Society for Industrial and Applied Mathematics,* vol. 20, no. 5, pp. 1652-1670, 1999.

[45] Y. Saad, "ILUT: A dual threshold incomplete ILU preconditioner," *Numerical Linear Algebra Application,* vol. nº 1, pp. 387-402, 1994.

[46] K. Chen, *Matrix Preconditioning Techniques and Applications,* Cambridge, U.K.: Cambridge University Press, 2005.

[47] Y. Saad, "SPARSKIT: A basic tool kit for sparse matrix computations," Research Institute for Advance Computer Science, Moffett Field, CA, USA, 1990.

[48] E. D'Azevedo, F. Forsyth and W. Tang, "Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems," *SIAM-Journal on Matrix Analysis and Applications,* vol. 13, pp. 944-961, 1992.

[49] E. D'Azevedo, F. Forsyth and W. Tang, "Towards a cost effective ILU preconditioner with high level fill," *BIT,* vol. 31, pp. 442-463, 1992.

[50] D. Osei-Kuffuor and Y. Saad, "A comparison of preconditioners for complex-valued matrices," pp. 1-18, 2007.

[51] W. Tinney and J. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proceedings of the IEEE,* vol. 55, no. 11, pp. 1801-1809, 1967.

[52] I. Duff, M. Erisman and J. Reid, Direct Methods for Sparse Matrices, Oxford: Clarendon Press, 1986.

[53] I. Duff and J. Koster, "The design and use of algorithms for permuting large entries to the diagonal of sparse matrices," *SIAM J. Matrix Analysis and Applications,* vol. 20, pp. 889-901, 1999.

[54] H. Elman, "A stability analysis of incomplete LU factorizations," *Mathematics of Computation,* vol. 47, no. 175, pp. 191-217, 1986.

[55] T. Davis, "Direct Methods for Sparse Linear Systems," in *SIAM book Series on the Fundamentals of Algorithms*, SIAM, 2006.

[56] S. Sloan, "An algorithm for profile and wavefront reduction of sparse matrices," *International Journal for Numerical Methods in Engineering,* vol. 23, pp. 239-251, 1986.

[57] P. Amestoy, T. Davis and I. Duff, "An approximate minimum degree ordering algorithm," *SIAM Journal on Matrix Analysis and Applications,* vol. 17, no. 4, pp. 886-905, 1996.

[58] M. Benzi, J. Haws and M. Tuma, "Preconditioning highly indefinite and nonsymmetric matrices," *Society for Industrial and Applied Mathematics,* vol. 22, no. Nº 4, pp. 1333-1353, 2000.

[59] J. Zhang, "A grid based multilevel incomplete LU factorization preconditioning technique for general sparse matrices," *Applied Mathematics and Computation,* vol. 124, no. 1, pp. 95-115, 2001.

[60] A. Monticelli, A. Garcia and O. Saavedra, "Fast Decoupled Load Flow: Hypothesis, Derivations and Testing," *IEEE Transactions on Power Systems,* vol. 5, no. 4, pp. 1425-1431, 1990.

[61] B. Stott and O. Alsac, "Fast Decoupled Load Flow," *IEEE Transactions on Power Apparatus and Systems,* Vols. PAS-93, no. 3, pp. 859-869, July 1974.

[62] S. Biehl, *Uma nova abordagem para resolução de problemas de fluxo de carga com variáveis discretas,* São Carlos: Universidade de São Paulo, 2012.

[63] V. Sousa, *Resolução do problema de fluxo de potência ótimo reativo via método da função lagrangiana barreira modificada,* São Carlos: Universidade de São Paulo, 2006.

[64] M.Gabriel e T.Neunhoeffer, “Parallel point and domain oriented multilevel methods for elliptic PDE's on workstation networks,” *Journal of Computational and Applied Mathematics,* vol. 66, n. 1-2, pp. 267-278, 1996.

[65] P. Anderson and A. Fouad, Power system control and stability, New York: IEEE Press, 2003.

[66] J. Pessanha, C. Portugal, R. Prada and A. Paz, "Critical investigation of preconditioned GMRES via incomplete LU factorization applied to power flow simulation," *Electrical Power and Energy Systems,* vol. 33, no. 10, pp. 1695-1701, 2011.

[67] W. Tinney and C. Hart, "Power flow solution by Newton's method," *IEEE Transactions on Power Apparatus and Systems,* Vols. PAS-86, no. 11, pp. 1449-1460, 1967.

[68] J. Pessanha, A. Paz, R. Prada and C. Roma, "Making use of BDF-GMRES methods for solving short and long-term dynamics in power systems," *Electrical Power and Energy Systems,* vol. 45, no. 1, pp. 293-302, 2013.

[69] T. Davis, P. Amestroy and I. Duff, "AMD: approximate minimum degree ordering," GNU LGPL License, 2004. [Online]. Available: http://www.cise.ufl.edu/research/sparse/amd. [Accessed 19 dec 2012].

[70] T. Davis, P. Amestroy and I. Duff, "http://www.cise.ufl.edu/research/sparse/amd," GNU LGPL License, 2004. [Online]. [Accessed 19 dec 2012].

[71] D. Chaniotis and M. Pai, "A new preconditioning technique for the GMRES algorithm in power flow and P-V curve calculations," *International Journal of Electrical Power & Energy Systems,* vol. 25, no. 3, pp. 239-245, 2003.

[72] C. Poma, *Um solucionador iterativo para sistemas-lineares: aplicação no problema do fluxo de carga,* Rio de Janeiro: PUC-Rio, 2010.

[73] R. Zimmerman and C. Murillo-Sanchez, *Matpower 4.1 User´s Manual,* Power Systems Engineering Research Center , 2011.

[74] I. The MathWorks, "MATLAB The Language of Technical Computing," MathWorks, 1994. [Online]. Available: http://www.mathworks.com/products/matlab. [Accessed 5 jan 2013].

[75] R. Zimmerman, "Powerweb," Power Systems Engineering Research Center, [Online]. Available: http://www.pserc.cornell.edu:8082/powerweb/. [Accessed 5 jan 2013].

[76] F. S. Supporter, "GNU Operating System," Free Software Foundation,Inc.,

[Online]. Available: http://www.gnu.org/licenses/gpl.html. [Accessed 17 feb 2014].

[77] U. o. Washington, "University of Washington Electrical Engineering," University of Washington, [Online]. Available: http://www.ee.washington.edu/research/pstca/. [Accessed 23 ap 2013].

[78] K. Hedman, R. O'Neill, E. Fisher and S. Oren, "Optimal transmission switching with contingency analysis," in *Power and Energy Society General Meeting*, Minneapolis, MN, USA, 2010.

[79] E. Makram, Z. Zheng and A. Girgis, "An improved model in optimal PMU placement considering sensitivity analysis," in *Power Systems Conference and Exposition (PSCE)*, Phoenix, AZ, USA , 2011.

[80] D. Feng, J. Zhong and D. Gan, "Reactive market power analysis using must-run indices," *IEEE Transactions on Power Systems*, vol. 23, no. 2, pp. 755-765, 2008.

[81] C. Liang, C. Chung, K. Wong and X. Duan, "Parallel optimal reactive power flow based on cooperative co-evolutionary differential evolution and power system decomposition," *IEEE Transactions on Power Systems*, vol. 22, no. 1, pp. 249-257, 2007.

[82] S. Blumsack, *Network topologies and transmission investment under electric-industry restructuring*, Pittsburg, Pennsylvania: Carnegie Mellon University, 2006.

[83] CEPEL, *PacDyn, User´s Manual, versão 8.0*, Rio de Janeiro: CEPEL, 2008.

[84] CEPEL, "Anarede," Centro de Pesquisas de Energia Elétrica, [Online]. Available: http://www.anarede.cepel.br/pub.html. [Accessed 20 jan 2013].

[85] ONS, "ONS Operador Nacional do Sistema Elétrico," ONS, [Online]. Available: http://www.ons.org.br. [Accessed 20 dec 2013].

[86] P. T. S. Operator, "PSE S.A. Polish Transmission System Operator," PSE S.A., [Online]. Available: http://www.pse-operator.pl/index.php?dzid=78. [Accessed 23 ap 2013].

[87] N. J. Higham and F. Tisseur, "A Block Algorithm for Matrix 1-Norm Estimation with an Application to 1-Norm Pseudospectra," *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1185-1201, 2000.

[88] W. W. Hager, "Conditions Estimates," *SIAM Journal on Scientific and Statistical Computing*, vol. 5, no. 2, pp. 311-316, 1984.

[89] P. Amestoy, T. Davis and I. Duff, "Algorithm 837:AMD, An approximate minimum degree ordering algorithm," *ACM Transactions on Mathematical Software*, vol. 30, no. 3, pp. 381-388, 2004.

[90] M. Benzi and A. J. Whaten, "Some Preconditioning Techniques for Saddle Point Problems," University of Oxford, Oxford, 2006.

[91] M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409-436, 1952.

[92] M. Hestenes, Conjugate Direction Methods in Optimization, Berlin: Springer-Verlag, 1980.

[93] O. Axelsson, Solution of Linear Systems of Equations: Iterative Methods, Berlin: V.A. Barke,Springer-Verlag, 1977.

# Apêndice I

Programa de teste para solucionadores iterativos lineares

ambiente Matlab

```
clc
clear all
teste=1; %teste=input('defina o sistema: insira um valor de 1 a 4 e pressione enter')
if teste==1
%load jacobiano_case3375wp_A_b sistema de dimensão original
   load jac_case3375wp_sistema_6357;
   n1=var_teta;
   A=J; b=-F;
end
if teste==2
   load jac_case3375wp_sistema_duasvezes6357; % sistema com dimensão duplicada
   n1=var_teta;
   A=J; b=-F;
end
if teste==3
   load jac_case3375wp_sistema_tresvezes6357; % sistema com dimensão triplicada
   n1=var_teta;
   A=J; b=-F;
end
if teste==4
   load jac_case3375wp_sistema_quatrovezes6357; % sistema com dimensão quadruplicada
   n1=var_teta;
   A=J; b=-F;
end

%Definição de parâmetros iniciais
tol=1e-4;
n=size(A,1);
n1=var_teta;
n2=n-n1;
A1=sparse(A(1:n1,1:n1));

%Configurações das matrizes
B=sparse(A(1:n1,n1+1:n));
Bz=sparse(n1,n2,0);
C=sparse(A(n1+1:n,1:n1));
Cz=sparse(n2,n1,0);
D=sparse(A(n1+1:n,n1+1:n));
b=b;%b=sum(A,2);%%resposta esperada = 1
x0=zeros(n,1);
T=[A1 Bz;Cz D]; %permite configurações de T
tau=2.5e-5;
```

```
% tipo de ordenamento
%  ordena=0 ==> sem ordenamento
%  ordena=1 ==> RCM
%  ordena=2 ==> AMD
ordena=2;
if ordena==1
disp('ILU - Aplicando reordenamento RCM')
    p=symrcm(A);
end
if ordena==2
disp('ILU - Aplicando reordenamento AMD')
     p=amd(A);
end
if ordena ~=0
R1=T(p,p);
R=A(p,p);
bR=b(p); %bR=sum(R,2);%resposta forçada igual a 1
else
   disp('ILU - Sem ordenamento aplicado')
   R=A; R1=T; bR=b;
end
N=500; %número de repetições da simulação
if 1
%metodo direto
disp('Metodo direto - Reordenado')
 for k=1:N
ti0=tic;
     x=R\bR;
txbc0(k)=toc(ti0);
 end
ci0=mean(txbc0(5:N))
end

%%droptol=1.5; 2.5 e 6.5e-5
setup.type='ilutp';
setup.droptol=tau;
setup.udiag =0;
setup.thresh=0;
%  Se a opção setup.milu='col' for desativada - a fatoração é diferente e pior
disp(' Caso setup_milu opção 1 = col, 2=row e 0=sem opção.')
disp(' Este setup modifica drasticamente a maneira como ILUTP gera a fatoração ILU.')
disp('A despeito da fatoração ILU os tempos de CPU são similares para  row ou col,')
disp('Nos experimentos a melhor performance sempre ocorre para setup_milu=1')
%setup_milu=input('Escolha o setup.milu option ===>')
setup_milu=1;
if setup_milu==1
setup.milu='col';
else
   if setup_milu==2
      setup.milu='row';
   else
      disp('Nenhum setup.milu ');
```

```
   end
end

if 1
    for k=1:N
      tg2=tic;
       [LA,UA]=ilu(sparse(R),setup);
        txg1(k)=toc(tg2);
   end

tempo_LU_A=mean(txg1(5:N)); % as cinco primeiras amostras são rejeitadas
end
if 1
   for k=1:N
      tg2=tic;
      [LT,UT]=ilu(sparse(R1),setup);
        txg1(k)=toc(tg2);
   end

tempo_LU_T=mean(txg1(5:N))
%pause
end
if 1
tempo_LUA_LUT=[tempo_LU_A tempo_LU_T]
end
 % número máximo de iter=restart*maxit_gmres inner_iter=restart
     maxit=42; restart=6; maxit_gmres=7;
if 0
%GMRES
     disp('GMRES')
     [xg10]=gmres(R,bR,restart,tol,maxit_gmres,LA,UA,x0);
     [xg20]=gmres(R,bR,restart,tol,maxit_gmres,LT,UT,x0);
     disp('bicgstab')
     [xbc1]=bicgstab(R,bR,tol,maxit,LA,UA,x0);
     [xbc2]=bicgstab(R,bR,tol,maxit,LT,UT,x0);
     disp('bicg')
     [xbc3]=bicg(R,bR,tol,maxit,LA,UA,x0);
     [xbc4]=bicg(R,bR,tol,maxit,LT,UT,x0);
     disp('cgs')
     [xbc5]=cgs(R,bR,tol,maxit,LA,UA,x0);
     [xbc6]=cgs(R,bR,tol,maxit,LT,UT,x0);
end

if 1
   for k=1:N
      tg2=tic;
      [xg1,x]=gmres(R,bR,restart,tol,maxit_gmres,LA,UA,x0);
    %[xg1,z]=gmres(R,bR,restart,tol,maxit_gmres,LARA,UARA,x0);
     txg1(k)=toc(tg2);
   end

cg1=mean(txg1(5:N));
```

```
%pause
end
disp('GMRES-ilu(TRA)')
   for k=1:N
      tg4=tic;
      [xg10,x]=gmres(R,bR,restart,tol,maxit_gmres,LT,UT,x0);
    %[xg10,z]=gmres(R,bR,restart,tol,maxit_gmres,LTRA,UTRA,x0);
      txg10(k)=toc(tg4);
   end

cg2=mean(txg10(5:N));
%pause
if 1
%BiCGStab
disp('BiCGstab-ilu(ARA)')
for k=1:N
   ti2=tic;
  % [xbc1,z]=bicgstab(R,bR,tol,maxit,LARA,UARA,x0);
    [xbc1,x]=bicgstab(R,bR,tol,maxit,LA,UA,x0);
   txbc1(k)=toc(ti2);
 end
ci1=mean(txbc1(5:N));
%pause
end
disp('BiCGstab-ilu(TRA)')
for k=1:N
   ti4=tic;
   %[xbc2,z]=bicgstab(R,bR,tol,maxit,LTRA,UTRA,x0);
   [xbc2,x]=bicgstab(R,bR,tol,maxit,LT,UT,x0);
   txbc2(k)=toc(ti4);
 end
ci2=mean(txbc2(5:N));
%pause
if 1
%BiCG
disp('BiCG-ilu(ARA)')
ti6=tic;
   for k=1:N
      ti6=tic;
      [xbc3,z]=bicg(R,bR,tol,maxit,LA,UA,x0);
      txbc3(k)=toc(ti6);
   end
%ci3=txbc3(k)/N;
ci3=mean(txbc3(5:N));
%pause
end

disp('BiCG-ilu(TRA)')
   for k=1:N
      ti8=tic;
      [xbc4,z]=bicg(R,bR,tol,maxit,LT,UT,x0);
      txbc4(k)=toc(ti8);
```

```
   end
%ci4=txbc4(k)/N;
ci4=mean(txbc4(5:N));
%pause
if 1
%CGS
disp('CGS-ilu(ARA)')
   for k=1:N
      ti10=tic;
      [xbc5,z]=cgs(R,bR,tol,maxit,LA,UA,x0);
      txbc5(k)=toc(ti10);
   end
%ci5=txbc5(k)/N;
ci5=mean(txbc5(5:N));
%pause
end
disp('CGS-ilu(TRA)')
ti12=tic;
   for k=1:N
      ti12=tic;
      [xbc6,z]=cgs(R,bR,tol,maxit,LT,UT,x0);
      txbc6(k)=toc(ti12);
   end
%ci6=txbc6(k)/N;
ci6=mean(txbc6(5:N));

opcao=2;
if opcao==2
   disp('todos tempos CPU')
cp4=[ci0 cg1 cg2 ci1 ci2 ci3 ci4 ci5 ci6]
else
   disp('somente tempos de cpu dos métodos pré-condicionados com fatores ILU de T')
   cp4=[cg2 ci2 ci4 ci6]
end
```

# Apêndice II

Programa de teste para solucionadores iterativos lineares

ambiente MATPOWER

```
function [V, converged, i] = newtonpf(Ybus, Sbus, V0, ~, pv, pq, mpopt)
ti0=tic;
loops=1000;
for k_loops=1:loops
%% default arguments
if nargin < 7
   mpopt = mpoption; % ver mpoption.m para alterações na forma de entrada, saída de dados, definição
de parâmetros, constantes etc
end

%% options
tol    = mpopt(2);  % fixada em 1e-6 (tol do resíduo absoluto para o flow)
max_it  = mpopt(3);   % máximo número de iterações do método Newton-Raphson
verbose = mpopt(31);
tol_it=100*tol;  % tol relativa para o resíduo do sistema iterativo Ax=b
%% initialize
converged = 0;
j=sqrt(-1);
i = 0;
V = V0; % tensões iniciais
Va = angle(V); % angulo de fase inicial
Vm = abs(V); % magnitudes iniciais e valores fixados nas barras PV

%disp('barras pv e pq')
%% set up indexing for updating V
npv = length(pv);
npq = length(pq);
j1 = 1;        j2 = npv;             %% j1:j2 - V angle of pv buses
j3 = j2 + 1;    j4 = j2 + npq;     %% j3:j4 - V angle of pq buses
j5 = j4 + 1;    j6 = j4 + npq;     %% j5:j6 - V mag of pq buses

%% evaluate F(x0)
mis = V .* conj(Ybus * V) - Sbus; % voltar
F = [   real(mis([pv; pq]));
       imag(mis(pq))   ];
   var_teta=npq+npv;
   var_V=npq;

%% check tolerancia
normF = norm(F, inf);
if verbose > 1
   fprintf('\n it    max P & Q mismatch (p.u.)');
   fprintf('\n----  ------------------------');
   fprintf('\n%3d       %10.3e', i, normF);
```

```
end
if normF < tol
   converged = 1;
   if verbose > 1
      fprintf('\nConverged!\n');
   end
end

%=====  Dados para resolução dos métodos iterativos
   maxit=3;   % máximo número de iterações do loop interno para GMRES e Broyden
   maxrestart=15; % máximo numero de restarts para GMRES e Broyden
   maxit_bicg=30;  % máximo número de iterações para BiCGstab e CGS
    %define o método iterative de acordo com as opções disponíveis
   metodo=2;  % se método =1 GMRES;  =2 BiCGstab;  =3 Broyden;  =4 CGS.

%% do Newton iterações
while (~converged && i < max_it)
   %% update iteration counter
   i = i + 1;

   %% evolução da Jacobiana
   [dSbus_dVm, dSbus_dVa] = dSbus_dV(Ybus, V);
   j11 = real(dSbus_dVa([pv; pq], [pv; pq]));
   j12 = real(dSbus_dVm([pv; pq], pq));
   j21 = imag(dSbus_dVa(pq, [pv; pq]));
   j22 = imag(dSbus_dVm(pq, pq));

   J = [   j11 j12;
   j21 j22;    ];
iteractive_method=1;% se método iterativo ~=0 => método iterativo ativado caso contrário método
direto
   if iteractive_method
      if i==1
    %    if i<max_it
%save jac_sistema_IEEE300 J F var_teta var_V; % ==> SALVAR JACOBIANO da iteração inicial
% modificações efetuadas para contemplar métodos iterativos
        [n1,n2]=size(j21);
        % p=symrcm(J);
        p=amd(J);
        R=J(p,p);
        bR=-F(p);
        a1=sparse(n1,n2,0);
        T=[j11 j12;
          a1 j22];
       R1=T(p,p);
       setup.type='ilutp';
       setup.droptol=2.5e-5;  % tcpu_broyden=0.109
       % setup.droptol=6.5e-5;   % tcpu_broyden=0.125
       setup.udiag =0;
       setup.thresh=0;
       setup.milu='col';
        [LTR,UTR]=ilu(sparse(R1),setup); % fatoração ILU é feita apenas
```

```
        x0=sparse(var_teta+var_V,1);
      else
         R=J(p,p);
         bR=-F(p);
      end
   %% cálculo da solução e escolha do método
   Comutação do metodo
      case 1
   % MÉTODO GMRES
    [dxR,q]=gmres(R,bR,maxrestart,tol_it,maxit,LTR,UTR,x0);
      case 2
   % MÉTODO BICGStAB
    [dxR,q]=bicgstab(R,bR,tol_it,maxit_bicg,LTR,UTR,x0);
      case 3
    % MÉTODO GOOD BROYDEN
    [dxR,iter_restart,rel_res]=good_broyden(R,bR,maxrestart,maxit,tol_it,LTR,UTR,x0);
      otherwise
        % cgs
    [dxR,q]=cgs(R,bR,tol_it,maxit_bicg,LTR,UTR,x0);
     end
   [qq,qx]=sort(p);  % qx é igual a ordem pela qual os valores de p ocorrem ou P' em P.A.P'.y=P.b x=P'.y
    dx=dxR(qx);
   else
   dx=-J\F;
   end

   %% atualização da tensão
   if npv
      Va(pv) = Va(pv) + dx(j1:j2);
   end
   if npq
      Va(pq) = Va(pq) + dx(j3:j4);
      Vm(pq) = Vm(pq) + dx(j5:j6);
   end

   % voltar linhas abaixo <==========
   V = Vm .* exp(1j * Va);
   Vm = abs(V);%% atualização Vm e Va novamente no caso
   Va = angle(V);

%% evalute F(x)
 %   mis = Vc.* conj(Ybus * Vc) - Sbus;
     mis = V .* conj(Ybus * V) - Sbus;   % voltar
   F = [   real(mis(pv));
         real(mis(pq));
         imag(mis(pq))   ];

   %% check para convergencia
   normF = norm(F, inf);
   if verbose > 1
      fprintf('\n%3d       %10.3e', i, normF);
```

```
   end
   if normF < tol
      converged = 1;
      if verbose
%          fprintf('\nNewton''s method power flow convergiu em %d iterações.\n', i);
         save convergiu i;  % GRAVA arquivo para indicar se convergiu
      end
   end
end

if verbose
   if ~converged
      fprintf('\nNewton''s method power não convergiu in %d iterações.\n', i);
      save divergiu i;  % GRAVA arquivo para indicar se divergiu
   end
end

end   % fim do loop para CPU

tcpu_NR=toc(ti0);
tcpu_NR=tcpu_NR/loops;
save cpu_runpf tcpu_NR;
```