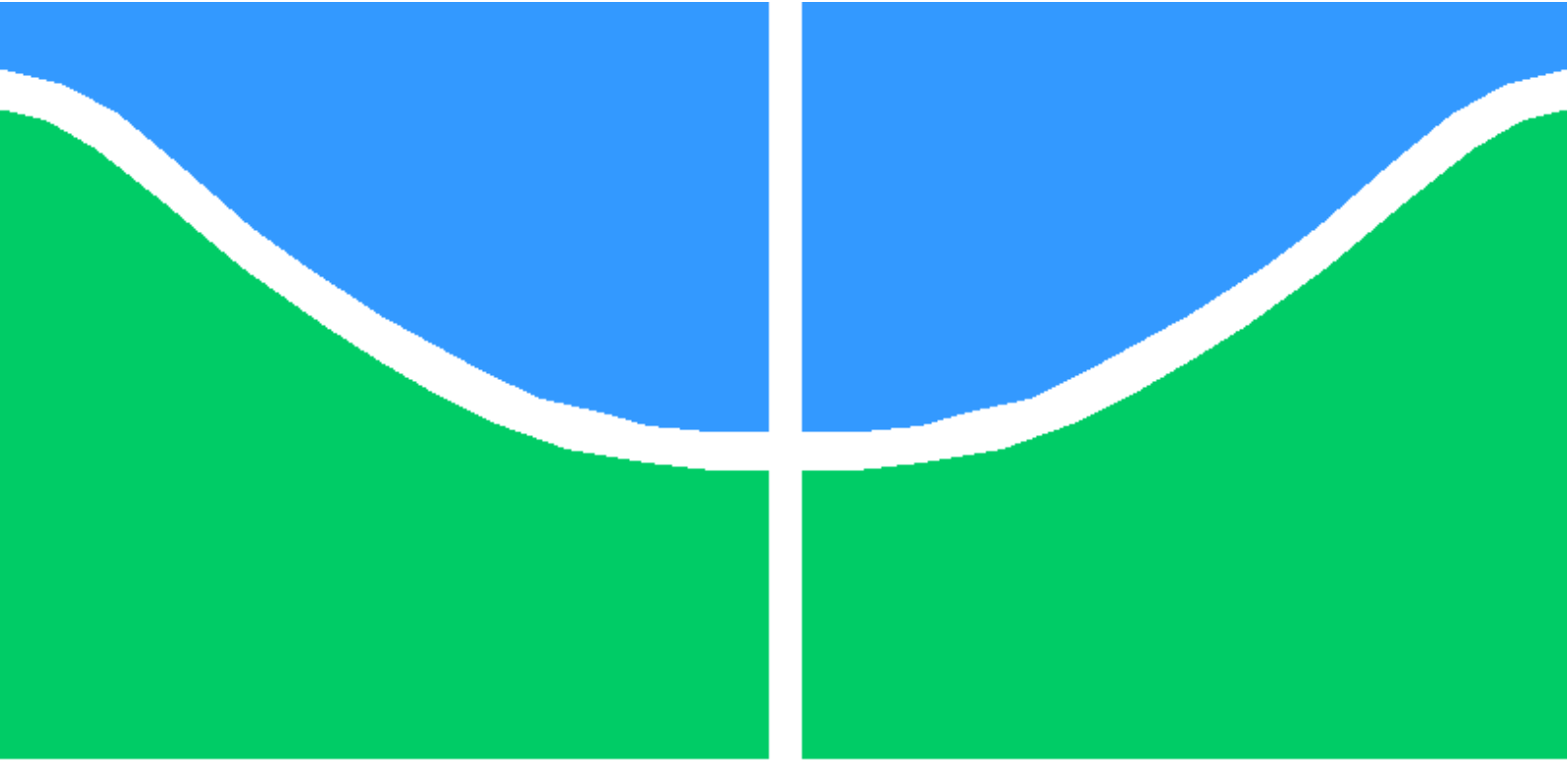


TRABALHO DE GRADUAÇÃO

ANÁLISE DE MÉTODOS DIRETOS E ITERATIVOS PARA RESOLUÇÃO DE PROBLEMAS DE FLUXO DE CARGA

Rafael Zymler

Brasília, julho de 2014

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

TRABALHO DE GRADUAÇÃO

ANÁLISE DE MÉTODOS DIRETOS E ITERATIVOS PARA RESOLUÇÃO DE PROBLEMAS DE FLUXO DE CARGA

Rafael Zymler

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Francisco Damasceno Freitas, Dr., ENE/UnB _____
Orientador

Prof. João Yoshiyuki Ishihara, Dr., ENE/UnB _____
Examinador interno

Prof. Abiezer Amarilia Fernandes, Dr., UniCEUB _____
Examinador externo

Dedicatória

Dedico este trabalho a todos que tornaram possível chegar onde estou, principalmente aos meus pais, que amo incondicionalmente, e professores.

Rafael Zymler

Agradecimentos

A toda a minha família, principalmente ao meu pai Benjamin, à minha mãe Lenir e à minha irmã Evelyn, que me apoiaram desde pequeno e me incentivaram a estudar e buscar meus sonhos.
À minha namorada, Luisa, que amo e está sempre ao meu lado me apoiando.
Aos meus professores, que tornaram possível alcançar meus objetivos.
Ao professor Damasceno, que me orientou e ajudou a concluir este trabalho.

Rafael Zymler

RESUMO

Os estudos dos sistemas elétricos de potência envolvem problemas de fluxo de carga. Esses tipos de problema resultam em equações não-lineares e exigem a resolução sistemas lineares esparsos de grande porte como parte de processos iterativos. Nesse trabalho, foram realizados estudos de diferentes formulações matemáticas para a resolução do problema de fluxo de carga pelo método de Newton-Raphson, utilizando coordenadas polares, coordenadas retangulares e modelagem por injeção de corrente. Métodos diretos e iterativos foram apresentados e estudados em aplicações de sistemas lineares esparsos de grande porte para verificar o desempenho computacional. Foram realizados estudos de sensibilidade aos parâmetros dos métodos utilizados. Um novo método direto para resolução de sistemas lineares baseado na expansão linear da série de Taylor foi apresentado e testado. As simulações foram realizadas no MATPOWER utilizando principalmente o sistema elétricos de 3375 barras.

Palavras chave: Problemas de Fluxo de Carga, Método de Newton-Raphson, Métodos Diretos, Métodos Iterativos, Pré-condicionador, Fatoração

ABSTRACT

Electric power system studies involve power flow problem solutions. This type of problem requires the calculation of large and sparse linear systems as part of a nonlinear iterative process. This report presents different strategies to solve linear systems applied to the power flow problem. The system mathematical representation takes into account bus voltages in the form of polar and rectangular coordinates, as well as current injection model. Direct and iterative methods for solving large and sparse linear systems are presented and studied. The performance of each method is evaluated for several numerical experiments, including sensitivity of parameters. A new direct method for solving linear systems based on the linear expansion of Taylor's series was presented and tested. The simulations were performed in the software MATPOWER, by using mainly a 3375-bus system.

Keywords: Power Flow Problems, Newton-Raphson Method, Direct Methods, Iterative Methods, Preconditioner, Factorization

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E SIGLAS

Símbolos

A	matriz coeficientes
b	vetor dos valores independentes
x	vetor das incógnitas
n	dimensão de uma matriz
L	matriz triangular inferior
U	matriz triangular superior
K_m	subespaço de Krylov
r	resíduo
H_v	matriz de Hessemberg
ν	iteração
$y^{(\nu)}$	solução do problema de norma mínima
$\hat{e}_1$	vetor unitário canônico de dimensão ν
λ	matriz de autovalores
I	matriz identidade
D	matriz diagonal dos autovalores de A
α	fator diferença soma de linhas
β	fator diferença soma de colunas
τ	tolerância
nz	elementos não nulos (*non-zero*)
$x^{(0)}$	vetor dos valores iniciais estimados
y_{km}	admitância entre as barras k e m
Z_{km}	impedância entre as barras k e m
R_{km}	resistência entre as barras k e m
X_{km}	reatância entre as barras k e m
b_k^{Sh}	susceptância shunt conectada à barra k
b_{km}^{Sh}	susceptância shunt entre as barras k e m
G	Matriz de Condutância
B	Matriz de Susceptância
j	número complexo ou índice de contagem
i	índice de contagem
V_k	módulo da tensão na barra k
θ_k	fase da tensão na barra k
θ_{km}	diferença das fases das tensões das barras k e m
$\vec{E}_k$	tensão na barra k
$\vec{I}_k$	corrente injetada na barra k

$\vec{I_{km}}$	corrente entre as barras k e m
$\vec{S_k}$	potência complexa injetada na barra k
$\vec{P_k}$	potência ativa injetada na barra k
$\vec{Q_k}$	potência reativa injetada na barra k
$\vec{S_{km}}$	fluxo de potência complexa entre as barras k e m
$\vec{P_{km}}$	fluxo de potência ativa entre as barras k e m
$\vec{Q_{km}}$	fluxo de potência reativa entre as barras k e m
Ω_k	conjunto de barras *m* adjacentes à barra k
K_k	conjunto de barras *m* adjacentes à barra k, incluindo a barra k
Y_{bus}	matriz de admitâncias
$f(x)$	função f(x)
$\frac{\partial f}{\partial x}$	derivada parcial de f(x)
Δx	incrementos de x
J	matriz jacobiana
ΔP_k	resíduo de potência ativa da barra k
ΔQ_k	resíduo de potência reativa da barra k
P^{esp}	potência ativa especificada
Q^{esp}	potência reativa especificada
ΔV	incremento na tensão
$\Delta\theta$	incremento na fase
F	vetor dos valores independentes
M	pré-condicionador
T	matriz truncada de A
$\mathbf{J_P}$	matriz jacobiana por coordenadas polares
H,N,M,L	submatrizes de $\mathbf{J_P}$
V	vetor da tensão
V_r	parte real do vetor V
V_{imag}	parte imaginária do vetor V
ΔV_k	resíduo da tensão na barra k
$\mathbf{J_R}$	matriz jacobiana por coordenadas retangulares
I'_{r_k},I'_{imag_k}	resíduos de corrente na barra k
$\Delta I'_{r_k}$,$\Delta I'_{imag_k}$	variações dos resíduos de corrente na barra k
$\mathbf{J_I}$	matriz jacobiana por injeção de corrente
B_{prim},G_{prim}	submatrizes superiores de $\mathbf{J_I}$
B_{sec},G_{sec}	submatrizes inferiores de $\mathbf{J_I}$
$\mathbf{J_{I(reord)}}$	matriz jacobiana por injeção de corrente reordenada

Siglas

AMD	*Approximate Minimum Degree*
BiCG	*Bi-Conjugate Gradient*
BiCGSTAB	*Bi-Conjugate Gradient Stabilized*
CG	*Conjugate Gradient*
GMRES	*Generalized Minimum Residual*
ILU	fatoração LU incompleta
LU	fatoração LU completa
MNR	Método de Newton-Raphson
ONS	Operador Nacional do Sistema
PFC	Problema de Fluxo de Carga
PQ	barra de carga
PV	barra de geração
RCM	*Reverse* Cuthill-McKee
SEP	Sistema Elétrico de Potência
SIN	Sistema Interligado Nacional

Capítulo 1

Introdução

1.1 Introdução Geral

Sistemas elétricos de potência (SEP) durante grande parte do tempo operam em regime permanente. Em simulações digitais, a determinação do ponto de operação requer o cálculo da solução de problemas, como fluxo de carga. Problemas desta natureza são do tipo não-linear. Consequentemente, uma solução deve ser calculada iterativamente, uma vez que se trata de problema em que a tentativa de busca por uma solução analítica é inviável [1].

No problema de fluxo de carga (PFC) é utilizado o método de Newton-Raphson (MNR) como técnica para se determinar a solução dos sistemas de equações não-lineares [1]. Há diversas variantes para aplicação do método de Newton-Raphson ao PFC. Entretanto, neste trabalho será considerada somente a abordagem relativa ao método tradicional.

Por meio do PFC determina-se o estado do sistema, caracterizado pelas magnitudes e ângulos de fase das tensões de barras. No entanto, abordagens que tratam o problema na forma de outras variáveis, como parte real e imaginária da tensão são possíveis.

Apesar de o problema ser do tipo não-linear e por consequência demandar a utilização do processo iterativo de Newton-Raphson, surge a necessidade de a cada iteração ser resolvido um sistema de equações lineares. Dependendo da modelagem adotada, o sistema linear poderá ser apresentado de diversas formas. Evidentemente, a solução final do problema não-linear de interesse deve ser a mesma, independentemente da abordagem adotada. Entretanto, uma ou outra pode ser numericamente mais apropriada.

1.2 Motivação

Existem diferentes formulações matemáticas para resolução do PFC [2]. Genericamente a parte mais custosa do ponto de vista computacional resulta na resolução de subproblemas lineares do tipo $Ax = b$, em que a matriz A, geralmente, é esparsa e associada a sistema de grande porte. Vários métodos podem ser utilizados para resolução desse problema. Recentemente, a aplicação de técnicas iterativas tem sido alvo de diversas pesquisas como em [3], [4], [5], [6] e [7].

Há vários métodos iterativos que encontram aplicações em PFC. Os métodos que apresentam melhor desempenho computacional são os iterativos que utilizam o subespaço de Krylov, alvos de diversos estudos como em [3], [4], [5], [6] e [7]. Considerando que o PFC pode ser formulado de diversas formas, torna-se atrativo investigar o desempenho destes métodos frente às diversas abordagens possíveis dos subproblemas lineares.

1.3 Objetivo

No presente trabalho, o objetivo principal é analisar e comparar o desempenho computacional das diferentes formas de resolução do PFC através do método de Newton-Raphson. Em particular, busca-se avaliar os resultados levando-se em conta o desempenho dos métodos para resolução de sistemas lineares. Serão desenvolvidas simulações computacionais no aplicativo MATPOWER [8] em diferentes sistemas teste para determinar as abordagens com melhor desempenho. Adicionalmente, será estudada a sensibilidade dos métodos iterativos em relação a diferentes parâmetros de controle e será testado um novo método para resolução do PFC pelo MNR utilizando expansão linear da jacobiana em série de Taylor.

1.4 Apresentação do manuscrito

No Capítulo 2 é feita uma revisão bibliográfica sobre os métodos diretos e iterativos, incluindo os conceitos de pré-condicionadores, reordenamento e outros relacionados. Em seguida, o Capítulo 3 descreve os principais parâmetros dos sistemas de potência, a resolução do PFC utilizando o método de Newton-Raphson e diferentes formulações matemáticas do PFC: coordenadas polares, coordenadas retangulares e por injeção de corrente. A metodologia e os dados das simulações e testes realizados são apresentados no Capítulo 4. Resultados experimentais são discutidos e analisados no Capítulo 5, seguido das conclusões no Capítulo 6. Os anexos contém material complementar.

Capítulo 2

Métodos Diretos e Iterativos

2.1 Introdução

Nesse capítulo, apresenta-se a base teórica que compõe a resolução de sistemas lineares por métodos diretos e iterativos. Em geral, problemas de fluxo de carga resultam em sistemas lineares com matrizes esparsas de grande porte. Para esse casos, os métodos iterativos que utilizam o subespaço de Krylov, como o GMRES e o BiCGStab , apresentam resultados satisfatórios [9] [10] [11] e são o foco de estudo desse trabalho. Adicionalmente, serão apresentados os conceitos de pré-condicionadores, pivoteamento, fatoração LU completa e incompleta e tipos de reordenamento.

2.2 Resolução de Sistemas Lineares

Dado um sistema linear com m equações e n incógnitas, pode-se representá-lo pela equação:

$$Ax = b \tag{2.1}$$

Um caso particular ocorre quando $m = n$. Nesses casos, a matriz A é quadrada, chamada de matriz de coeficientes, b é o vetor de valores independentes e x o vetor das incógnitas. A solução desse problema pode ser representada por x^* dado pela equação (2.2).

$$x^* = A^{-1}b \tag{2.2}$$

Existem dois métodos gerais para a resolução de sistemas lineares: direto e iterativo [7]. Esses métodos serão discutidos a seguir.

2.3 Métodos Diretos

Os métodos diretos permitem que se encontre todas as variáveis do sistema simultaneamente. São capazes de encontrar a solução exata, porém sujeitos a erros de arredondamento. Esses métodos podem ser tornar ineficientes para matrizes esparsas e de grande porte. São exemplos de métodos diretos: regra de Cramer, eliminação de Gauss, eliminação de Gauss-Jordan e decomposição LU [5].

2.3.1 Decomposição LU

Umas das técnicas amplamente utilizadas para a solução de sistemas lineares é baseada na decomposição LU de A [12]. Considere a matriz A do sistema (2.1). A fatoração LU consiste em encontrar uma matriz triangular inferior L e uma matriz triangular superior U de forma que $A = LU$.

O uso da fatoração LU na resolução de sistemas lineares é uma forma de eliminação Gaussiana [12]. Após encontrar as matrizes L e U, a resolução do sistema pode ser feita utilizando:

$$x = U^{-1}L^{-1}b \tag{2.3}$$

Dependendo do tipo de sistema, o cálculo das matrizes L e U completas pode demandar um longo tempo computacional. No MATLAB, a função "lu"fornece as duas matrizes L e U a partir de uma matriz A. Além disso, pode-se utilizar a fatoração LU incompleta (ILU), que consiste na obtenção matrizes L e U aproximadas [13]. Esse método será discutido na Subseção 2.4.3.1.

2.4 Métodos Iterativos

Os métodos iterativos são utilizados para os casos em que pode-se calcular iterativamente e a sua obtenção torna-se mais vantajosa computacionalmente. Esses métodos requerem de uma solução inicial $x^{(0)}$ e geram sequências de soluções aproximadas $x_i^{(\nu)}$ para cada iteração ν, de forma que após um número finito de iterações determina-se uma solução aproximada de x^* baseada em uma tolerância preestabelecida. Os métodos iterativos estão sujeitos a falhas, que são relacionadas diretamente ao condicionamento da matriz A. Portanto, em geral, processos de reordenamento e pré-condicionamento são necessários para tornar os métodos iterativos mais eficientes [6].

Os métodos iterativos podem ser divididos em estacionários e não-estacionários. Os métodos estacionários não alteram a matriz de iteração, enquanto que os não-estacionários utilizam informações das iterações anteriores para obter uma melhor aproximação a cada iteração [14]. A seguir serão apresentados diferentes métodos iterativos.

2.4.1 Métodos Iterativos Estacionários

2.4.1.1 Método Jacobi

O método de Jacobi trata cada uma das equações do sistema linear de maneira isolada [12]. Considerando o sistema $Ax = b$ com n incógnitas $(i = 1, \cdots, n)$, as equações podem ser examinadas como:

$$\sum_{j=1}^{n} a_{i,j} x_j = b_i \tag{2.4}$$

Assumindo-se $a_{i,i} \neq 0$, determina-se a solução de x_i pela equação:

$$x_i = \left(b_i - \sum_{j=1}^{n} a_{i,j} x_j \right) \Big/ a_{i,i} \tag{2.5}$$

o que possibilita realizar o seguinte processo iterativo:

$$x_i^{(v)} = \left(b_i - \sum_{j=1}^{n} a_{i,j} x_j^{(v-1)} \right) \Big/ a_{i,i} \tag{2.6}$$

Percebe-se que a ordem de resolução das equações não importa, de forma que as atualizações podem ser feitas simultaneamente. É um método de simples implementação, porém possui convergência lenta [15].

2.4.1.2 Método de Gauss-Seidel

Diferentemente do método de Jacobi que resolve as equações para encontrar as soluções simultaneamente, o método de Gauss-Seidel examina uma equação por vez e utiliza os valores obtidos nas iterações subsequentes [12], gerando o seguinte processo iterativo:

$$x_i^{(v)} = \left(b_i - \sum_{j<i} a_{i,j} x_j^{(v)} - \sum_{j>i} a_{i,j} x_j^{(v-1)} \right) \Big/ a_{i,i} \tag{2.7}$$

A convergência torna-se mais acelerada do que no método de Jacobi e cada nova iteração de $x_i^{(v)}$ depende da ordem como as equações são analisadas.

2.4.2 Métodos Iterativos Não-Estacionários

Os métodos iterativos não-estacionários se diferenciam dos estacionários por utilizarem matrizes que são alteradas a cada iteração. Eles são baseados em projeções de subespaços de Krylov [15]. Considere os resíduos do sistema (2.1) dados por:

$$r^{(v)} = b - Ax^{(v)} \tag{2.8}$$

Define-se subespaços de Krylov por:

$$K_m(A, r^{(0)}) := span\left\{r^{(0)}, \ldots, A^{\nu-1} r^{(0)}\right\} \tag{2.9}$$

sendo m a dimensão do subespaço, $r^{(0)}$ o resíduo dado por $r^{(0)} = b - Ax^{(0)}$ e $span\left\{r^{(0)}, \ldots, A^{\nu-1} r^{(0)}\right\}$ a base geradora do subespaço vetorial K_m. Os métodos não-estacionários utilizam diferentes algoritmos para a construção do subespaço de Krylov e buscam a solução dentro do subespaço utilizando diferentes condições de busca. A Tabela 2.1 mostra algumas características dos métodos iterativos não-estacionários que serão discutidos a seguir.

Tabela 2.1: Métodos Iterativos Não-Estacionários

Método Iterativo	Tipo de Matriz de Coeficientes	Algoritmo de Construção do Subespaço de Krylov	Condições de Busca da Solução
CG	Simétrica definida positiva	Lanczos	Ritz-Galerkin
BiCG	Não-Simétrica	Bi-Lanczos	Petrov-Galerkin
BiCGStab	Não-Simétrica	Bi-Lanczos Modificado	Híbrido
GMRES	Não-Simétrica	Arnoldi	Norma mínima residual

2.4.2.1 Resíduo Mínimo Generalizado

No método resíduo mínimo generalizado (*generalized minimum residual* - GMRES) estima-se a solução aproximada de $x^{(\nu)}$, ou seja, x para a ν-ésima iteração, baseado na condição da norma residual mínima. Para isso, forma-se uma base ortogonal de vetores baseada no subespaço de Krylov (2.9), aplicando o método de ortogonalização denominado método de Arnoldi [6] [7]. Sob essa base, busca-se $x^{(\nu)}$ para que sua correspondente norma euclidiana do resíduo seja mínima.

A solução aproximada $x^{(\nu)}$ para a ν-ésima iteração pode ser expressa por:

$$x^{(\nu)} = x^{(0)} + V y^{(\nu)} \tag{2.10}$$

sendo $x^{(0)}$ uma estimativa inicial, V uma matriz cujas colunas são formadas pelos vetores $v^{(\nu)}$ gerados pelo método de Arnoldi e $y^{(\nu)}$ a solução do problema de norma mínima residual dado por:

$$H_\nu y^{(\nu)} = \left\| r^{(0)} \right\|_2 \hat{e}_1 \tag{2.11}$$

Em (2.11), H_ν é a matriz de Hessenberg superior de dimensões $(\nu+1) \times (\nu)$, $r^{(0)}$ é o resíduo inicial dado por $r^{(0)} = b - Ax^{(0)}$ e $\hat{e}_1$ é o vetor unitário canônico de dimensão ν dado por $\hat{e}_1 = [1 \quad 0 \quad 0 \quad \cdots \quad 0]^T$. A matriz de Hessenberg é construída utilizando os vetores calculados durante o processo de Arnoldi e pode ser expressa por:

$$H_v = \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1v} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2v} \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & h_{v,v-1} & h_{vv} \\ 0 & \cdots & \cdots & 0 & h_{v+1,v} \end{bmatrix} \tag{2.12}$$

Por fim, após determinar-se a matriz V e o vetor $y^{(v)}$, calcula-se a aproximação $x^{(v)}$ dada por (2.10) e o resíduo $r^{(v)}$ dado por (2.8). O processo iterativo é repetido para cada iteração v até a condição de convergência ser satisfeita.

Um procedimento utilizado para reduzir o número de operações de ponto flutuante e o armazenamento é a reinicialização, que considera um número fixo de elementos v_r na base de Krylov. Após um determinado número v_r de iterações, é gerado um novo subespaço vetorial de Krylov com v_r elementos na base. Para v_r pequenos, o método necessita de mais iterações, porém menor carga computacional é requerida a cada iteração. Já para v_r elevados, o método GMRES fica mais robusto e a tendência é apresentar convergência em menos iterações, mas cada iteração demanda mais processamento [7] [5].

O Algoritmo 2.1 apresentado a seguir, presente no livro do Golub [12], permite a solução de um sistema linear pelo método GMRES.

Algoritmo 2.1 GMRES [12]: Se $A \in \mathbb{R}^{n\times n}$ é uma matriz não singular, $b \in \mathbb{R}^n$ e $x^{(0)} \in \mathbb{R}^n$ é uma suposição inicial $(Ax^{(0)} \approx b)$, então o algoritmo a seguir computa $x \in \mathbb{R}^n$ para $Ax = b$

1: $r^{(0)} = b - Ax^{(0)}$
2: $h_{10} = \|r^{(0)}\|_2$
3: $v = 0$
4: **while** $(h_{v+1,v} > 0)$ **do**
5: $\quad q_{v+1} = r^{(v)}/h_{v+1,v}$, onde q_{v+1} é uma componente da matriz Q_v gerada pelo método de Arnoldi
6: $\quad v = v + 1$
7: $\quad r^{(v)} = Aq_v$
8: $\quad$ **for** $(i = 1 : v)$ **do**
9: $\quad\quad h_{iv} = q_i^T r^{(v)}$
10: $\quad\quad r^{(v)} = r^{(v)} - h_{iv}q_i$
11: $\quad$ **end for**
12: $\quad h_{v+1,v} = \|r^{(v)}\|_2$
13: $\quad x^{(v)} = x^{(0)} + Q_v y^{(v)}$ where $\|h_{10}\hat{e}_1 - H_v y^{(v)}\|_2 =$min
14: **end while**

No último passo do algoritmo, é necessário resolver o problema de mínimos quadrados dado por:

$$\|h_{10}\hat{e}_1 - H_v y^{(v)}\|_2 = min \tag{2.13}$$

Pode-se encontrar o valor de $y^{(v)}$ através de:

$$y^{(v)} = (H_v^T H_v)^{-1} H_v^T h_{10} \hat{e}_1 \tag{2.14}$$

Por fim, calcula-se o valor de $x^{(v)}$:

$$x^{(v)} = x^{(0)} + Q_v y^{(v)} \tag{2.15}$$

2.4.2.2 Gradiente Conjugado

O método gradiente conjugado (*conjugate gradient* - CG) é eficiente para sistemas simétricos e positivos definidos. Por essa técnica, são geradas sequências de vetores com sucessivas aproximações da solução e resíduos das iterações. As atualizações são feitas buscando encontrar o mínimo resíduo. A cada iteração são realizadas dois produtos internos para satisfazer condições de ortogonalidade [14]. Para um sistema positivamente definido, essas condições implicam em uma melhora na aproximação da solução real [15].

O método CG determina o elemento $x^{(v)}$ como $x^{(0)} + span\{r^{(0)}, \ldots, A^{v-1} r^{(0)}\}$, de forma que $(x^{(v)} - x^*)^T A(x^{(v)} - x^*)$ é minimizado, onde x^* é a solução exata para $Ax = b$. O mínimo só é garantido caso A seja uma matriz simétrica e definida positiva [15].

2.4.2.3 Gradiente Bi-Conjugado

Diferentemente do método gradiente conjugado, que utiliza uma sequência ortogonal de resíduos e funciona apenas para sistemas simetricos positivos definidos, o método gradiente bi-conjugado (*bi-conjugate gradient* - BiCG) utiliza duas sequências mutualmente ortogonais de resíduos e é funcional para sistemas não simétricos [14]. Para sistemas simétricos, o método BiCG fornece os mesmos resultados do que o método CG, mas com o dobro do número de iterações. Já para sistema não simétricos, nos casos onde há redução da norma dos resíduos, o sistema apresenta resultados semelhantes ao GMRES. Nesse método, diferentemente do método CG, não há minimização [15].

2.4.2.4 Gradiente Bi-Conjugado Estabilizado

O método gradiente bi-conjugado estabilizado (*bi-conjugate gradient stabilized* - BiCGStab) foi desenvolvido para resolver sistemas lineares não simétricos, evitando os padrões irregulares de convergência do método gradiente conjugado. Interpreta-se o BiCGStab como o resultado do método BiCG e a aplicação repetitiva do método GMRES [15]. Ocorre uma minimização localizada de um vetor de resíduos, o que implica em um comportamento de convergência mais suave. No entanto, se a aplicação do método GMRES estagnar, o espaço de Krylov não é expandido e o método BiCGStab não converge.

O Algoritmo 2.2 apresentado a seguir, permite a solução de um sistema linear pelo método BiCGStab [15].

Algoritmo 2.2 BiCGStab: Se $A \in \mathbb{R}^{n\times n}$ é uma matriz não singular, $b \in \mathbb{R}^n$ e $x^{(0)} \in \mathbb{R}^n$ é uma suposição inicial ($Ax^{(0)} \approx b$), então o algoritmo a seguir computa $x \in \mathbb{R}^n$ para $Ax = b$

1: $r^{(0)} = b - Ax^{(0)}$
2: $\widetilde{r} = r^{(0)}$
3: **for** $(i = 1, 2, \cdots)$ **do**
4: $\quad \rho_{i-1} = \widetilde{r}^T r^{(i-1)}$
5: $\quad$ **if** $\rho_{i-1} = 0$ **then**
6: $\quad\quad$ O método falhou
7: $\quad$ **end if**
8: $\quad$ **if** $i = 1$ **then**
9: $\quad\quad p^{(i)} = r^{(i=1)}$
10: $\quad$ **else**
11: $\quad\quad \beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
12: $\quad\quad p^{(i)} = r^{(i-1)} + \beta_{i-1}(p^{(i-1)} - \omega_{i-1}\upsilon^{(i-1)})$
13: $\quad$ **end if**
14: $\quad \upsilon^{(i)} = Ap^{(i)}$
15: $\quad \alpha_i = \rho_{i-1}/\widetilde{r}^T\upsilon^{(i)}$
16: $\quad s = r^{(i-1)} - \alpha_i\upsilon^{(i)}$
17: $\quad$ Checar a norma de s; se for pequena suficiente, definir $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$ e parar
18: $\quad t = As$
19: $\quad \omega_i = t^T s/t^T t$
20: $\quad x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)} + \omega_i s$
21: $\quad r^{(i)} = s - \omega_i t$
22: $\quad$ Checar a convergência; continuar se necessário
23: $\quad$ Para continuar é necessário que $\omega_i \neq 0$
24: **end for**

2.4.3 Pré-Condicionadores

Para uma matriz A, o cálculo de seus autovalores e autovetores é determinado pela equação (2.16).

$$(A - \lambda I)x = 0 \tag{2.16}$$

onde λ é o autovalor de A, x o seu respectivo autovetor à direita e I a matriz identidade de ordem n [16]. O autovetor x^L, à esquerda de A e associado a λ, pode ser obtido por (2.17).

$$x^L(A - \lambda I) = 0 \tag{2.17}$$

Calculando-se todos os autovalores de A e montando-se uma matriz M cujas colunas são os autovetores à direita, calcula-se:

$$D = M^{-1}AM \tag{2.18}$$

onde D é uma matriz diagonal composta pelos autovalores de A. A matriz M^{-1} pode ser formada pelos respectivos autovetores à esquerda de A.

Através da equação (2.18), qualquer matriz quadrada pode ser escrita como um produto entre seus autovalores e autovetores. Para um processo iterativo, a taxa de convergência para cada autovetor depende da magnitude de seu respectivo autovalor, de maneira que a convergência do autovetor é mais rápida quanto maior for a magnitude do seu respectivo autovalor. Dessa forma, a convergência de um método iterativo dependendo da magnitude do menor autovalor do sistema [16].

Caso os autovalores de um sistema tenham magnitudes muito diferentes, o número de iterações necessárias para encontrar a solução aumenta consideravelmente, propagando mais erros de arredondamento.

Os pré-condicionadores são utilizados para modificar a matriz A de forma que seus autovalores tenham magnitudes similares, diminuindo a quantidade de iterações necessárias durante o processo iterativo. Por fim, os verdadeiros autovalores são recuperados e o sistema original é obtido após a convergência [16].

Existem dois tipos de pré-condicionamento: aproximação esparsa da matriz inversa, que busca $M \approx A^{-1}$ e matriz decomposta, que busca $M \approx A$ [7]. A seguir são apresentadas diferentes formas de se realizar o pré-condicionamento para ambos os tipos:

A - Pré-condicionadores de aproximação esparsa da matriz inversa: $M \approx A^{-1}$

I - Lado esquerdo: $MAx = Mb$

II - Lado direito: $AMy = b \quad \therefore \quad x = My$

III - Ambos os lados: $M_2AM_1y = M_2b \quad \therefore \quad x = M_1y$

B - Pré-condicionadores de matriz decomposta: $M \approx A$

I - Lado esquerdo: $M^{-1}Ax = M^{-1}b$

II - Lado direito: $AM^{-1}y = b \quad \therefore \quad x = M^{-1}y$

III - Ambos os lados: $M_2^{-1}AM_1^{-1}y = M_2^{-1}b \quad \therefore \quad x = M_1^{-1}y$

Neste trabalho, é utilizado pré-condicionadores de matriz decomposta pelo lado esquerdo, por apresentar resultados satisfatórios em outros trabalhos como em [3], [4] e [7]. A matriz M é decomposta no produto de fatores L e U, em geral, calculados por fatoração ILU, que será apresentada a seguir. Os pré-condicionadores podem melhorar a eficiência e a robustez de técnicas iterativas, como o GMRES e o BiCGStab. A diferença do tempo computacional para a convergência desses métodos muda drasticamente com a utilização de um pré-condicionador adequado. Dessa forma, a busca por um pré-condicionador para cada problema específico é alvo de diversos estudos [14].

2.4.3.1 Fatoração LU incompleta (ILU)

A fatoração LU consiste em decompor uma matriz quadrada A em uma matriz triangular superior U e uma matriz triangular inferior L, de forma que $A = LU$. No entanto, a construção de um pré-condicionador para uma matriz esparsa A baseada na fatoração LU completa resultaria em um preenchimento de muitos elementos nulos de A em M. Dessa forma, a fatoração ILU busca $M = \widetilde{L}\widetilde{U} \simeq A$ [5] [13].

Durante o processo de fatoração, alguns elementos não-nulos que iriam substituir elementos inicialmente nulos são ignorados. O método ILU está sujeito a condições de interrupção devido a problemas com os elementos pivôs. Em casos onde os elementos pivôs são nulos ou negativos, deve-se utilizar estratégias como substituição por um número positivo arbitrário ou algum tipo de reordenamento [7].

Existem diferentes métodos de ILU. O ILU(0) não admite preenchimentos em nenhuma etapa da fatoração, de forma que $\widetilde{L}$ e $\widetilde{U}$ possuem o mesmo padrão de esparsidade e a mesma quantidade de elementos não nulos de A. O ILU(p) admite preenchimento até o nível p, de forma que quando $p = n - 1$, tem-se a fatoração LU completa.

No entanto, esses métodos consideram apenas a posição dos elementos, sem levar em consideração o valor numérico dos elementos dos fatores triangulares. Dessa forma, esse tipo de pré-condicionador pode provocar perda da robustez e velocidade de convergência para alguns métodos iterativos, como o GMRES.

Outro método utilizando é o ILUTP(τ,ρ), onde ρ limita o número máximo de elementos permitidos em cada linha dos fatores L e U e τ é a tolerância *droptol* usada para descartar elementos considerados pequenos. Dessa forma, ρ controla o uso da memória e τ reduz o tempo computacional [6].

O MATLAB possui a função "ilu", que permite a utilização de diferentes tipos de fatoração ILU. Alguns parâmetros utilizados são:

I. *type* - Define o tipo de fatoração utilizado. Os tipos podem ser:
   - *nofill* - Fatoração ILU(0), com nível 0 de preenchimento.
   - *crout* - Fatoração ILUC, conhecida como versão Crout da fatoração ILU.
   - *ilutp* - Fatoração ILUTP, com pivoteamento e tolerância.

II. *droptol* - Tolerância de redução da fatoração ILU: é um valor escalar não negativo. Seu valor padrão é 0. Considere $A_{col}(j)$ como o vetor com os componentes da coluna j da matriz A. Os valores não

nulos da matriz U a serem preservados devem satisfazer:

$$|U(i,j)| \geq droptol|A_{col}(j)| \tag{2.19}$$

Apenas os elementos da diagonal são mantidos independente do valor de *droptol*. Para a matriz L, os valores não nulos devem satisfazer:

$$|L(i,j)| \geq droptol|A_{col}(j)|/U(j,j) \tag{2.20}$$

III. *milu* - fatoração ILU modificada. Os valores para '*milu*' são dados por:

- *row* - Produz a fatoração ILU modificada pela soma das linhas. As entradas das novas colunas formadas dos fatores são subtraídas da diagonal da matriz triangular superior U, preservando a soma das linhas. Ou seja, $Ae = LUe$, onde e é um vetor contendo apenas 1.
- *col* - Produz a fatoração ILU modificada pela soma das colunas. As entradas das novas colunas formadas dos fatores são subtraídas da diagonal da matriz triangular superior U, preservando a soma das colunas. Ou seja, $e^T A = e^T LU$.
- *off* - Não produz fatoração ILU modificada.

IV. *udiag* - Se udiag for igual a 1, qualquer elemento nulo na diagonal principal da matriz triangular superior U é substituido pela tolerância de queda '*droptol*'. O valor padrão é udiag=0.

V. *thresh* - Tolerância de pivoteamento entre 0 e 1. O valor tresh=0 força o pivoteamento pela diagonal, enquanto que tresh=1 seleciona o maior valor da coluna como pivô. O valor padrão é thresh=1.

Utilizando os parâmetros type='ilutp', milu='off', droptol=0 e thresh=1, obtem-se uma fatoração próxima da LU completa nos tempos computacionais e nas matrizes L e U geradas. Alterando thresh=0, obtem-se uma fatoração com os valores das matrizes triangulares L e U próximas das matrizes geradas na fatoração LU completa, porém o tempo computacional demandado para sua construção é consideravelmente inferior. Esses resultados serão discutidos no Capítulo 5.

2.4.4 Pivoteamento

Elementos pivôs são elementos de uma matriz primeiramente selecionados por um algoritmo para realizar determinados cálculos. Geralmente, é necessário que esse elemento seja não-nulo. Diversos algoritmos para a solução de sistemas lineares estão sujeitos a falhas de acordo com a distribuição dos elementos da matriz de coeficientes A. Um desses problemas é a presença de elementos nulos na diagonal principal. Por exemplo, a construção da matriz triangular superior (ou inferior) presente nos métodos LU e ILU se torna inviável pois não é possível realizar a substituição sucessiva (ou reversa) [16] [5].

Outro problema é a propagação do erro de arredondamento, que pode afetar a precisão do resultado final e causar instabilidade numérica, possivelmente impedindo a convergência. Algumas soluções para este problema são o pivoteamento parcial ou o reordenamento da matriz A para eliminar elementos nulos na diagonal principal [16].

2.4.5 Reordenamento RCM e AMD

O reordenamento consiste na permutação de linhas, colunas ou ambas, com objetivo de melhorar a estabilidade durante a fatoração triangular LU. No caso dos métodos iterativos, o reordenamento é aplicando antes da execução do método, alterando estruturalmente a matriz de coeficientes A, mas seus benefícios só podem ser observados com a utilização de técnicas de pré-condicionamento. Sem o uso de pré-condicionadores, o reordenamento não modifica as propriedades espectrais da matriz A, mantendo os mesmos autovalores [6].

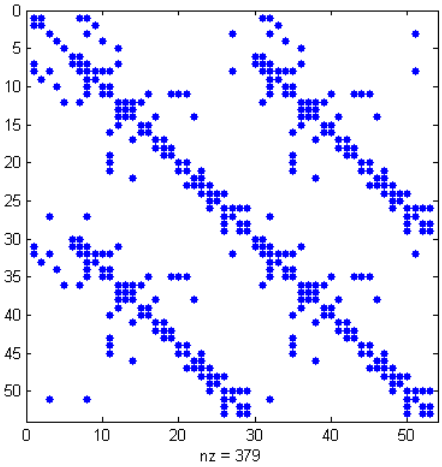


Figura 2.1: Matriz esparsa A de ordem 53x53

O uso do reordenamento permite melhorar a qualidade dos pré-condicionadores e reduz o tempo de sua construção, além de reduzir a quantidade de iterações necessária para a convergência dos métodos iterativos. Sua influência é indireta, pois depende do uso do pré-condicionador [6]. Alguns métodos de reordenamento são o *Reverse Cuthill-McKee* (RCM) e o *Approximate Minimum Degree* (AMD) [5].

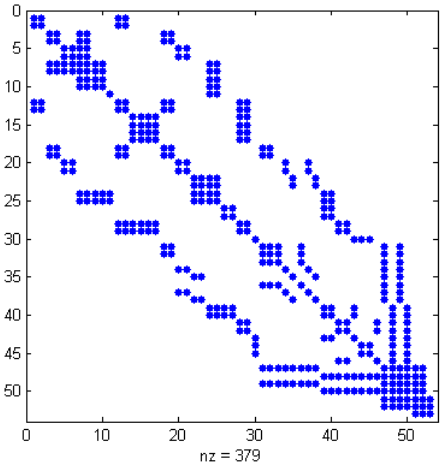


Figura 2.2: Reordenamento RCM da matriz A

Proposto por Cuthill e McKee (1969), o RCM é um método de reordenamento aplicado a matrizes

esparsas. Inicialmente definido como método Cuthill-McKee, foi verificado que invertendo a ordem de numeração do algoritmo original a solução era melhorada, sendo redefinido como *reverse Cuthill-McKee*.

Para melhor avaliação do método RCM, primeiramente, é necessário definir os conceitos de envelope e largura de banda. Envelope é definido como a soma das distâncias entre o primeiro elemento não-nulo de cada linha da matriz até a diagonal principal. Largura de banda é a maior distância entre o primeiro elemento não-nulo de uma linha até a diagonal principal. O objetivo do método RCM é reduzir o envelope e a largura de banda da matriz. Além disso, também se reduz o número de novos elementos não-nulos da matriz [5]. Esse método é o mais popular entre os métodos iterativos e é aplicado com sucesso em diferentes áreas da engenharia. A Figura 2.1 representa a estrutura da matriz esparsa A de ordem 53x53 formada pelo jacobiano do método polar para o caso de 30 barras, que será apresentada detalhadamente nos Capítulos 3 e 4. A Figura 2.2 ilustra a aplicação do reordenamento RCM na matriz A.

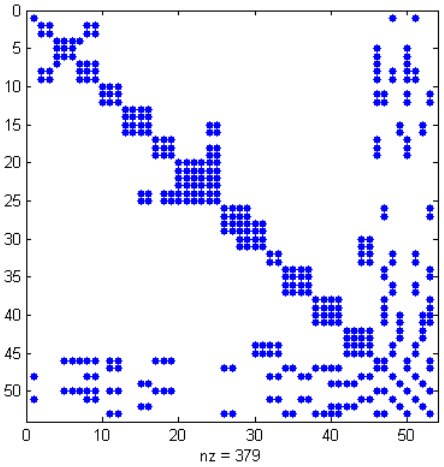


Figura 2.3: Reordenamento AMD da matriz A

O algoritmo do *approximate minimum degree* (AMD) é utilizando para permutar linhas e colunas de uma matriz esparsa simétrica antes da aplicação da decomposição de Cholesky, para reduzir a quantidade de elementos não-nulos no fator Cholesky [5]. O método AMD desloca a coluna com o menor número de elementos não nulos para uma nova posição, de forma a eliminar o vértice de menor grau do grafo da matriz dos coeficientes [5]. Pode-se observar na Figura 2.3 a matriz A (ver Figura 2.1) reordenada pelo método AMD. Os conceitos de pré-condicionador, pivoteamento e reordenamento são essenciais para a aplicação dos métodos iterativos nos problemas de fluxo de carga, abordados no Capítulo 3.

Capítulo 3

Formulações para Resolução do Problema de Fluxo de Carga

3.1 Introdução

As fontes de energia possuem um papel fundamental para o progresso industrial e para melhoria do padrão de vida da população. Atualmente a principal ferramenta utilizada no transporte e conversão de energia é o sistema elétrico de potência. Um sistema elétrico de potência é composto basicamente por geradores, linhas de transmissão e o sistema de distribuição. Os geradores fornecem a potência necessária para abastecer as cargas e as perdas presentes na transmissão. As linhas de transmissão conectam as centrais geradoras com centros de distribuição e constituem as interconexões entre diferentes sistemas de potência [17].

Para realizar o planejamento da expansão de um sistema de potência, deve-se conhecer os efeitos causados pela adição de novas cargas, linhas de transmissão, interligação de sistema e de novas centrais geradoras antes de serem instaladas. É essencial resolver o problema de fluxo de carga (PFC), que consiste na determinação da tensão, corrente, fator de potência, potência ativa, potência reativa e fluxos de potência para diversos pontos de uma rede elétrica. Este capítulo apresentará formas de resolução do PFC a partir de diferentes métodos de resolução de sistemas lineares.

3.2 Fluxo de Carga e matriz Ybus

Nos PFC são conhecidas a quantidade de barras, a forma como elas estão interligadas, os valores das potências das cargas instaladas e alguns parâmetros relacionados aos tipos de barra [18]. Existem três tipos de barras:

- **Barra Swing:** é a barra de referência. Possui os valores do módulo da tensão V e da fase θ fixos. A partir da tensão dessa barra, os parâmetros das outras barras serão calculados. Deve-se fixar somente uma barra swing em cada sistema síncrono, sendo que para o cálculo do fluxo de potência, ela é responsável por absorver os desvios de potência necessários para atender a rede.

- **Barra de Geração:** Também conhecidas como barras PV, são as barras onde os valores de potência ativa P e o módulo da tensão V são fixos. Em geral, são as barras onde estão localizadas as fontes de potência, como usinas geradoras. A fase da tensão θ é desconhecida.
- **Barra de Carga:** Também conhecida como barras PQ, são as barras onde os valores de potência ativa P e reativa Q são fixos. Geralmente são as barras onde estão localizadas as cargas do sistema, como uma residência consumidora. O valor do módulo da tensão V e da fase θ são desconhecidos.

Para a barra swing, é informado o valor do módulo da tensão V e da fase θ de referência. Para as barras de geração PV, são informados os valores da potência ativa injetada P e do módulo da tensão V. Para as barras de carga PQ, são informados os valores da potência ativa injetada P e da potência reativa injetada Q.

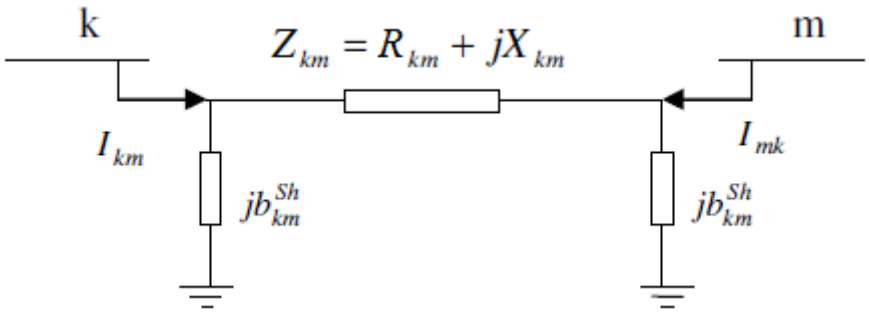


Figura 3.1: Modelo π para linha de transmissão

É necessário conhecer a forma como as barras estão interligadas. Considere a Figura 3.1, na qual duas barras k e m estão interconectadas por uma linha de transmissão, sendo R_{km} a resistência série, X_{km} a reatância série, b_{km}^{Sh} a susceptância shunt da linha e Z_{km} a impedância série. Define-se a admitância série da linha km como:

$$y_{km} = \frac{1}{Z_{km}} = \frac{1}{R_{km} + jX_{km}} \tag{3.1}$$

Utilizando a notação fasorial, podemos definir as tensões das barras k e m como:

$$\vec{E_k} = V_k e^{j\theta_k} \quad e \quad \vec{E_m} = V_m e^{j\theta_m} \tag{3.2}$$

A corrente $\vec{I_{km}}$ que flui da barra k para a m pode ser definida como:

$$\vec{I_{km}} = y_{km}\left(\vec{E_k} - \vec{E_m}\right) + jb_{km}^{Sh}\vec{E_k} \tag{3.3}$$

O fluxo de potência complexa $\vec{S_{km}}$ que flui da barra k para a barra m pode ser definido por:

$$\vec{S_{km}} = \vec{E_k}\vec{I_{km}}^{*} \tag{3.4}$$

Para simplificação, considere uma linha sem perdas ($R_k m = 0$). Dessa forma, o fluxo de potência ativa P_{km} e o fluxo de potência reativa Q_{km} da equação $\vec{S_{km}} = P_{km} + jQ_{km}$ podem ser escritos como:

$$P_{km} = \frac{V_k V_m}{X_{km}} \operatorname{sen}\theta_{km} \tag{3.5}$$

$$Q_{km} = \frac{V_k}{X_{km}} \left(V_k - V_m \cos\theta_{km}\right) \tag{3.6}$$

Os fluxos *Pmk* e *Qmk* são dados por:

$$P_{mk} = \frac{V_k V_m}{X_{km}} \operatorname{sen}\theta_{mk} = -\frac{V_k V_m}{X_{km}} \operatorname{sen}\theta_{km} = -P_{km} \tag{3.7}$$

$$Q_{mk} = \frac{V_m}{X_{km}} \left(V_m - V_k \cos\theta_{mk}\right) \tag{3.8}$$

Agora, considerando a situação para a barra k na Figura 3.2, onde b_k^{Sh} é a susceptância shunt conectada diretamente à barra k, diferentemente da b_{km}^{Sh} que corresponde à susceptância equivalente da linha de transmissão *km*. Pela primeira lei de Kirchhof, calcula-se a corrente injetada I_k como:

$$\vec{I_k} = jb_k^{Sh}\vec{E}_k + \sum_{m\epsilon\Omega_k} \vec{I_{km}} \tag{3.9}$$

onde Ω_k é um conjunto de barras *m* que têm ligação com a barra *k*.

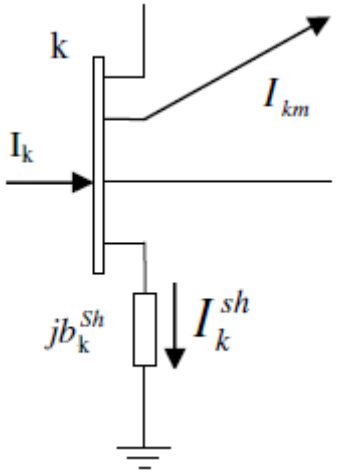


Figura 3.2: Injeção de corrente na barra K

Utilizando a equação (3.3) e reorganizando, obtém-se a formulação para I_k dada pela equação (3.10).

$$\vec{I_k} = \left[jb_k^{Sh} + \sum_{m\epsilon\Omega_k} \left(y_{km} + jb_{km}^{Sh}\right)\right] \vec{E}_k + \sum_{m\epsilon\Omega_k} \left(-y_{km}\right) \vec{E_m} \tag{3.10}$$

A equação (3.10) pode ser reescrita na forma matricial como:

$$\mathbf{I} = \mathbf{Y_{bus}}\mathbf{E} \tag{3.11}$$

Em (3.11), **I** é o vetor de injeções de corrente, cujas componentes são $\vec{I_k}$ e com dimensão $(NB \times 1)$, **E** o vetor de tensões cujas componentes são $\vec{E_k}$, com dimensões $(NB \times 1)$ e $\mathbf{Y_{bus}}$ a matriz admitância nodal, com dimensões $(NB \times NB)$.

Observando a estrutura da equação (3.10), constrói-se a matriz $\mathbf{Y_{bus}}$ como:

$$Y_{km} = -y_{km} \tag{3.12}$$

$$Y_{kk} = jb_k^{Sh} + \sum_{m \epsilon \Omega_k} \left(y_{km} + jb_{km}^{Sh} \right) \tag{3.13}$$

Reescrevendo I_k em termos das equações (3.13) e (3.12), determina-se:

$$\vec{I_k} = Y_{kk}\vec{E_k} + \sum_{m \epsilon \Omega_k} Y_{km}\vec{E_m} \tag{3.14}$$

Define-se o conjunto $K_k = \Omega_k \cup \{k\}$, ou seja, K_k é o conjunto das barras adjacentes à barra k e a própria barra k, ou seja, inclui os elementos shunts conectados diretamente à barra k. Logo, escreve-se a corrente injetada (3.14) como:

$$\vec{I_k} = \sum_{m \epsilon K_k} Y_{km}\vec{E_m} \tag{3.15}$$

Considerando Y_{km} e $\vec{E_m}$ dados por:

$$Y_{km} = G_{km} + jB_{km} \tag{3.16}$$

$$\vec{E_m} = V_m e^{j\theta_m} \tag{3.17}$$

Então, $\vec{I_k}$ é dado por:

$$\vec{I_k} = \sum_{m \epsilon K_k} (G_{km} + jB_{km}) V_m e^{j\theta_m} \tag{3.18}$$

A injeção de potência S_k na barra k é dada por:

$$\vec{S_k} = P_k + jQ_k = \vec{E_k}\vec{I_k}^* \tag{3.19}$$

Considerando $\vec{E_k} = V_k e^{j\theta_k}$ e (3.18), tem-se:

$$\vec{S_k} = V_k \sum_{m \epsilon K_k} V_m (G_{km} - jB_{km}) e^{j\theta_{km}} \tag{3.20}$$

Sendo que $\theta_{km} = \theta_k - \theta_m$.

Finalmente, calcula-se a potência ativa injetada P_k e a potência reativa injetada Q_k por:

$$P_k = Re\left[\vec{S}_k\right] = V_k \sum_{m \epsilon K_k} V_m \left(G_{km}\cos\theta_{km} + B_{km}\,\mathrm{sen}\theta_{km}\right) \tag{3.21}$$

$$Q_k = Im\left[\vec{S}_k\right] = V_k \sum_{m \epsilon K_k} V_m \left(G_{km}\,\mathrm{sen}\theta_{km} - B_{km}\cos\theta_{km}\right) \tag{3.22}$$

3.3 Método de Newton-Raphson

O Método de Newton-Raphson (MNR) é bastante utilizado para a resolução de problemas de fluxo de carga [1]. A expansão da série de Taylor de uma função com duas ou mais variáveis é a base do MNR.

Seja uma função contínua de duas variáveis x_1 e x_2 expressa pelas seguintes equações:

$$\begin{aligned} f_1(x_1,x_2) &= K_1 \\ f_2(x_1,x_2) &= K_2 \end{aligned} \tag{3.23}$$

onde K_1 e K_2 são constantes.

Considere $x_1^{(0)}$ e $x_2^{(0)}$ como as estimativas iniciais da solução dessa equação. Designa-se $\Delta x_1^{(0)}$ e $\Delta x_2^{(0)}$ como os incrementos que devem ser somados a $x_1^{(0)}$ e $x_2^{(0)}$ para buscar a estimativa mais precisa. Logo:

$$\begin{aligned} K_1 &= f_1(x_1,x_2) = f_1(x_1^{(0)} + \Delta x_1^{(0)}, x_2^{(0)} + \Delta x_2^{(0)}) \\ K_2 &= f_2(x_1,x_2) = f_2(x_1^{(0)} + \Delta x_1^{(0)}, x_2^{(0)} + \Delta x_2^{(0)}) \end{aligned} \tag{3.24}$$

Utilizando a expansão da série de Taylor de (3.24) temos:

$$\begin{aligned} K_1 &= f_1(x_1^{(0)}, x_2^{(0)}) + \Delta x_1^{(0)} \left.\frac{\partial f_1}{\partial x_1}\right|_{(0)} + \Delta x_2^{(0)} \left.\frac{\partial f_1}{\partial x_2}\right|_{(0)} + \cdots \\ K_2 &= f_2(x_1^{(0)}, x_2^{(0)}) + \Delta x_1^{(0)} \left.\frac{\partial f_2}{\partial x_1}\right|_{(0)} + \Delta x_2^{(0)} \left.\frac{\partial f_2}{\partial x_2}\right|_{(0)} + \cdots \end{aligned} \tag{3.25}$$

Desprezando as derivadas parciais de ordem maior do que 1, escreve-se (3.25) na forma matricial:

$$\begin{bmatrix} K_1 - f_1(x_1^{(0)}, x_2^{(0)}) \\ K_2 - f_2(x_1^{(0)}, x_2^{(0)}) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \end{bmatrix} \tag{3.26}$$

A matriz quadrada formada pelas derivadas parciais é chamada de Jacobiano J, nesse caso $J^{(0)}$, pois as estimativas iniciais $x_1^{(0)}$ e $x_2^{(0)}$ foram utilizadas para calcular o valor numérico das derivadas parciais.

Pode-se definir os valores de K calculados e K especificados, respectivamente, por:

$$\begin{aligned} K_1^{calc} &= f_1(x_1^{(0)}, x_2^{(0)}) \\ K_2^{calc} &= f_2(x_1^{(0)}, x_2^{(0)}) \end{aligned} \tag{3.27}$$

$$\begin{aligned} K_1^{esp} &= K_1 \\ K_2^{esp} &= K_2 \end{aligned} \tag{3.28}$$

Define-se ΔK_n como a diferença entre K_n^{esp} e $K_n^{(\nu)calc}$:

$$\begin{aligned} \Delta K_1^{(0)} &= K_1^{esp} - K_1^{(0)calc} \\ \Delta K_2^{(0)} &= K_2^{esp} - K_2^{(0)calc} \end{aligned} \tag{3.29}$$

Utilizando (3.29), pode-se escrever (3.26) como:

$$\begin{bmatrix} \Delta K_1^{(0)} \\ \Delta K_2^{(0)} \end{bmatrix} = \left[J^{(0)} \right] \begin{bmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \end{bmatrix} \tag{3.30}$$

Logo, os incrementos $\Delta x_1^{(0)}$ e $\Delta x_2^{(0)}$ podem ser encontrados por:

$$\begin{bmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \end{bmatrix} = \left[J^{(0)} \right]^{-1} \begin{bmatrix} \Delta K_1^{(0)} \\ \Delta K_2^{(0)} \end{bmatrix} \tag{3.31}$$

Após encontrar os incrementos, deve-se definir o novo valor $x_1^{(1)}$ e $x_2^{(1)}$ dados por:

$$\begin{aligned} x_1^{(1)} &= x_1^{(0)} + \Delta x_1^{(0)} \\ x_2^{(1)} &= x_2^{(0)} + \Delta x_2^{(0)} \end{aligned} \tag{3.32}$$

Como foi utilizado um truncamento na expansão da série de Taylor, esses novos valores obtidos não necessariamente determinam a solução correta. Logo, repete-se o processo até que as correções se tornem pequenas o suficiente.

Para aplicar o método de Newton-Raphson na resolução das equações do fluxo de carga, pode-se escolher três diferentes formas para expressar as tensões de barra e as admitâncias de linha, que são a forma polar, retangular e injeção de corrente. Essas três diferentes formas serão abordadas a seguir.

3.3.1 Forma Polar

No método de Newton-Raphson, a forma polar é usada para determinar o módulo e a fase das tensões nas barras do sistema elétrico. Inicialmente, são dados P_k e Q_k nas barras PQ, e P_k e V_k nas barras PV. O

objetivo é calcular os valores de V_k e θ_k nas barras PQ e θ_k nas barras PV. A barra de referência swing não entra na formulação pois o módulo e fase de sua tensão são conhecidos [18]. Logo, trata-se de um sistema de $2NPQ+NPV$ equações algébricas não-lineares e $2NPQ+NPV$ incógnitas, dado pelas equações:

$$P_k^{esp} - V_k \sum_{m \in K_k} V_m \left(G_{km} \cos\theta_{km} + B_{km} \operatorname{sen}\theta_{km}\right) = 0 \tag{3.33}$$

$$Q_k^{esp} - V_k \sum_{m \in K_k} V_m \left(G_{km} \operatorname{sen}\theta_{km} - B_{km} \cos\theta_{km}\right) = 0 \tag{3.34}$$

As incógnitas são dadas por:

$$x = \begin{bmatrix} \theta \\ V \end{bmatrix} \tag{3.35}$$

em que θ é o vetor dos ângulos das tensões com dimensão $NPV+NPQ$ e x, o vetor das magnitudes das tensões com dimensão NPQ.

As expressões (3.33) e (3.34) podem ser reescritas de acordo com os tipos de barra.

Para barras PQ e PV:

$$\Delta P_k = P_k^{esp} - P_k(V,\theta) = 0 \tag{3.36}$$

$$P_k^{esp} = P_k^{gerado} - P_k^{consumido} \tag{3.37}$$

Para barras PQ:

$$\Delta Q_k = Q_k^{esp} - Q_k(V,\theta) = 0 \tag{3.38}$$

$$Q_k^{esp} = Q_k^{gerado} - Q_k^{consumido} \tag{3.39}$$

Utilizando a forma vetorial, tem-se:

$$\Delta P_k = P_k^{esp} - P(V,\theta) \tag{3.40}$$

$$\Delta Q_k = Q_k^{esp} - Q(V,\theta) \tag{3.41}$$

Aplicando a expansão de Taylor, o sistema a ser resolvido determina-se:

$$\begin{bmatrix} \Delta P^{(\nu)} \\ \Delta Q^{(\nu)} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial(\Delta P)}{\partial\theta} & \dfrac{\partial(\Delta P)}{\partial V} \\ \dfrac{\partial(\Delta Q)}{\partial\theta} & \dfrac{\partial(\Delta Q)}{\partial V} \end{bmatrix} \begin{bmatrix} \Delta\theta^{(\nu)} \\ \Delta V^{(\nu)} \end{bmatrix} \tag{3.42}$$

Considerando as equações (3.40) e (3.41), e que P_k^{esp} e Q_k^{esp} são constantes, tem-se:

$$\begin{bmatrix} \Delta P^{(\nu)} \\ \Delta Q^{(\nu)} \end{bmatrix} = - \begin{bmatrix} \dfrac{\partial P}{\partial \theta} & \dfrac{\partial P}{\partial V} \\ \dfrac{\partial Q}{\partial \theta} & \dfrac{\partial Q}{\partial V} \end{bmatrix} \begin{bmatrix} \Delta \theta^{(\nu)} \\ \Delta V^{(\nu)} \end{bmatrix} \tag{3.43}$$

Logo, a matriz jacobiana $\mathbf{J_P}$ de dimensões $2NPQ + NPV$ é dada por:

$$\mathbf{J_P}(x^{(\nu)}) = \begin{bmatrix} \dfrac{\partial P}{\partial \theta} & \dfrac{\partial P}{\partial V} \\ \dfrac{\partial Q}{\partial \theta} & \dfrac{\partial Q}{\partial V} \end{bmatrix} \tag{3.44}$$

A matriz $\mathbf{J_P}$ pode ser reescrita em função de submatrizes como:

$$\mathbf{J_P} = \begin{bmatrix} H & N \\ M & L \end{bmatrix} \tag{3.45}$$

As submatrizes que compõem a matriz jacobiana são representadas por:

$$H = \frac{\partial P}{\partial \theta} \tag{3.46}$$

$$N = \frac{\partial P}{\partial V} \tag{3.47}$$

$$M = \frac{\partial Q}{\partial \theta} \tag{3.48}$$

$$L = \frac{\partial Q}{\partial V} \tag{3.49}$$

As componentes das submatrizes jacobianas H, N, M e L são dadas por:

$$H_{km} = \frac{\partial P_k}{\partial \theta_m} = V_k V_m (G_{km} \operatorname{sen}\theta_{km} - B_{km} \cos\theta_{km}) \tag{3.50}$$

$$H_{kk} = \frac{\partial P_k}{\partial \theta_k} = -Q_k - V_k^2 B_{kk} \tag{3.51}$$

$$N_{km} = \frac{\partial P_k}{\partial V_m} = V_k (G_{km} \cos\theta_{km} + B_{km} \operatorname{sen}\theta_{km}) \tag{3.52}$$

$$N_{kk} = \frac{\partial P_k}{\partial V_k} = V_k^{-1} (P_k + V_k^2 G_{kk}) \tag{3.53}$$

$$M_{km} = \frac{\partial Q_k}{\partial \theta_m} = -V_k V_m (G_{km} \cos\theta_{km} + B_{km} \operatorname{sen}\theta_{km}) \tag{3.54}$$

$$M_{kk} = \frac{\partial Q_k}{\partial \theta_k} = P_k - V_k^2 G_{kk} \tag{3.55}$$

$$L_{km} = \frac{\partial Q_k}{\partial V_m} = V_k(G_{km}\,\mathrm{sen}\theta_{km} - B_{km}\cos\theta_{km}) \tag{3.56}$$

$$L_{kk} = \frac{\partial Q_k}{\partial V_k} = V_k^{-1}(Q_k - V_k^2 B_{kk}) \tag{3.57}$$

Os incrementos para cada iteração ν são calculados por:

$$\begin{bmatrix} \Delta\theta^{(\nu)} \\ \Delta V^{(\nu)} \end{bmatrix} = -\begin{bmatrix} H & N \\ M & L \end{bmatrix}^{-1} \begin{bmatrix} \Delta P^{(\nu)} \\ \Delta Q^{(\nu)} \end{bmatrix} \tag{3.58}$$

Com a formulação acima, pode-se mostrar que o método de Newton aplicado à resolução do PFC segue os seguintes passos:

1. Fazer $\nu = 0$ e definir valores iniciais dos ângulos e das tensões para barras PQ e PV, e as magnitudes das tensões das barras PQ.
2. Calcular $P_k(V^{(\nu)}, \theta^{(\nu)})$ para as barras PQ e PV, e $Q_k(V^{(\nu)}, \theta^{(\nu)})$ para as barras PQ utilizando as equações (3.21) e (3.22). Determinar os resíduos $\Delta P_k^{(\nu)}$ e $\Delta Q_k^{(\nu)}$ pelas equações (3.36) e (3.38).
3. Testar a convergência verificando se $max|\Delta P_k^{(\nu)}| \leq \varepsilon_P$ e $max|\Delta Q_k^{(\nu)}| \leq \varepsilon_Q$. Caso as duas condições sejam satisfeitas, o processo iterativo convergiu para a solução $(V^{(\nu)}, \theta^{(\nu)})$. Caso alguma das condições não seja satisfeita, ir para o passo seguinte.
4. Calcular a matriz jacobiana dada por (3.44) para ν.
5. Encontrar a nova solução $(V^{(\nu+1)}, \theta^{(\nu+1)})$ dada por:

$$\theta^{(\nu+1)} = \theta^{(\nu)} + \Delta\theta^{(\nu)} \tag{3.59}$$

$$V^{(\nu+1)} = V^{(\nu)} + \Delta V^{(\nu)} \tag{3.60}$$

   sendo que os valores de $\Delta\theta^{(\nu)}$ e $\Delta V^{(\nu)}$ são calculados por (3.58).
6. Fazer $\nu + 1 \rightarrow \nu$ e voltar para o passo 2.

A seguir serão apresentadas outras formas de resolução do PFC.

3.3.2 Forma Retangular

Para utilizar a forma retangular do PFC é necessário escrever as equações (3.33) e (3.34) na forma retangular.

Pode-se reescrever as tensões nas barras k e m como:

$$\vec{E_k} = V_k e^{j\theta_k} = V_{r_k} + jV_{imag_k} \tag{3.61}$$

$$\vec{E_m} = V_m e^{j\theta_m} = V_{r_m} + jV_{imag_m} \tag{3.62}$$

As equações de potência ativa e reativa injetadas em uma barra k podem ser escritas na forma retangular como:

$$P_k = \sum_{m \epsilon K_k} [V_{r_k}(G_{km}V_{r_m} - B_{km}V_{imag_m}) + V_{m_k}(G_{km}V_{imag_m} - B_{km}V_{r_m})] \tag{3.63}$$

$$Q_k = \sum_{m \epsilon K_k} [V_{imag_k}(G_{km}V_{r_m} - B_{km}V_{imag_m}) - V_{r_k}(G_{km}V_{imag_m} + B_{km}V_{r_m})] \tag{3.64}$$

Considerando inicialmente apenas as barras PQ, a partir das equações (3.63) e (3.64), reformula-se o problema dado pelas equações (3.33) e (3.34) em $2NPQ$ equações como:

$$P_k^{esp} - \sum_{m \epsilon K_k} [V_{r_k}(G_{km}V_{r_m} - B_{km}V_{imag_m}) + V_{m_k}(G_{km}V_{imag_m} - B_{km}V_{r_m})] = 0 \tag{3.65}$$

$$Q_k^{esp} - \sum_{m \epsilon K_k} [V_{imag_k}(G_{km}V_{r_m} - B_{km}V_{imag_m}) - V_{r_k}(G_{km}V_{imag_m} + B_{km}V_{r_m})] = 0 \tag{3.66}$$

As incógnitas são dadas pelo vetor V_r das partes reais das magnitudes das tensões e pelo vetor V_{imag} das partes imaginárias das magnitudes das tensões para as barras PQ.

Aplicando a expansão de Taylor, obtém-se o seguinte sistema:

$$\left[\begin{array}{c} \Delta P_1 \\ \Delta Q_1 \\ \hline \Delta P_2 \\ \Delta Q_2 \\ \hline \vdots \\ \hline \Delta P_n \\ \Delta Q_n \end{array}\right] = - \left[\begin{array}{cc|cc|c|cc} H_{11} & N_{11} & H_{12} & N_{12} & \cdots & H_{1n} & N_{1n} \\ M_{11} & L_{11} & M_{12} & L_{12} & \cdots & M_{1n} & L_{1n} \\ \hline H_{21} & N_{21} & H_{22} & N_{22} & \cdots & H_{2n} & N_{2n} \\ M_{21} & L_{21} & M_{22} & L_{22} & \cdots & M_{2n} & L_{2n} \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline H_{n1} & N_{n1} & H_{n2} & N_{n2} & \cdots & H_{nn} & N_{nn} \\ M_{n1} & L_{n1} & M_{n2} & L_{n2} & \cdots & M_{nn} & L_{nn} \end{array}\right] \left[\begin{array}{c} \Delta V_{r_1} \\ \Delta V_{imag_1} \\ \hline \Delta V_{r_2} \\ \Delta V_{imag_2} \\ \hline \vdots \\ \hline \Delta V_{r_n} \\ \Delta V_{imag_n} \end{array}\right] \tag{3.67}$$

As componentes da matriz jacobiana podem ser calculadas como:

$$H_{km} = \frac{\partial P_k}{\partial V_{r_m}} = V_{r_k} G_{km} + V_{imag_k} B_{km} \tag{3.68}$$

$$H_{kk} = \frac{\partial P_k}{\partial V_{r_k}} = V_{r_k} G_{kk} + V_{imag_k} B_{kk} + I_{r_k} \tag{3.69}$$

$$N_{km} = \frac{\partial P_k}{\partial V_{imag_m}} = -V_{r_k} B_{km} + V_{imag_k} G_{km} \tag{3.70}$$

$$N_{kk} = \frac{\partial P_k}{\partial V_{imag_k}} = -V_{r_k} B_{kk} + V_{imag_k} G_{kk} + I_{m_k} \tag{3.71}$$

$$M_{km} = \frac{\partial Q_k}{\partial V_{r_m}} = -V_{r_k} B_{km} + V_{imag_k} G_{km} = N_{km} \tag{3.72}$$

$$M_{kk} = \frac{\partial Q_k}{\partial V_{r_k}} = -V_{r_k} B_{kk} + V_{imag_k} G_{kk} - I_{m_k} \tag{3.73}$$

$$L_{km} = \frac{\partial Q_k}{\partial V_{imag_m}} = -V_{r_k} G_{km} - V_{imag_k} B_{km} = -H_{km} \tag{3.74}$$

$$L_{kk} = \frac{\partial Q_k}{\partial V_{imag_k}} = -V_{r_k} G_{kk} - V_{imag_k} B_{kk} + I_{r_k} \tag{3.75}$$

Os valores de I_{r_k} e I_{m_k} utilizados em (3.69), (3.71), (3.73) e (3.75) são encontrados por:

$$\mathbf{I} = \mathbf{YV} = \mathbf{I_r} + j\mathbf{I_{imag}} \tag{3.76}$$

Para considerar uma determinada barra k do tipo PV, é necessário utilizar a equação de restrição de tensão, dada por:

$$V_k^2 = V_{r_k}^2 + V_{imag_k}^2 \tag{3.77}$$

Linearizando (3.77), tem-se:

$$\Delta V_k^2 = 2V_{r_k} \Delta V_{r_k} + 2V_{imag_k} \Delta V_{imag_k} \tag{3.78}$$

onde:

$$\Delta V_k^2 = (V_k^{esp})^2 - (V_k^{calc})^2 \tag{3.79}$$

Dessa forma, é necessário adicionar a equação (3.79) ao sistema (3.67) para considerar as barras PVs. Dessa forma, adicionando a barra k do tipo PV e reorganizando as equações, pode-se reescrever o sistema como:

$$\begin{bmatrix} \Delta P_1 \\ \Delta P_2 \\ \vdots \\ \Delta P_k \\ \vdots \\ \Delta P_n \\ \hline \Delta Q_1 \\ \Delta Q_2 \\ \vdots \\ \Delta V_k^2 \\ \vdots \\ \Delta Q_n \end{bmatrix} = - \left[\begin{array}{cccccc|cccccc} H_{11} & H_{12} & \cdots & H_{1k} & \cdots & H_{1n} & N_{11} & N_{12} & \cdots & N_{1k} & \cdots & N_{1n} \\ H_{21} & H_{22} & \cdots & H_{2k} & \cdots & H_{2n} & N_{21} & N_{22} & \cdots & N_{2k} & \cdots & N_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ H_{k1} & H_{k2} & \cdots & H_{kk} & \cdots & H_{kn} & N_{k1} & N_{k2} & \cdots & N_{kk} & \cdots & N_{kn} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ H_{n1} & H_{n2} & \cdots & H_{nk} & \cdots & H_{nn} & N_{n1} & N_{n2} & \cdots & N_{nk} & \cdots & N_{nn} \\ \hline M_{11} & M_{12} & \cdots & M_{1k} & \cdots & M_{1n} & L_{11} & L_{12} & \cdots & L_{1k} & \cdots & L_{1n} \\ M_{21} & M_{22} & \cdots & M_{2k} & \cdots & M_{2n} & L_{21} & L_{22} & \cdots & L_{2k} & \cdots & L_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2V_{r_k} & \cdots & 0 & 0 & 0 & \cdots & 2V_{imag_k} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ M_{n1} & M_{n2} & \cdots & M_{nk} & \cdots & M_{nn} & L_{n1} & L_{n2} & \cdots & L_{nk} & \cdots & L_{nn} \end{array}\right] \begin{bmatrix} \Delta V_{r_1} \\ \Delta V_{r_2} \\ \vdots \\ \Delta V_{r_k} \\ \vdots \\ \Delta V_{r_n} \\ \hline \Delta V_{imag_1} \\ \Delta V_{imag_2} \\ \vdots \\ \Delta V_{imag_k} \\ \vdots \\ \Delta V_{imag_n} \end{bmatrix} \tag{3.80}$$

Designa-se a matriz jacobiana da equação (3.80) por $\mathbf{J_R}$,de ordem $(2NPQ+2NPV)\times(2NPQ+2NPV)$. Para resolver o PFC deve-se seguir os seguintes passos:

1. Fazer $\nu = 0$ e definir valores iniciais de V_{r_k} e V_{imag_k} para barras PQ e PV, sendo que para as barras PV os valores devem respeitar a equação de restrição de tensão (3.77).
2. Calcular $P_k(V_{r_k}^{(\nu)}, V_{imag_k}^{(\nu)})$ para as barras PQ e PV e $Q_k(V_{r_k}^{(\nu)}, V_{imag_k}^{(\nu)})$ para as barras PQ utilizando as equações (3.63) e (3.64). Determinar os resíduos $\Delta P_k^{(\nu)}$, $\Delta Q_k^{(\nu)}$ e $\Delta V_k^{2(\nu)}$.
3. Testar a convergência verificando se $max|\Delta P_k^{(\nu)}| \leq \varepsilon_P$ e $max|\Delta Q_k^{(\nu)}| \leq \varepsilon_Q$. Caso as duas condições sejam satisfeitas, o processo iterativo convergiu para a solução $(V_{r_k}^{(\nu)}, V_{imag_k}^{(\nu)})$. Caso alguma das condições não seja satisfeita, ir para o passo seguinte.
4. Calcular a matriz jacobiana $\mathbf{J_R}$ para ν.
5. Utilizando (3.80), encontrar os incrementos para a nova solução $(V_{r_k}^{(\nu+1)}, V_{imag_k}^{(\nu+1)})$ dada por:

$$V_r^{(\upsilon+1)} = V_r^{(\upsilon)} + \Delta V_r^{(\upsilon)} \tag{3.81}$$

$$V_{imag}^{(\upsilon+1)} = V_{imag}^{(\upsilon)} + \Delta V_{imag}^{(\upsilon)} \tag{3.82}$$

6. Fazer $\nu + 1 \rightarrow \nu$ e voltar para o passo 2.

3.3.3 Injeção de Corrente

Para utilizar o método de Newton-Raphson por Injeção de Corrente é necessário utilizar um conjunto de equações de injeção de correntes escritas em coordenadas retangulares [2]. Inicialmente, para NPQ barras PQ, existe um conjunto de $2NPQ$ equações dos resíduos de corrente dadas por:

$$I'_{r_k} = \sum_{i \epsilon \Omega_k} (G_{k_i} V_{r_i} - B_{k_i} V_{imag_i}) - \frac{V_{r_k} P_k + V_{imag_k} Q_k}{V_{r_k}^2 + V_{imag_k}^2} \tag{3.83}$$

$$I'_{imag_k} = \sum_{i \epsilon \Omega_k} (G_{k_i} V_{imag_i} + B_{k_i} V_{r_i}) - \frac{V_{imag_k} P_k + V_{r_k} Q_k}{V_{r_k}^2 + V_{imag_k}^2} \tag{3.84}$$

As variações de injeção de corrente são dadas por:

$$\Delta I_{imag_k} = \frac{V_{imag_k} \Delta P_k - V_{r_k} \Delta Q_k}{V_{r_k}^2 + V_{imag_k}^2} \tag{3.85}$$

$$\Delta I_{r_k} = \frac{V_{r_k} \Delta P_k + V_{imag_k} \Delta Q_k}{V_{r_k}^2 + V_{imag_k}^2} \tag{3.86}$$

Sabe-se que:

$$V_k = \sqrt{V_{r_k}^2 + V_{imag_k}^2} \tag{3.87}$$

Logo, as variações podem ser escritas como:

$$\Delta I_{imag_k} = \frac{V_{imag_k} \Delta P_k}{V_k^2} - \frac{V_{r_k} \Delta Q_k}{V_k^2} \tag{3.88}$$

$$\Delta I_{r_k} = \frac{V_{r_k} \Delta P_k}{V_k^2} + \frac{V_{imag_k} \Delta Q_k}{V_k^2} \tag{3.89}$$

Aplicando a expansão de Taylor, chega-se ao seguinte sistema:

$$\left[\begin{array}{c} \Delta I_{imag_1} \\ \Delta I_{r_1} \\ \hline \Delta I_{imag_2} \\ \Delta I_{r_2} \\ \hline \vdots \\ \hline \Delta I_{imag_n} \\ \Delta I_{r_n} \end{array}\right] = - \left[\begin{array}{cc|cc|c|cc} B'_{11} & G'_{11} & B_{12} & G_{12} & \cdots & B_{1n} & G_{1n} \\ G''_{11} & B''_{11} & G_{12} & -B_{12} & \cdots & G_{1n} & -B_{1n} \\ \hline B_{21} & G_{21} & B'_{22} & G'_{22} & \cdots & B_{2n} & G_{2n} \\ G_{21} & -B_{21} & G''_{22} & B''_{22} & \cdots & G_{2n} & -B_{2n} \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline B_{n1} & G_{n1} & B_{n2} & G_{n2} & \cdots & B'_{nn} & G'_{nn} \\ G_{n1} & -B_{n1} & G_{n2} & -B_{n2} & \cdots & G''_{nn} & B''_{nn} \end{array}\right] \left[\begin{array}{c} \Delta V_{r_1} \\ \Delta V_{imag_1} \\ \hline \Delta V_{r_2} \\ \Delta V_{imag_2} \\ \hline \vdots \\ \hline \Delta V_{r_n} \\ \Delta V_{imag_n} \end{array}\right] \tag{3.90}$$

As componentes da matriz jacobiana $\mathbf{J_I}$ são:

$$B'_{kk} = \frac{\partial I'_{imag_k}}{\partial V_{r_k}} = B_{kk} - a_k \tag{3.91}$$

$$G'_{kk} = \frac{\partial I'_{imag_k}}{\partial V_{imag_k}} = G_{kk} - b_k \tag{3.92}$$

$$G''_{kk} = \frac{\partial I'_{r_k}}{\partial V_{r_k}} = G_{kk} - c_k \tag{3.93}$$

$$B''_{kk} = \frac{\partial I'_{r_k}}{\partial V_{imag_k}} = -B_{kk} - d_k \tag{3.94}$$

Onde os parâmetros a_k, b_k, c_k e d_k dependem do modelo de carga adotado [2]. Para o modelo do tipo potência constante, esses valores são dados por:

$$a_k = d_k = \frac{Q_k(V_{r_k}^2 - V_{imag_k}^2) - 2V_{r_k}V_{imag_k}P_k}{V_k^4} \tag{3.95}$$

$$b_k = -c_k = \frac{P_k(V_{r_k}^2 - V_{imag_k}^2) + 2V_{r_k}V_{imag_k}Q_k}{V_k^4} \tag{3.96}$$

Para uma barra PV, o resíduo de potência reativa ΔQ_k é desconhecido, de forma que ele é tratado como uma nova variável de estado [2]. Para isso, adiciona-se uma nova equação, que impõe uma restrição de tensão à barra k do tipo PV, dada por:

$$\Delta V_k = \frac{V_{r_k}}{V_k}\Delta V_{r_k} + \frac{V_{imag_k}}{V_k}\Delta V_{imag_k} \tag{3.97}$$

Para as barras PV, o valor de ΔV_k é dado pela equação:

$$\Delta V_k = V_k^{esp} - V_k^{calc} \tag{3.98}$$

Dessa forma, o novo sistema (3.90) pode ser reordenado e escrito como:

$$\left[\begin{array}{c} \Delta I_{imag_1} \\ \Delta I_{imag_2} \\ \vdots \\ \frac{V_{imag_k}}{V_k^2}\Delta P_k \\ \vdots \\ \Delta I_{imag_n} \\ \hline \Delta I_{r_1} \\ \Delta I_{r_2} \\ \vdots \\ \frac{V_{r_k}}{V_k^2}\Delta P_k \\ \vdots \\ \Delta I_{r_n} \\ \hline \Delta V_k \end{array}\right] = - \left[\begin{array}{cccccc|cccccc|c} B'_{11} & B_{12} & \cdots & B_{1k} & \cdots & B_{1n} & G'_{11} & G_{12} & \cdots & G_{1k} & \cdots & G_{1n} & 0 \\ B_{21} & B'_{22} & \cdots & B_{2k} & \cdots & B_{2n} & G_{21} & G'_{22} & \cdots & G_{2k} & \cdots & G_{2n} & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ B_{k1} & B_{k2} & \cdots & B'_{kk} & \cdots & B_{kn} & G_{k1} & G_{k2} & \cdots & G'_{kk} & \cdots & G_{kn} & \frac{V_{r_k}}{V_k^2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ B_{n1} & B_{n2} & \cdots & B_{nk} & \cdots & B'_{nn} & G_{n1} & G_{n2} & \cdots & G_{nk} & \cdots & G'_{nn} & 0 \\ \hline G''_{11} & G_{12} & \cdots & G_{1k} & \cdots & G_{1n} & B''_{11} & -B_{12} & \cdots & -B_{1k} & \cdots & -B_{1n} & 0 \\ G_{21} & G''_{22} & \cdots & G_{2k} & \cdots & G_{2n} & -B_{21} & B''_{22} & \cdots & -B_{2k} & \cdots & -B_{2n} & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ G_{k1} & G_{k2} & \cdots & G''_{kk} & \cdots & G_{kn} & -B_{k1} & -B_{k2} & \cdots & B''_{kk} & \cdots & -B_{kn} & -\frac{V_{imag_k}}{V_k^2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nk} & \cdots & G''_{nn} & -B_{n1} & -B_{n2} & \cdots & -B_{nk} & \cdots & B''_{nn} & 0 \\ \hline 0 & 0 & \cdots & \frac{V_{r_k}}{V_k} & \cdots & 0 & 0 & 0 & \cdots & \frac{V_{imag_k}}{V_k} & \cdots & 0 & 0 \end{array}\right] \left[\begin{array}{c} \Delta V_{r_1} \\ \Delta V_{r_2} \\ \vdots \\ \Delta V_{r_k} \\ \vdots \\ \Delta V_{r_n} \\ \hline \Delta V_{imag_1} \\ \Delta V_{imag_2} \\ \vdots \\ \Delta V_{imag_k} \\ \vdots \\ \Delta V_{imag_n} \\ \hline \Delta Q_k \end{array}\right] \tag{3.99}$$

Define-se a matriz jacobiana $\mathbf{J_I}$, de ordem $(2NPQ + 3NPV) \times (2NPQ + 3NPV)$, em relação às suas submatrizes como:

$$\mathbf{J_I} = \left[\begin{array}{c|c|c} B_{prim} & G_{prim} & V_{rv} \\ \hline G_{sec} & B_{sec} & V_{imagv} \\ \hline V_{rh} & V_{imagh} & 0 \end{array}\right] \tag{3.100}$$

onde:

$$B_{prim} = \begin{bmatrix} B'_{11} & \cdots & B_{1k} & \cdots & B_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ B_{k1} & \cdots & B'_{kk} & \cdots & B_{kn} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nk} & \cdots & B'_{nn} \end{bmatrix} \tag{3.101}$$

$$G_{prim} = \begin{bmatrix} G'_{11} & \cdots & G_{1k} & \cdots & G_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ G_{k1} & \cdots & G'_{kk} & \cdots & G_{kn} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ G_{n1} & \cdots & G_{nk} & \cdots & G'_{nn} \end{bmatrix} \tag{3.102}$$

$$G_{sec} = \begin{bmatrix} G''_{11} & \cdots & G_{1k} & \cdots & G_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ G_{k1} & \cdots & G''_{kk} & \cdots & G_{kn} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ G_{n1} & \cdots & G_{nk} & \cdots & G''_{nn} \end{bmatrix} \tag{3.103}$$

$$B_{sec} = \begin{bmatrix} B''_{11} & \cdots & -B_{1k} & \cdots & -B_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ -B_{k1} & \cdots & B''_{kk} & \cdots & -B_{kn} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ -B_{n1} & \cdots & -B_{nk} & \cdots & B''_{nn} \end{bmatrix} \tag{3.104}$$

$$V_{rh} = \begin{bmatrix} 0 & \cdots & \dfrac{V_{r_k}}{V_k} & \cdots & 0 \end{bmatrix} \tag{3.105}$$

$$V_{imagh} = \begin{bmatrix} 0 & \cdots & \dfrac{V_{imag_k}}{V_k} & \cdots & 0 \end{bmatrix} \tag{3.106}$$

$$V_{rv} = \begin{bmatrix} 0 \\ \vdots \\ \dfrac{V_{r_k}}{V_k^2} \\ \vdots \\ 0 \end{bmatrix} \tag{3.107}$$

$$V_{imagv} = \begin{bmatrix} 0 \\ \vdots \\ -\dfrac{V_{imag_k}}{V_k^2} \\ \vdots \\ 0 \end{bmatrix} \tag{3.108}$$

A característica importante do método de Injeção de Correntes é que, para as submatizes B_{prim}, G_{prim}, G_{sec} e B_{sec}, os elementos fora da diagonal principal permanecem constantes, enquanto que os elementos da diagonal principal devem ser atualizados a cada iteração. Essa característica é interessante em aplicações como o estudo do fluxo de potência continuado, no qual é necessário realizar sucessivas soluções do problema de fluxo de potência convencional de um sistema com variações na demanda. Devido principalmente ao cálculo da matriz Jacobiana a cada passo do processo iterativo, o método por injeção de correntes permite reduzir os esforço computacional nesses estudos [19].

Para resolver o problema do fluxo de carga pelo método de Injeção de Correntes, deve-se seguir os seguintes passos:

1. Fazer $\nu = 0$ e definir valores iniciais de V_{r_k} e V_{imag_k} para barras PQ e PV, sendo que para as barras PV os valores devem respeitar a equação de restrição de tensão (3.77).
2. Calcular $\Delta I'^{(\nu)}_{imag_k}$ e $\Delta I'^{(\nu)}_{r_k}$ para as barras PQ pelas equações (3.88) e (3.89) e calcular ΔV_k para as barras PV utilizando a equação (3.98).
3. Testar a convergência verificando se $max|\Delta I'^{(\nu)}_{imag_k}| \leq \varepsilon$, $max|\Delta I'^{(\nu)}_{r_k}| \leq \varepsilon$ e, para as barras PV, $Max|\Delta V'^{(\nu)}_k| \leq \varepsilon$. Caso as três condições sejam satisfeitas, o processo iterativo convergiu para a solução $(V^{(\nu)}_{r_k}, V^{(\nu)}_{imag_k})$. Caso alguma das condições não seja satisfeita, ir para o passo seguinte.
4. Calcular a matriz jacobiana $\mathbf{J_I}$ para ν.
5. Utilizando (3.99), encontrar os incrementos para a nova solução $(V^{(\nu+1)}_{r_k}, V^{(\nu+1)}_{imag_k})$ dada por:

$$V_r^{(\upsilon+1)} = V_r^{(\upsilon)} + \Delta V_r^{(\upsilon)} \tag{3.109}$$

$$V_{imag}^{(\upsilon+1)} = V_{imag}^{(\upsilon)} + \Delta V_{imag}^{(\upsilon)} \tag{3.110}$$

6. Fazer $\nu + 1 \rightarrow \nu$ e voltar para o passo 2.

3.3.4 Expansão da Jacobiana em Série de Taylor

Nessa seção será apresentado um método direto de solucionar o subproblema linear do MNR para solução do PFC utilizando uma aproximação da matriz jacobiana.

Considere um PFC através da resolução do MNR. Na primeira iteração ($\nu = 0$) será necessário encontrar os incrementos Δx_0. Para isso, serão calculados os mismatches F_0 e a matriz jacobiana $\mathbf{J_0}$. Logo, Δx_0 pode ser calculado através do seguinte sistema:

$$-\mathbf{J_0}\Delta x_0 = F_0 \tag{3.111}$$

Em seguida, será possível encontrar $x_1 = x_0 + \Delta x_0$ e os novos valores para F_1. Caso não ocorra convergência para a tolerância desejada, será necessário o cálculo dos incrementos (Δx_1). Encontra-se a nova matriz jacobiana $\mathbf{J_1}$ e novamente deve-se resolver o sistema:

$$-\mathbf{J_1}\Delta x_1 = F_1 \tag{3.112}$$

Considere que a nova matriz jacobiana J_1 pode ser escrita através de uma perturbação na matriz $\mathbf{J_0}$. Logo:

$$\mathbf{J_1} = \mathbf{J_0} + \Delta\mathbf{J_1} \tag{3.113}$$

Dessa forma, o sistema (3.112) pode ser reescrito como:

$$-(\mathbf{J_0} + \Delta\mathbf{J_1})\Delta x_1 = F_1 \tag{3.114}$$

Multiplicando $\mathbf{J_0}^{-1}$ dos dois lados, tem-se:

$$-\mathbf{J_0}^{-1}(\mathbf{J_0} + \Delta\mathbf{J_1})\Delta x_1 = \mathbf{J_0}^{-1}F_1 \tag{3.115}$$

$$-(I + \mathbf{J_0}^{-1}\Delta\mathbf{J_1})\Delta x_1 = \mathbf{J_0}^{-1}F_1 \tag{3.116}$$

Considere $B_1 = (\mathbf{J_0}^{-1}F_1)$ e $K_1 = (\mathbf{J_0}^{-1}\Delta\mathbf{J_1})$. Logo, Δx_1 será dado por:

$$\Delta x_1 = -(I + K_1)^{-1}B_1 \tag{3.117}$$

Logo, para a equação (3.117), se $\|K_1\| \ll 1$, o termo $(I + K_1)^{-1}$ pode ser reescrito como:

$$(I + K_1)^{-1} = I - K_1 + K_1^2 - K_1^3 + \cdots \simeq I - K_1 \tag{3.118}$$

Dessa forma, o sistema (3.117) pode ser reescrito como:

$$\Delta x_1 = -(I - K_1)B_1 = -(I - \mathbf{J_0}^{-1}\Delta\mathbf{J_1})B_1 \tag{3.119}$$

Por fim, Δx_1 será dado por:

$$\Delta x_1 = -(B_1 - \mathbf{J_0}^{-1}\Delta\mathbf{J_1}B_1) \tag{3.120}$$

De maneira geral, os incrementos para iterações $\nu > 0$ podem ser escritos como:

$$\Delta x_\nu = -(B_\nu - \mathbf{J_0}^{-1}\Delta\mathbf{J}_\nu B_\nu) \tag{3.121}$$

Sendo que B_ν é dado por:

$$B_\nu = \mathbf{J_0}^{-1}F_\nu \tag{3.122}$$

Dessa forma, não é necessário encontrar o valor exato dos jacobianos para $\nu > 0$ e nem suas inversas, sendo necessário apenas encontrar os valores das variações $\Delta\mathbf{J}_\nu$, o que pode reduzir a quantidade de cálculos e diminuir o tempo computacional. Além disso, é necessário realizar uma única fatoração LU de $\mathbf{J_0}$.

Pode-se utilizar reordenamento AMD ou RCM e realizar a fatoração LU completa ou incompleta para substituir o valor de $\mathbf{J_0}$, de forma que $LU = \mathbf{J_0}$. Assim, os cálculos das inversas $\mathbf{J_0}^{-1} = (LU)^{-1}$ podem ser armazenados e utilizados nas iterações subsequentes. Esse método foi desenvolvido principalmente devido a característica do jacobiano por injeção de corrente, no qual apenas os elementos das diagonais das submatrizes e outros poucos elementos são modificados nas iterações.

Capítulo 4

Metodologia para Resolução do Fluxo de Carga

4.1 Introdução

A resolução de problemas de fluxo de carga envolvem sistemas não-lineares e processos iterativos. Dependendo do tamanho do sistema estudado, podem ser necessários diversos cálculos, exigindo computadores e softwares para serem viáveis. Nessa seção serão apresentados os softwares utilizados nas simulações, os sistemas de teste e suas principais características.

4.2 Matpower

Um dos aplicativos utilizados para resolução de PFC é o MATPOWER [8]. MATPOWER é um aplicativo escrito na linguagem do MATLAB que permite a resolução de problemas de fluxo de carga e problemas de otimização. Suas programação é otimizada para obter um bom desempenho computacional e é possível implementar modificações nos métodos de resolução. Ele foi desenvolvido devido as exigências da plataforma PoweWeb [8]. Neste trabalho, foram utilizadas as versões MATLAB R2012a e MATPOWER 4.1.

O MATPOWER permite utilizar diferentes algoritmos de fluxo de potência, entre eles: método de Newton Raphson, método Gauss-Seidel e método DC. Os dados são inseridos através de arquivos-M, com scripts em MATLAB. As barras do sistemas são numeradas sequencialmente e identificadas de acordo com seu tipo. Devem ser informadas as características das linhas de transmissão, os valores demandados e/ou gerados pelas cargas e geradores instalados, etc.

4.2.1 Dados

No MATPOWER, os dados são inseridos em arquivos-M e organizados em colunas. As características das barras devem ser preenchidas de acordo com a Tabela 4.1, incluindo informações como o tipo de barra,

estimativas dos módulos da fase e da tensão [20].

Tabela 4.1: Dados de Barras

Nome	Coluna	Descrição
BUS_I	1	Número da barra (inteiro positivo)
BUS_TYPE	2	Tipo de barra (1=PQ, 2=PV, 3=Swing, 4=isolada)
PD	3	Potência real demandada (MW)
QD	4	Potência reativa demandada (MVAr)
GS	5	Condutância shunt (MW demandado para $V = 1.0$ p.u.)
BS	6	Potência shunt gerada (MVAr demandado para $V = 1.0$ p.u.)
BUS_AREA	7	Número da área (inteiro positivo)
VM	8	Magnitude da tensão (p.u.)
VA	9	Fase da tensão (graus)
BASE_KV	10	Base da tensão (kV)
ZONE	11	Zona de perda (inteiro positivo)
VMAX	12	Magnitude máxima da tensão (inteiro positivo)
VMIN	13	Magnitude mínima da tensão (inteiro positivo)

As características dos geradores devem ser preenchidas de acordo com a Tabela 4.2, incluindo informações como a potência ativa e reativa geradas, módulo da tensão [20].

Tabela 4.2: Dados dos Geradores

Nome	Coluna	Descrição
GEN_BUS	1	Número da barra (inteiro positivo)
PG	2	Potência real de saída (MW)
QG	3	Potência reativa de saída (MVAr)
QMAX	4	Potência reativa máxima de saída (MVAr)
QMIN	5	Potência reativa mínima de saída (MVAr)
VG	6	Magnitude da tensão na barra geradora (p.u.)
MBASE	7	Potência total de base da máquina (MVA) (inteiro positivo)
GEN_STATUS	8	Status da máquina (> 0 em serviço, ≤ 0 fora de serviço)
PMAX	9	Potência real máxima de saída (MW)
PMIN	10	Potência real mínima de saída (MW)

As características das linhas devem ser preenchidas de acordo com a Tabela 4.3, incluindo informações como a impedância e a resistência séries equivalentes da linha, tap dos transformadores, susceptância shunt [20].

4.2.2 Algoritmos para resolução do Fluxo de Carga

O algoritmo mais utilizado para a resolução do PFC é o Newton Raphson, pois ele apresenta melhor desempenho computacional. Como foi apresentado na Subseção 3.3, pode-se resolver o problema utilizando

Tabela 4.3: Dados dos Ramos

Nome	Coluna	Descrição
F_BUS	1	Conexão de saída
T_BUS	2	Conexão de chegada
BR_R	3	Resistência (p.u.)
BR_X	4	Reatância (p.u.)
BR_B	5	Susceptância total da linha (p.u.)
TAP	9	Tap do transformador
SHIFT	10	Ângulo de defasagem do transformador (graus)
BR_STATUS	11	Status da linha (1 = em serviço, 0 = fora de serviço)

coordenadas polares, retangulares ou por injeção de corrente. O MATPOWER utiliza coordenadas polares e encontra os valores dos incrementos através do cálculo direto da inversa do jacobiano. No entanto, é possível implementar scripts que utilizam diferentes métodos diretos, método iterativos, coordenadas retangulares, injeção de corrente.

O script principal utilizado no MATPOWER para rodar o problema de fluxo de carga é denominado "runpf". Esse script lê os dados principais das barras e encontra os valores da matriz admitância Y_{bus}, os valores de potência injetados, entre outros. Em seguida, chama-se o script "newtonpf", que resolve o problema do fluxo de carga utilizando o método de Newton Raphson por coordenadas polares para a tolerância especificada, através do cálculo direto da inversa do jacobiano.

Foram desenvolvidos diferentes scripts que substituem o "newtonpf", de modo a resolver o PFC utilizando coordenadas polares, coordenadas retangulares e injeção de corrente. Os incrementos são calculados através do jacobiano para a resolução do método de Newton Raphson utilizando os métodos apresentados a seguir:

I- Métodos Diretos

1- Direto

2- LU Direto

3- ILU Direto

4- Expansão Linear

II- Métodos Iterativos com reordenamento AMD ou RCM

1- BiCGSTAB

2- GMRES

4.3 Sistemas Teste

Foram utilizados diferentes sistemas teste disponíveis no MATPOWER para simular a resolução do problema de fluxo de carga. Os sistemas bases utilizados foram o IEEE 30 barras e o IEEE 3375 barras,

que serão apresentados a seguir.

4.3.1 Sistema de 30 barras

O sistema com 30 barras é composto por 5 barras PV e 24 barras PQ. Ele possui carga total de 283,40 MW e 126,20 MVAr. Os dados foram convertidos do arquivo "ieee30cdf.txt"do IEEE em 20 de Setembro de 2004.

4.3.2 Sistema de 3375 barras

O sistema IEEE 3375 barras representa o sistema elétrico da Polônia com carregamento base relativo ao pico de carga do inverno de 2007-2008. Também são incluidos equivalentes de interconexão com Alemanha, República Checa e Eslováquia [3].

O sistema é composto por 3375 barras, sendo 293 barras PV e 2982 barras PQ. A barra #10287 é isolada do sistema. De forma a evitar interferências na convergência dos métodos iterativos, essa barra foi removida no arquivo de dados. Portanto, na realidade, trabalha-se com 3374 barras.

4.3.3 Sistemas com número de barras múltiplos de 3375

Foram utilizados sistemas fictícios de ordem superior ao de 3375 para realizar simulações e verificar o desempenho dos melhores métodos em sistemas com maiores dimensões. Esses sistemas foram gerados em [3] através de expansões e interconexões do sistema de 3375 barras, pois não há dados de sistemas maiores do que o 3775 barras para o MATPOWER. Como mencionado antes, umas das barras do sistema 3375 é isolada do sistema, logo ela foi removida. Os dados dos sistemas múltiplos são apresentados na Tabela 4.4.

Tabela 4.4: Dados dos sistemas multiplos de 3375

Sistema	barras	n_{PQ}	n_{PV}	n
Original	3374	2982	391	6355
Dobro	6747	5964	782	12710
Triplo	10120	8946	1173	19065
Quadruplo	13493	11928	1564	25420

4.4 Metodologia dos testes

Os testes foram realizados em um computador Mac Pro da Apple com processador Intel(R) Core(TM) i7-3520M CPU, frequência 2.90GHz e 8 GB de memória RAM com sistema operacional Windows 7 Professional. Para efetuar medidas de desempenho em termos de tempo de CPU de cada método, foram realizados 100 testes, sendo que os primeiros cinco resultados foram ignorados por apresentarem tempos

consideravelmente mais elevados do que a média. Atribuiu-se um mnemônico, conforme apresentado nas Tabelas 4.5 e 4.6.

Tabela 4.5: Variáveis Scripts MATLAB

Variável	Descrição
alg	Seleciona o Algoritmo utilizado na resolução do problema de Fluxo de Carga
md	Seleciona o Método Direto
me	Seleciona o Método Iterativo
re	Seleciona o Reordenamento
pre	Seleciona o tipo de Pré-Condicionador
droptol	Seleciona o DropTolerance

O método GMRES foi utilizado com restart definido em 10 iterações para todos os casos estudados. A tolerância de convergência do método de Newton-Raphson foi definida como tol=1e-4 para todos os casos.

Tabela 4.6: Valores das Variáveis Scripts MATLAB

Variável	Valor	Descrição
alg	1	coordernadas polares direto
	2	coordenadas retangulares direto
	3	injeção de corrente direto
	4	coordenadas polares iterativo
	5	coordenadas retangulares iterativo
	6	injeção de corrente reduzida iterativo
	7	injeção de corrente reordenado sem redução
md	1	Método Direto pela Inversa
	2	Método LU
	3	Método ILU
	4	Método Expansão Linear
me	1	Método Iterativo BiCGStab
	2	Método Iterativo GMRES
re	1	Reordenamento AMD
	2	Reordenamento RCM
pre	1	Pré-Condicionador T_1
	2	Pré-Condicionador T_2
	3	Pré-Condicionador T_3
	4	Pré-Condicionador Completo

Capítulo 5

Testes e Resultados Experimentais

5.1 Introdução

Nesse capítulo, descreve-se as simulações realizadas e os resultados são apresentados e analisados. Incialmente é realizado um estudo para avaliar o desempenho das funções *built-in* do MATLAB para métodos iterativos em relação à funções desenvolvidas externamente. Em seguida, são realizadas simulações para avaliação do desempenho computacional dos métodos empregados na resolução do problema de fluxo de carga (PFC). Analisa-se os resultados obtidos por diferentes formulações matemáticas do PFC e compara-se os tempos médios de CPU dos métodos estudados com o método padrão utilizado no MATPOWER. Análises de sensibilidade à parâmetros das funções utilizadas nas simulações são realizadas para verificar variações significativas. Os melhores desempenhos são testados para diferentes sistemas a fim de se obter o método com o melhor desempenho geral.

5.2 Teste GMRES

Inicialmente, foi feito um estudo dos métodos iterativos mais utilizados na engenharia, que são o GMRES e o BiCGSTAB [15]. Através dos seus algoritmos, foram desenvolvidos programas computacionais para verificar a funcionalidade e o tempo computacional demandado.

Em uma etapa inicial, com base no algoritmo proposto em [12], foi desenvolvido um script computacional no MATLAB para resolução de sistemas lineares pelo método iterativo GMRES. Como exemplo, foi utilizado a primeira iteração de um PFC com três barras resolvido pelo Método de Newton-Raphson por coordenadas polares. O objetivo do teste é comparar o tempo de CPU do script desenvolvido com o script do próprio MATLAB para a resolução de sistemas lineares pelo método iterativo GMRES.

5.2.1 Problema com três barras

Considere o seguinte problema de três barras:

A base do sistema é igual a 100 MVA. Logo, as potências injetadas nas barras serão: $P_2 = -0,5\,pu$,

Tabela 5.1: Dados para o PFC com 3 barras

Tipo de barra	**V** [pu]	θ [rad]	$\mathbf{P_G}$ [MW]	$\mathbf{P_D}$ [MW]	$\mathbf{Q_G}$ [MVar]	$\mathbf{Q_D}$ [MVar]
Swing (1)	1	0	-	-	-	-
PQ (2)	-	-	-	50	-	30
PV (3)	1,05		80	60	-	30

Tabela 5.2: Impedâncias das linhas

Linha	$\mathbf{z_{linha}}$ [pu]
1-2	$j0,10$
2-3	$0,05 + j0,10$

$Q_2 = -0,3\,pu$ e $P_3 = 0,8 - 0,6 = 0,2\,pu$. Considere como valor iniciais para as variáveis desconhecidas: $V_2^{(0)} = 1\,pu$ e $\theta_2^{(0)} = \theta_3^{(0)} = 0\,rad$.

Será considerada apenas a primeira iteração ($\nu = 0$). Utilizando o método de Newton-Raphson por coordenadas polares é necessário resolver o sistema linear dado em (3.43). Dessa forma, para um sistema $Ax = b$, tem-se:

$$A = -J^{(0)} \tag{5.1}$$

$$x = \begin{bmatrix} \Delta\theta^{(0)} \\ \Delta V^{(0)} \end{bmatrix} = \begin{bmatrix} \Delta\theta_2^{(0)} \\ \Delta\theta_3^{(0)} \\ \Delta V_2^{(0)} \end{bmatrix} \tag{5.2}$$

$$b = \begin{bmatrix} \Delta P^{(0)} \\ \Delta Q^{(0)} \end{bmatrix} = \begin{bmatrix} \Delta P_2^{(0)} \\ \Delta P_3^{(0)} \\ \Delta Q_2^{(0)} \end{bmatrix} \tag{5.3}$$

Dada pela equação (3.44), a primeira iteração fornece um jacobiano igual a:

$$J^{(0)} = \begin{bmatrix} 18,4 & -8,4 & 3,8 \\ -8,4 & 8,4 & -4,2 \\ -4,2 & 4,2 & 17,6 \end{bmatrix} \tag{5.4}$$

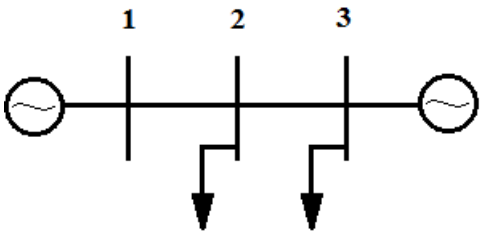


Figura 5.1: Problema de 3 barras

Os mismatches da primeira iteração, calculados por (3.40) e (3.41), resultam em:

$$\begin{bmatrix} \Delta P_2^{(0)} \\ \Delta P_3^{(0)} \\ \Delta Q_2^{(0)} \end{bmatrix} = \begin{bmatrix} 0,3 \\ 0,01 \\ -0,1 \end{bmatrix} \tag{5.5}$$

Logo, a resposta esperada para os incrementos da primeira iteração pode ser definida como:

$$\begin{bmatrix} \Delta\theta_2^{(0)} \\ \Delta\theta_3^{(0)} \\ \Delta V_2^{(0)} \end{bmatrix} = - \begin{bmatrix} 18,4 & -8,4 & 3,8 \\ -8,4 & 8,4 & -4,2 \\ -4,2 & 4,2 & 17,6 \end{bmatrix}^{-1} \begin{bmatrix} 0,3 \\ 0,01 \\ -0,1 \end{bmatrix} = \begin{bmatrix} -0,0308 \\ -0,0293 \\ 0,0053 \end{bmatrix} \tag{5.6}$$

Dessa forma, somando os incrementos obtidos aos valores inicias, os novos valores das variáveis desconhecidas passam a ser: $V_2^{(1)} = 1,0053\,pu$, $\theta_2^{(1)} = -0,0308\,rad$ e $\theta_3^{(1)} = -0,0293\,rad$.

5.2.2 Código GMRES

Com base no Algortimo 2.1 apresentado na Seção 2.4.2.1 para o método iterativo GMRES, foi desenvolvido uma função em MATLAB para resolução de um sistema linear pelo método GMRES, cujo código será apresentado em anexo. Os resultados obtidos foram:

Tabela 5.3: Resultados script GMRES

Variável	**Valor obtido**
$\Delta\theta_2^{(0)}$	-0,0308
$\Delta\theta_3^{(0)}$	-0,0293
$\Delta V_2^{(0)}$	1,0053
Resíduo	6.65E-11

Foram necessárias 3 iterações para obter a convergência. Vê-se que esses resultados estão de acordo com o esperado, apresentados em (5.6). O resíduo obtido é muito baixo e indica que houve convergência. Em seguida, foi testado o tempo computacional demandado pelo script para a resolução de um sistema linear maior com 9 barras. Os tempos computacionais médios demandados após 100 testes do script GMRES *built-in* do MATLAB e do script desenvolvido são comparandos na Tabela 5.4.

Tabela 5.4: Tempos testes GMRES para o problema com 9 barras

Método	**Tempo Médio [s]**
GMRES MATLAB	0,0497
GMRES Desenvolvido	0,1058

Um gráfico dos tempos computacionais em função dos testes é mostrado na Figura 5.2. Vê-se que o tempo computacional para o caso de 9 barras é menor para o script *built-in* do MATLAB. Essa diferença é ainda maior para sistemas com grandes quantidades de barras. Além disso, percebe-se pelo gráfico que

o script *built-in* apresenta um tempo computacional mais estável. Isso também se aplica para o método BiCGStab. Dessa forma, nota-se que a melhor opção é utilizar os scripts *built-in* do MATLAB para os métodos iterativos, pois estes já estão otimizados. Este foi o procedimento adotado nas demais simulações.

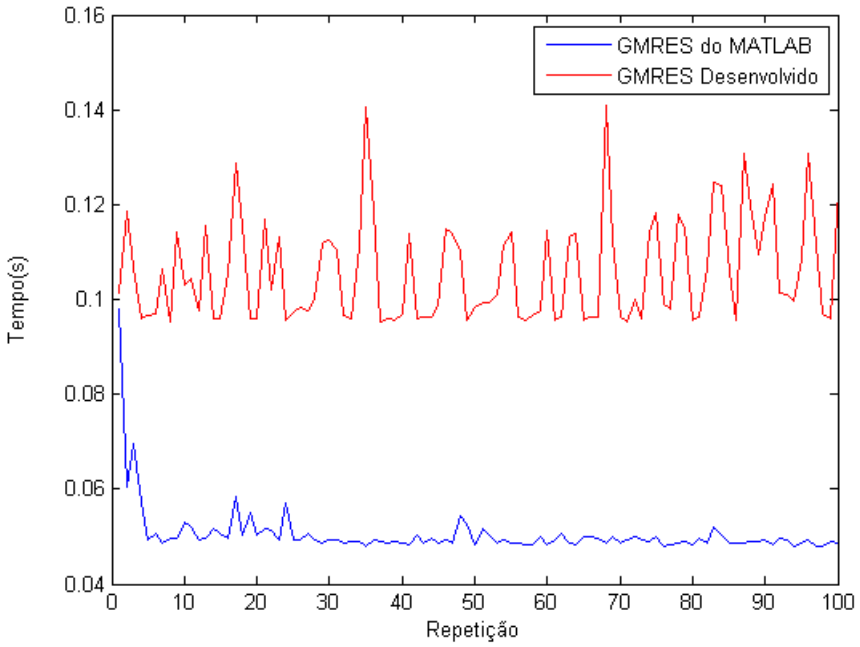


Figura 5.2: Comparação dos tempos computacionais para diferentes scripts com o método GMRES

5.3 Teste LU e ILU

Foram realizados testes para comparar os tempos computacionais demandados para a construção das matrizes triangulares L e U pelos método LU e ILUTP no MATLAB para a matriz jacobiana do caso das 3375 barras utilizando coordenadas polares. O método ILU foi utilizado com os parâmetros milu='off' e type='ilutp'. Os parâmetros *droptol* e thresh foram alterados de acordo com a Tabela 5.5.

Tabela 5.5: Tempos computacionais para construção das matrizes L e U, conforme o tipo de fatoração

Método	Tempo[s]
LU	0,0052
ILUTP thresh=1 droptol=0	0,0053
ILUTP thresh=0 droptol=0	**0,0041**
ILUTP thresh=0 droptol=6.5e-5	0,0047

Observa-se pela Tabela 5.5 e pela Figura 5.3 que o melhor tempo foi obtido utilizando o método ILU com os parâmetros thresh=0 e droptol=0. No entanto, deve-se verificar se os valores de L e U obtidos por esse método propiciam resultados satisfatórios como pré-condicionadores nos métodos iterativos. Esses testes serão realizados nas seções seguintes.

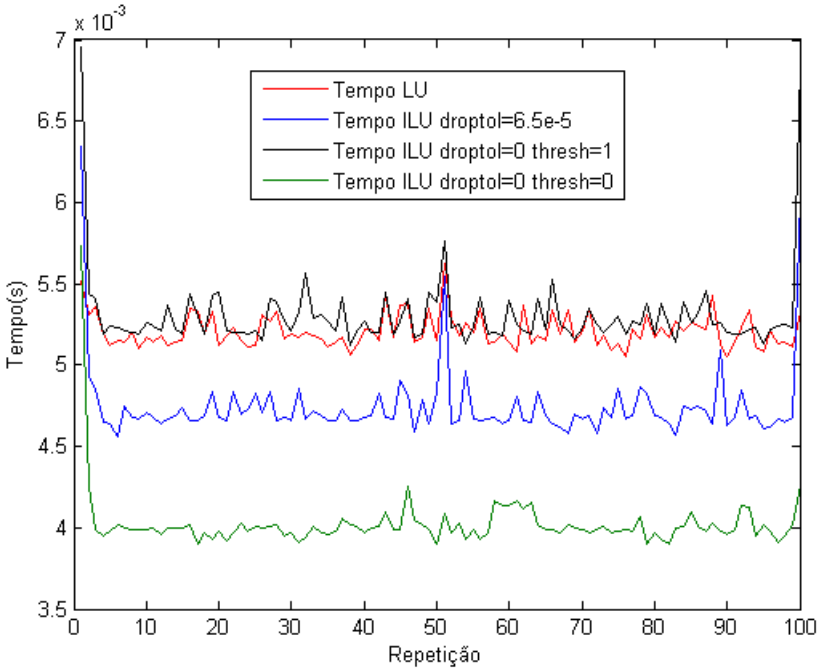


Figura 5.3: Comparação dos tempos computacionais para construção das matrizes L e U

5.4 Teste Preenchimento

Considere uma matriz A esparsa com ordem 53x53 de um sistema $Ax = b$ formada pelo jacobiano do método polar para o caso de 30 barras, apresentado no Capítulo 4. Observa-se na Figura 5.4 a estrutura das matrizes A, L, U e LU. A matriz A possui quantidade de elementos não nulos nz=379. Já a matriz LU gerada pela função 'lu' do MATLAB possui nz=527, consideravelmente superior. Esses valores, anteriormente nulos na matriz A, são pequenos, de forma que a matriz LU converge diretamente ao resultado, ou seja, $x = U^{-1}L^{-1}b = A^{-1}b$.

Para sistemas maiores, o preenchimento aumenta significativamente. Considere uma matriz A esparsa com ordem 6355x6355 de um sistema $Ax = b$ formada pelo jacobiano do método polar para o caso de 3375 barras, apresentado no Capítulo 4. Observa-se na Figura 5.5 a estrutura das matrizes A, L, U e LU. A matriz A possui quantidade elementos não nulos nz=40703. Já a matriz LU gerada pela função 'lu' do MATLAB possui nz=2057088, ou seja, 50 vezes mais elementos. Isso torna o processo para encontrar as matrizes L e U muito lento. Dessa forma, a fatoração ILU busca evitar o preenchimento utilizando $M = \widetilde{L}\widetilde{U} \simeq A$.

Considerando a matriz esparsa A de ordem 53x53 apresentada na Figura 5.4, utilizando a fatoração ILUTP com os parâmetros milu='off', droptol=0 e thresh=0 no MATLAB, pode-se observar que a diferença de elementos não nulos passou de nz=527 para nz=443. O preenchimento pode ser reduzido ainda mais utilizado reordenamento.

O reordenamento contribui para redução considerável do preenchimento que ocorre na fatoração LU completa em matrizes esparsas. Considere a matriz A de ordem 53x53 apresentada na Figura 5.4. Observa-se a estrutura das matrizes nas Figuras 5.7 e 5.8 utilizando, respectivamente, reordenamento RCM e AMD na matriz A e aplicando a fatoração ILUTP com os parâmetros milu='off', droptol=0 e thresh=0 no MA-

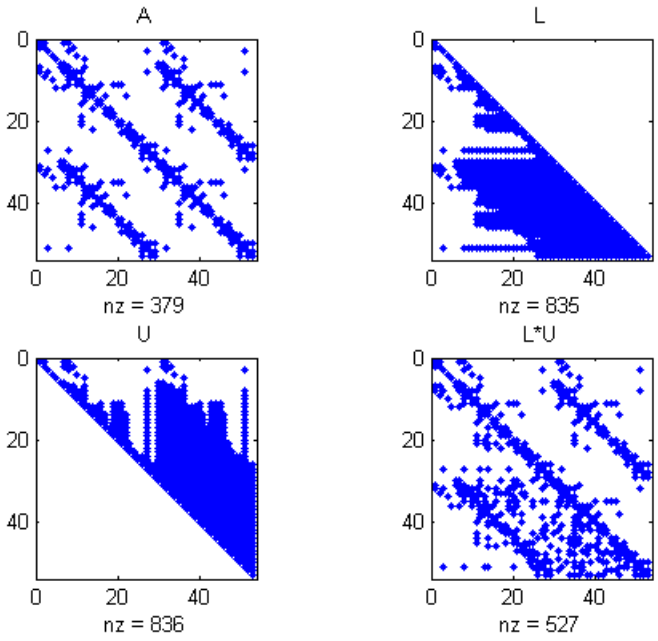


Figura 5.4: Estruturas das matrizes *A*, *L*, *U* e *LU* para A 53x53

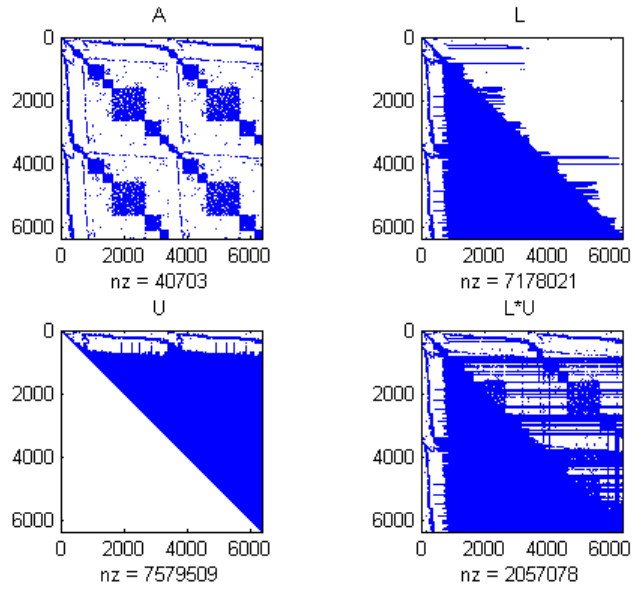


Figura 5.5: Estruturas das matrizes *A*, *L*, *U* e *LU* para A 6355x6355

TLAB.

Percebe-se que a quantidade de elementos não-nulos passou de nz=379 para nz=394 com o reordenamento RCM e para nz=382 com o reordenamento AMD, consideravelmente inferior ao nz=443 obtido utilizando ILUTP sem reordenamento na Figura 5.6. Para sistemas de grande porte com matrizes esparsas, o reordenamento permite diminuir consideravelmente o preenchimento. Considere a matriz A esparsa com ordem 6355x6355 apresentada na Figura 5.5. Observa-se na Figura 5.9 a estrutura das matrizes apli-

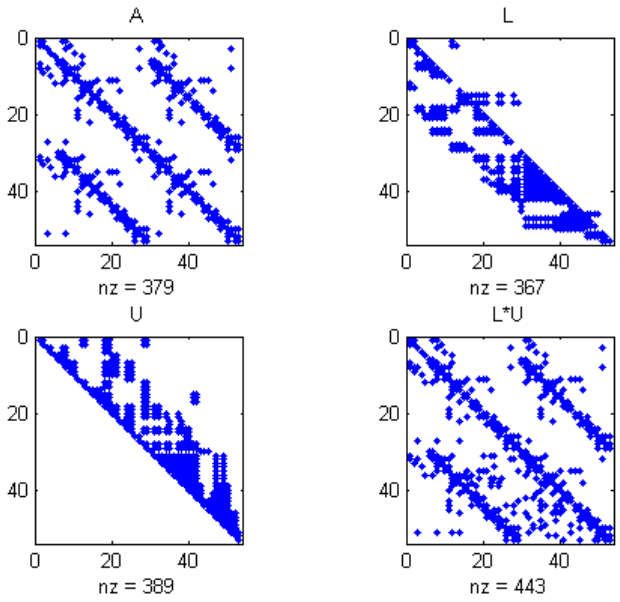


Figura 5.6: Estruturas das matrizes *A*, *L*, *U* e *LU* utilizando ILUTP para A 53x53

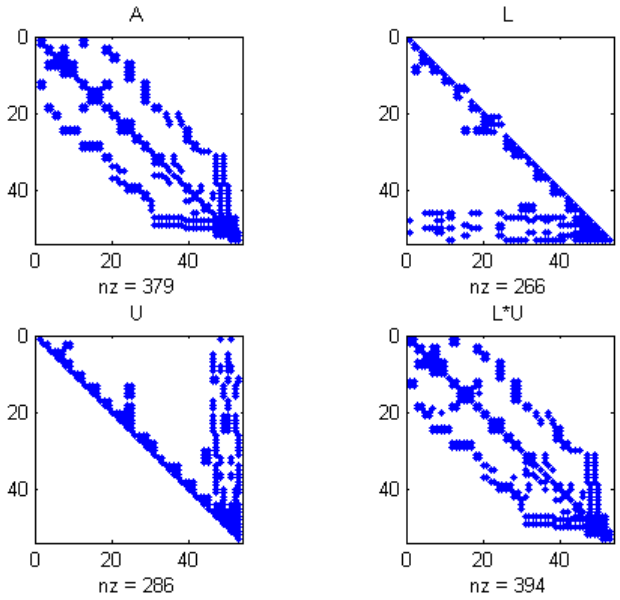


Figura 5.7: Estruturas das matrizes *A*, *L*, *U* e *LU* utilizando reordenamento RCM e ILUTP em A 53x53

cando o reordenamento AMD na matriz *A* e fatoração ILUTP com os parâmetros milu='off', droptol=0 e thresh=0 no MATLAB.A quantidade de elementos não nulos passou de nz=40703 para nz=43629, consideravelmente melhor do que nz=2057088 obtido pela fatoração LU completa sem reordenamento na na Figura 5.5. Os resultados obtidos estão resumidos nas Tabelas 5.6 e 5.7. Percebe-se que o preenchimento é consideravelmente reduzido com a aplicação do reordenamento AMD e a fatoração ILUTP.

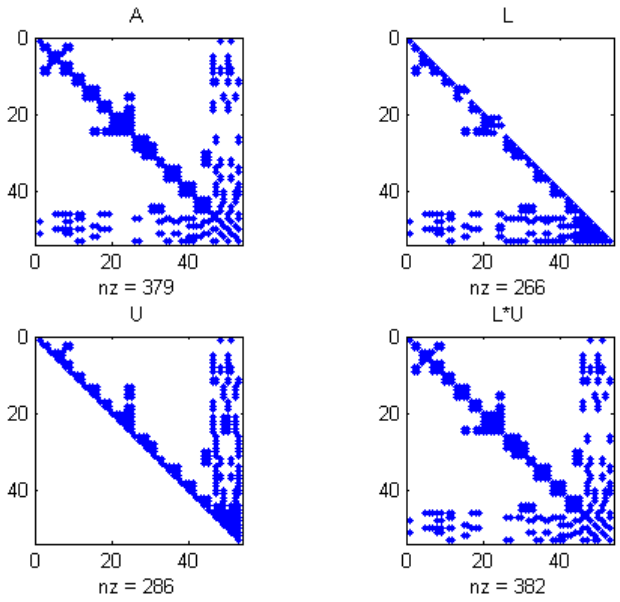


Figura 5.8: Estruturas das matrizes *A*, *L*, *U* e *LU* utilizando reordenamento AMD e ILUTP em A 53x53

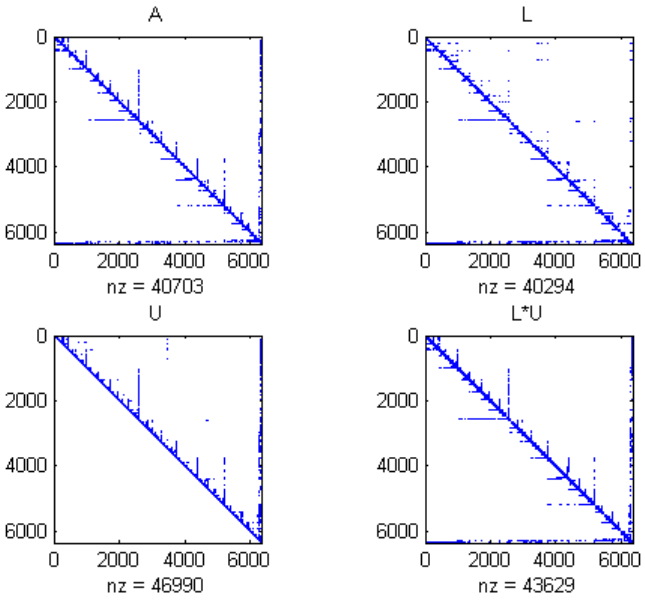


Figura 5.9: Estruturas das matrizes *A*, *L*, *U* e *LU* utilizando reordenamento AMD e ILUTP em A 6355x6355

5.5 Resultados com Coordenadas Polares

Nessa seção serão apresentados os testes feitos e os resultados obtidos para a resolução do PFC utilizando o Newton-Raphson com coordenadas polares por métodos diretos e por métodos iterativos para encontrar os incrementos. Foi utilizada uma tolerância tol=1e-4 para a convergência do método de Newton-Raphson em todas as simulações realizadas.

Tabela 5.6: Preenchimento das fatorações LU e ILU para matriz A 53x53

Fatoração	Reordenamento	Elementos não nulos de *LU*
LU Completa	-	527
ILUTP	-	443
ILUTP	RCM	394
ILUTP	AMD	382
Matriz Original A 53x53		379

Tabela 5.7: Preenchimento das fatorações LU e ILU para matriz A 6355x6355

Fatoração	Reordenamento	Elementos não nulos de *LU*
LU Completa	-	2057078
ILUTP	-	728735
ILUTP	RCM	53809
ILUTP	AMD	43629
Matriz Original A 6355x6355		40703

5.5.1 Métodos Diretos

Foram analisados quatro diferentes formas de método direto, sendo o primeiro utilizando o cálculo pela inversa do jacobiano, que é o método aplicado originalmente no MATPOWER. O segundo, utilizando o método com decomposição LU e ILU pela equação (5.7) utilizando fatores ILU apresentado na Subseção 2.4.3.1 e na Seção 5.3 com o parâmetros $LU = \mathbf{J_P}$, droptol=0, thresh=0, milu='off' e type='ilutp'.

$$x = U^{-1}L^{-1}b \tag{5.7}$$

Por fim, foi analisado o método direto utilizando a Expansão Linear apresentada na Subseção 3.3.4. O reordenamento não foi aplicado no método direto pela inversa do jacobiano, pois este não apresentou mudanças significativas, independente do tipo de reordenamento utilizado. Para a aplicação dos métodos LU e ILU, foram utilizados os reordenamentos RCM e AMD. Para a Expansão Linear, foi utilizado apenas o reordenamento AMD, pois com o reordenamento RCM o método não convergiu. Os resultados em termos de tempo médio de CPU obtidos podem ser observados na Tabela 5.8.

Tabela 5.8: Tempos médios para métodos diretos da solução do PFC por coordenadas polares

Método	Reordenamento	Tempo Solução PFC[s]
Direto	-	0,0935
LU	RCM	0,1702
	AMD	0,0519
ILU	RCM	0,1363
	AMD	**0,0414**
Expansão Linear	AMD	0,0537

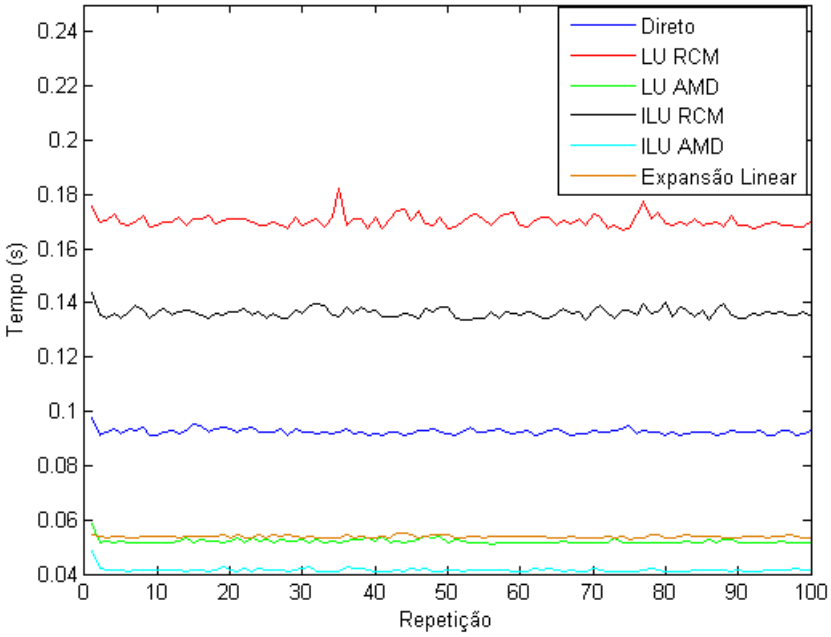


Figura 5.10: Tempos computacionais para métodos diretos com coordenadas polares

Pode-se observar que os melhores resultados foram obtidos utilizando o reordenamento AMD. Para o reordenamento RCM, os resultados foram piores do que o método direto pela inversa, enquanto que os métodos LU, ILU e Expansão Linear por AMD tiveram tempos melhores. O melhor tempo foi obtido utilizando ILU e reordenamento AMD.

5.5.2 Métodos Iterativos

Foi analisado o PFC com 3375 barras utilizando diferentes métodos iterativos para coordenadas polares, apresentados na Tabela 5.9.

Tabela 5.9: Métodos Iterativos

Nº	Reordenamento	Método
1	RCM	GMRES
2	RCM	BiCGSTAB
3	AMD	GMRES
4	AMD	BiCGSTAB

Foram testados diferentes tipos de pré-condicionadores e parâmetros da função ILU presente no MATLAB. Os resultados obtidos são apresentados e discutidos a seguir.

5.5.2.1 Pré-Condicionador Variável

Nessa seção serão apresentados os resultados obtidos por coordenadas polares utilizando o pré-condicionador variável baseado na matriz jacobiana $\mathbf{J_P}$ completa pelo método ILUTP modificado apresentado na Subseção 2.4.3.1. O reordenamento e o pré-condicionador foram calculados a cada nova iteração após a obtenção da nova matriz jacobiana. Foram analisados os dois métodos iterativos apresentados na Tabela 5.9.

Os parâmetros utilizados para a função ILU do MATLAB foram droptol=0, thresh=0, milu='off' e type='ilutp'.

Tabela 5.10: Tempos métodos médios de solução do PFC por MNR com coordenadas polares com pré-condicionador variável

Reordenamento	Método	Tempo Solução PFC[s]
-	Direto	0,0935
RCM	GMRES	0,1320
	BiCGSTAB	0,1183
AMD	GMRES	0,0672
	BiCGSTAB	**0,0577**

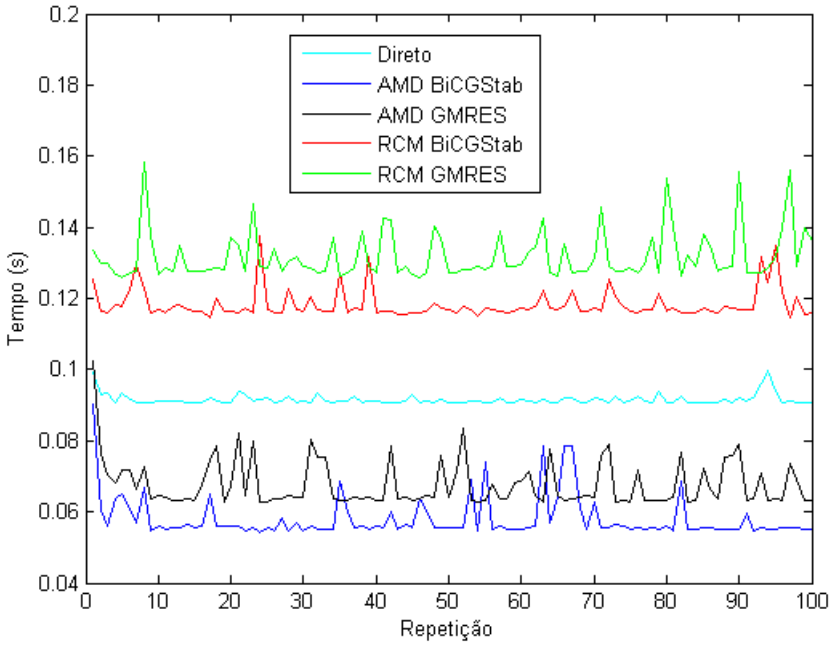


Figura 5.11: Tempos computacionais para métodos com coordenadas polares com pré-condicionador variável

Todos os métodos convergiram corretamente ao resultado esperado em 4 iterações do MNR. Observa-se que os métodos utilizando reordenamento AMD apresentaram resultados melhores do que o reordenamento RCM. O melhor resultado foi obtido utilizando reordenamento AMD e o método iterativo BiCGStab.

5.5.2.2 Pré-Condicionador Fixo

Nessa seção serão apresentados os resultados obtidos por coordenadas polares utilizando o pré-condicionador fixo baseado na matriz jacobiana $\mathbf{J_P}$ completa pelo método ILUTP modificado apresentado na Subseção 2.4.3.1. O reordenamento e o pré-condicionador foram calculados apenas na primeira iteração e utilizados nas iterações subsequentes. Foram analisados os dois métodos iterativos apresentados na Tabela 5.9.

Os parâmetros utilizados para a função ILU do MATLAB foram droptol=0, thresh=0, milu='off' e type='ilutp'.

Tabela 5.11: Tempos métodos médios de solução do PFC por MNR com coordenadas polares com pré-condicionador fixo

Reordenamento	Método	Tempo Solução PFC[s]
-	Direto	0,0935
RCM	GMRES	0,0722
	BiCGSTAB	0,0631
AMD	GMRES	0,0402
	BiCGSTAB	**0,0330**

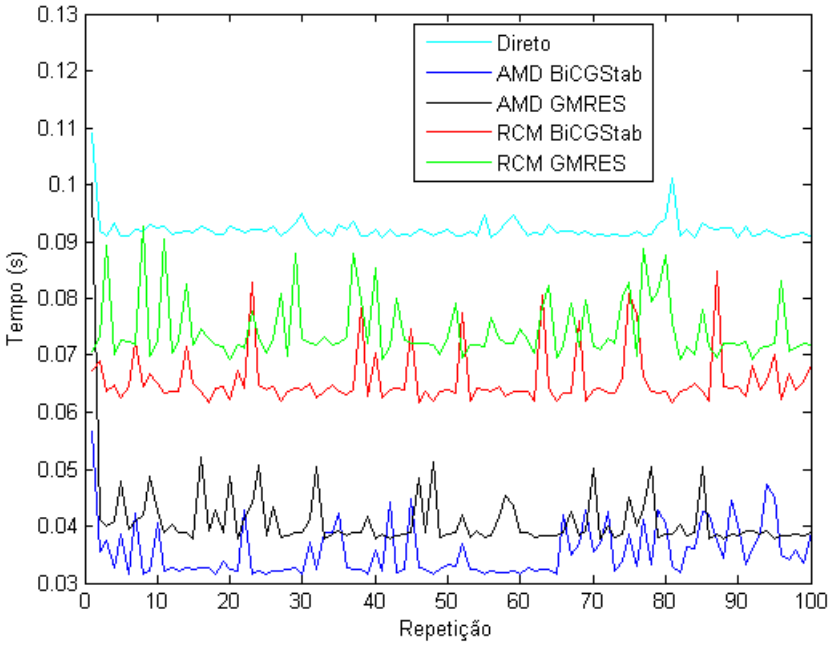


Figura 5.12: Tempos computacionais para métodos com coordenadas polares com pré-condicionador fixo

Observa-se redução significativa nos tempos médios de CPU em relação aos resultados utilizando pré-condicionadores variáveis. O melhor tempo, utilizando reordenamento AMD e método iterativo BiCGStab, teve uma redução de aproximadamente 43%. O melhor tempo é 2,8 vezes menor do que o tempo obtido pelo método direto utilizado no MATPOWER.

5.5.2.3 Outros Tipos de Pré-Condicionadores

Nos métodos utilizando coordenadas polares apresentados anteriormente, foram utilizados pré-condicionadores gerados pelo método ILU baseados na matriz jacobiana $\mathbf{J_P}$ completa. Foram analisadas outras três opções de pré-condicionadores, realizando o método ILU na matriz $\mathbf{J_P}$, apresentada na equação (3.45), modificada como:

1. T_1 zerando as duas submatrizes N e M, logo:

$$T_1 = \begin{bmatrix} H & 0 \\ 0 & L \end{bmatrix} \tag{5.8}$$

2. T_2 zerando a submatriz N, logo:

$$T_2 = \begin{bmatrix} H & 0 \\ M & L \end{bmatrix} \tag{5.9}$$

3. T_3 zerando a submatriz M, logo:

$$T_3 = \begin{bmatrix} H & N \\ 0 & L \end{bmatrix} \tag{5.10}$$

Os pré-condicionadores e os parâmetros do reordenamento foram calculados apenas na primeira iteração e usados nas iterações subsequentes. Os parâmetros utilizados para a função ILU do MATLAB foram droptol=0, thresh=0, milu='off' e type='ilutp'.

Tabela 5.12: Tempos médios para diferentes pré-condicionadores utilizados na solução do PFC por MNR com coordenadas polares

Reordenamento	Método	Tempo Solução PFC[s]			
		$\mathbf{J_P}$	T_1	T_2	T_3
RCM	GMRES	0,0722	0,0709	0,0679	0,0658
	BiCGSTAB	0,0631	0,0553	0,0566	0,0535
AMD	GMRES	0,0402	0,0607	0,0474	0,0471
	BiCGSTAB	**0,0330**	0,0440	0,0390	0,0362

Observa-se que os tempos obtidos utilizando os pré-condicionadores T_1, T_2 e T_3 foram superiores em relação ao tempo obtido utilizando o pré-condicionador baseado na matriz J_P completa para o reordenamento AMD. Já para o reordenamento RCM, os resultados foram melhores do que utilizando J_P completa. O pré-condicionador T_3 apresentou os melhores resultados dentre os diferentes pré-condicionadores estudados.

Verificou-se que o *droptol* utilizado no método ILUTP influencia diretamente nos tempos dos métodos utilizando pré-condicionadores. Foram analisados os diferentes tipos de pré-condicionadores para o método com os melhores resultados, BiCGStab com reordenamento AMD, utilizando droptol=0 e droptol=$6,510^{-5}$.

Para *droptol* igual a $6,5 \times 10^{-5}$ o melhor desempenho foi obtido utilizando o pré-condicionador T_3, o que confirma os resultados obtidos em [3] e [4]. Porém, o melhor desempenho geral foi obtido utilizando *droptol* igual a 0, no qual o pré-condicionador completo apresentou o melhor resultado.

Tabela 5.13: Tempos médios para diferentes pré-condicionadores e *droptol* utilizados na solução do PFC por MNR com coordenadas polares

Droptol	Reordenamento	Método	Tempo Solução PFC[s]			
			$\mathbf{J_P}$	T_1	T_2	T_3
0	AMD	BiCGSTAB	**0,0330**	0,0440	0,0390	0,0362
$6,5 \times 10^{-5}$	AMD	BiCGSTAB	0,0393	0,0460	0,0427	**0,0374**

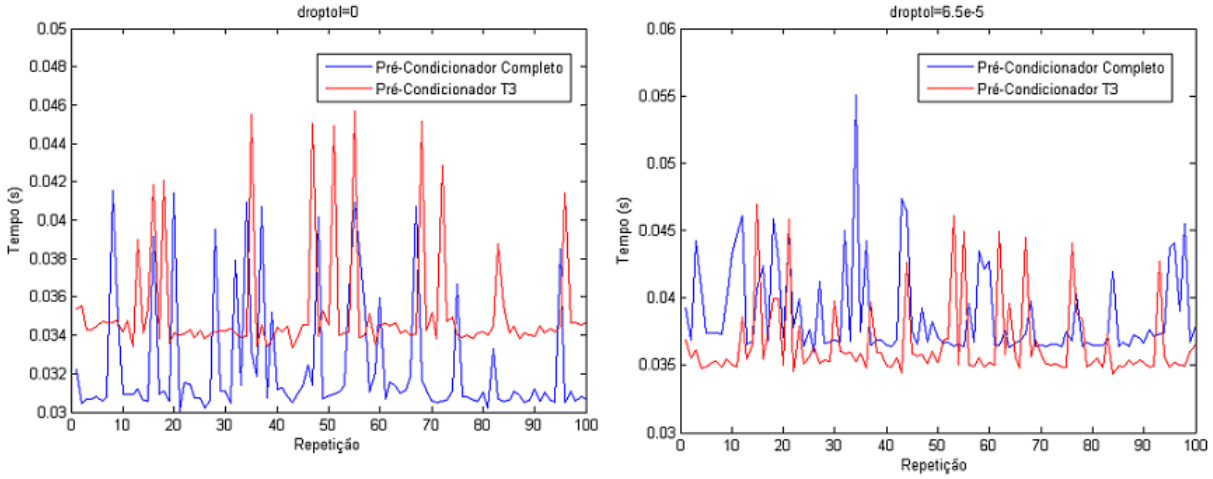


Figura 5.13: Tempos computacionais para Pré-Condicionador completo e T3 utilizando droptol=0 e droptol=6.5e-5

Foram analisados os tempos demandados e quantidade de iterações internas para cada iteração do método de Newton Raphson. Limitou-se a análise ao método iterativo BiCGStab com reordenamento AMD utilizando pré-condicionadores pela matriz completa $\mathbf{J_P}$ e a matriz T_3, que foram os métodos com os melhores resultados para coordenadas polares. Os testes foram realizados para droptol=0 e para droptol=$6,5 \times 10^{-5}$. Os dados estão apresentados na Tabela 5.14.

Tabela 5.14: Tempos computacionais médios para método por coordenadas polares para cada iteração do MNR

Iteração MNR	droptol=0				droptol=6.5e-5			
	Iter. BiCGStab		Tempos[s]		Iter. BiCGStab		Tempos[s]	
	$\mathbf{J_P}$	T_3	$\mathbf{J_P}$	T_3	$\mathbf{J_P}$	T_3	$\mathbf{J_P}$	T_3
1	0,5	1,5	0,0127	0,0136	2,5	2,5	0,0152	0,0146
2	1,5	2,5	0,0065	0,0071	3	2,5	0,0078	0,0073
3	2	3	0,0068	0,0078	3	4	0,0078	0,0077
4	2	3	0,0070	0,0077	3,5	3	0,0085	0,0078
Total	**6**	10	**0,0330**	0,0362	12	12	0,0393	0,0374

Observa-se que para os métodos utilizando droptol=$6,5 \times 10^{-5}$, a quantidade de iterações internas totais foram iguais, porém o tempo computacional foi inferior para o método utilizando o pré-condicionador

T_3. Os métodos utilizando droptol=0 apresentam menores tempos e menores quantidades de iterações internas totais.

O método utilizando a matriz $\mathbf{J_P}$ e droptol=0 apresentou a menor quantidade de iterações internas e o menor tempo computacional. Foram necessárias 0,5 iterações para o Método BiCGStab na primeira iteração do MNR. Isso mostra que as matrizes triangulares $\tilde{L}$ e $\tilde{U}$ geradas pelo método ILUTP com droptol=0 e thresh=0 são praticamente equivalentes às matrizes L e U completas sem truncamento ($\mathbf{J_P} = LU$) e convergem diretamente ao resultado esperado, porém demandam menos tempo para serem construídas, como foi mostrado na Seção 5.3.

Nas iterações subsequentes do MNR, a quantidade de iterações internas do método BiCGStab utilizando utilizando a matriz $\mathbf{J_P}$ e droptol=0 aumentou de 0,5 para 1,5 e 2. Isso ocorre pois o pré-condicionador obtido na primeira iteração foi fixado nas iterações subsequentes, de maneira que o resultado não é mais obtido de forma direta. No entanto, ao fixar o pré-condicionador, o tempo de CPU demandando pelas maior quantidade de iterações exigidas é inferior ao tempo necessário para o cálculo de um novo pré-condicionador para obter o resultado direto.

5.6 Resultados com Coordenadas Retangulares

Nessa seção serão apresentados os testes feitos e os resultados obtidos para a resolução do PFC utilizando o Newton-Raphson com coordenadas retangulares por métodos diretos e por métodos iterativos para encontrar os incrementos.

5.6.1 Métodos Diretos

Foram analisados quatro diferentes métodos diretos, sendo o primeiro utilizando o cálculo direto pela inversa do jacobiano; o segundo utilizando o método LU, em seguida utilizando o método ILU apresentado na Subseção 2.4.3.1 com os parâmetros droptol=0, thresh=0, milu='off' e type='ilutp', e por fim utilizando a expansão linear apresentada na Subseção 3.3.4. Para a aplicação dos métodos LU e ILU, foram utilizados os reordenamentos RCM e AMD. Para a expansão linear, foi utilizado o reordenamento AMD. Os resultados obtidos podem ser observados na Tabela 5.15.

Tabela 5.15: Tempos médios para métodos diretos de solução do PFC por coordenadas retangulares

Método	Reordenamento	Tempo Solução PFC[s]
Direto	-	0,1243
LU	RCM	0,3625
	AMD	0,1685
ILU	RCM	0,1956
	AMD	**0,0796**
Expansão Linear	AMD	0,1075

Observa-se que, assim como para coordenadas polares, os piores resultados foram obtido utilizando

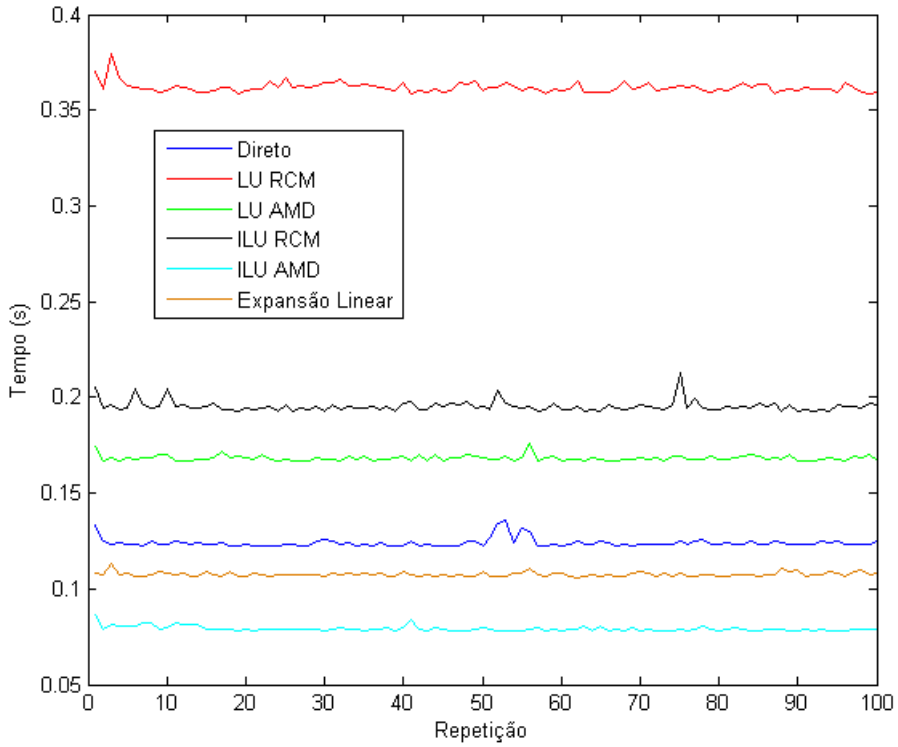


Figura 5.14: Tempos computacionais para métodos diretos com coordenadas retangulares

o reordenamento RCM. Os métodos ILU e Expansão Linear por AMD tiveram tempos melhores do que o método direto pela inversa do jacobiano. O melhor tempo foi obtido utilizando ILU e reordenamento AMD.

5.6.2 Métodos Iterativos

Foi analisado o PFC com 3375 barras utilizando diferentes métodos iterativos para coordenadas retangulares, apresentados na Tabela 5.9.

Foram testados diferentes tipos de pré-condicionadores e parâmetros da função ILU presente no MATLAB. Os resultados obtidos serão apresentados e discutidos a seguir.

5.6.2.1 Pré-Condicionador Variável

Nessa seção serão apresentados os resultados obtidos por coordenadas retangulares utilizando o pré-condicionador variável baseado na matriz jacobiana $\mathbf{J_R}$ completa pelo método ILUTP modificado apresentado na Subseção 2.4.3.1. O reordenamento e o pré-condicionador foram calculados a cada nova iteração após a obtenção da nova matriz jacobiana. Foram analisados os quatro métodos iterativos apresentados na Tabela 5.9.

Os parâmetros utilizados para a função ILU do MATLAB foram droptol=0, thresh=0, milu='off' e

type='ilutp'.

Tabela 5.16: Tempos médios dos métodos de solução do PFC por MNR com coordenadas retangulares com pré-condicionador variável

Reordenamento	Método	Tempo Solução PFC[s]
-	Direto	0,1243
RCM	GMRES	0,1971
	BiCGStab	0,2262
AMD	GMRES	**0,1129**
	BiCGStab	0,1263

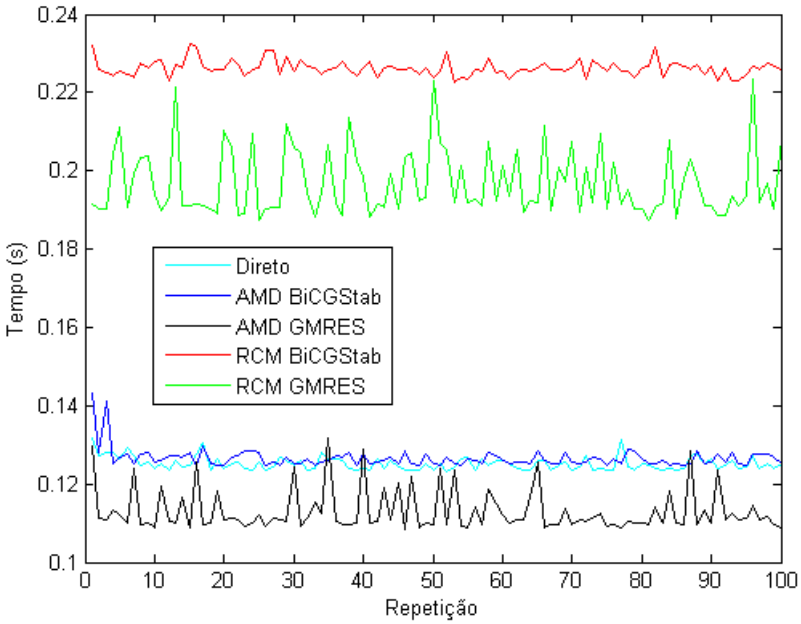


Figura 5.15: Tempos computacionais para métodos com coordenadas retangulares com pré-condicionador variável

Observa-se que, utilizando o pré-condicionador variável para coordenadas retangulares, os melhores resultados foram obtido para o reordenamento AMD. Diferentemente dos método com coordenadas polares, os tempos obtidos para o método iterativo GMRES foram melhores do que os tempos obtidos para BiCGStab para cada reordenamento. O melhor tempo foi obtido utilizando reordenamento AMD e o método iterativo GMRES.

5.6.2.2 Pré-Condicionador Fixo

Nessa seção serão apresentados os resultados obtidos por coordenadas retangulares utilizando o pré-condicionador fixo baseado na matriz jacobiana $\mathbf{J_R}$ completa pelo método ILUTP modificado apresentado na Subseção 2.4.3.1. O reordenamento e o pré-condicionador foram calculados apenas na primeira iteração

e utilizados nas iterações subsequentes. Foram analisados os quatro métodos iterativos apresentados na Tabela 5.9.

Os parâmetros utilizados para a função ILU do MATLAB foram droptol=0, thresh=0, milu='off' e type='ilutp'.

Tabela 5.17: Tempos médios dos métodos de solução do PFC por MNR com coordenadas retangulares com pré-condicionador fixo

Reordenamento	Método	Tempo Solução PFC[s]
-	Direto	0,1243
RCM	GMRES	0,1255
	BiCGStab	0,1239
AMD	GMRES	**0,0787**
	BiCGStab	0,0796

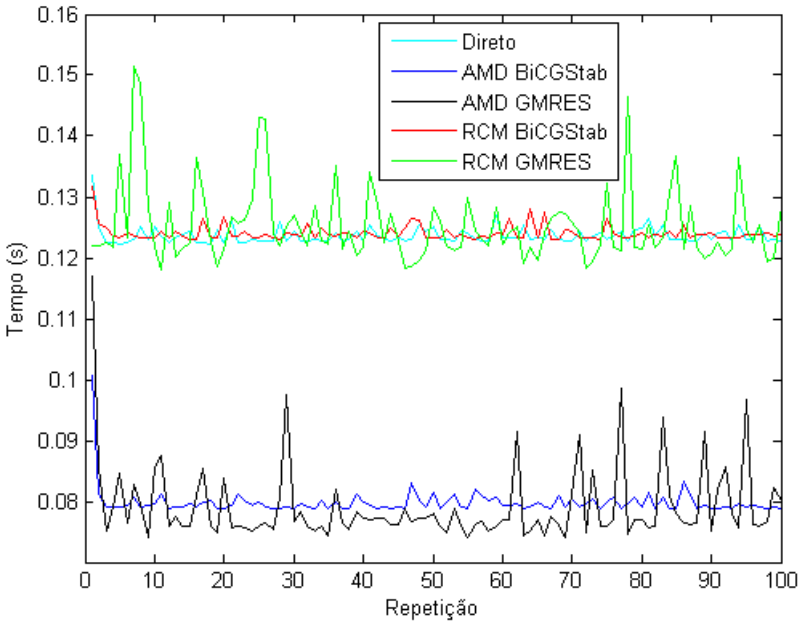


Figura 5.16: Tempos computacionais para métodos com coordenadas polares com pré-condicionador fixo

Observa-se que o padrão de resultados foi o mesmo obtido para o pré-condicionador variável, porém os tempos foram reduzidos. Isso se deve ao fato do pré-condicionador fixo ter garantido a convergência dos método iterativos, sem ser necessário utilizar tempo computacional para calcular novos valores para o pré-condicionador nas iterações subsequentes. O melhor tempo foi obtido utilizando reordenamento AMD e o método iterativo GMRES. Esse tempo foi aproximadamente 30% menor do que o tempo obtido para o mesmo método utilizado pré-condicionador variável.

5.6.2.3 Outros Tipos de Pré-Condicionadores

Nos métodos utilizando coordenadas retangulares apresentados anteriormente, foram utilizados pré-condicionadores gerados pelo método ILU baseados na matriz jacobiana $\mathbf{J_R}$ completa. Foram realizados outras opções de pré-condicionadores modificando a matriz $\mathbf{J_R}$ antes da aplicação do método ILU, porém não foi encontrada nenhuma opção com resultados favoráveis. Os testes mostraram que qualquer alteração significativa na matriz $\mathbf{J_R}$ aumentava substancialmente a quantidade de iterações necessárias pelos métodos iterativos GMRES e BiCGStab, demandando maior tempo computacional.

5.7 Resultados com a Modelagem por Injeção de Corrente

Nessa seção são apresentados os testes feitos e os resultados obtidos para a resolução do PFC utilizando o Newton-Raphson por injeção de corrente através de métodos diretos e métodos iterativos para encontrar os incrementos.

Para a resolução de um PFC utilizando Newton-Raphson, pode-se utilizar injeção de corrente como foi apresentado na Subseção 3.3.3.

Os métodos direto e LU foram implementados sem problemas. Porém, os métodos que utilizam ILU com o parâmetro thresh=0 não puderam ser utilizados diretamente na matriz $\mathbf{J_I}$ dada na equação (5.15), pois há elementos nulos na diagonal principal quando existem barras PV no sistema, o que gera erros nos pivôs no cálculo das matrizes triangulares. Definido o parâmetro thresh=1, é possível realizar o processo ILU com os elementos nulos da diagonal, porém os tempos computacionais ficaram praticamente iguais ao do método LU. Logo, para eliminar os elementos nulos da diagonal principal serão propostos dois métodos:

1. Reordenamento e redução da matriz $\mathbf{J_I}$;
2. Reordenamento sem redução da matriz $\mathbf{J_I}$.

Esses métodos serão apresentados nas seções a seguir.

5.7.1 Reordenamento e Redução da Matriz Jacobiana

Utilizando o jacobiano formado pelo método de injeção de correntes, verificou-se que a diagonal apresenta alguns elementos nulos. Esses elementos comprometem os processos iterativos que utilizam os valores da diagonal principal como pivôs. Dessa forma, propõe-se um método de redução da matriz jacobiana utilizando eliminação de Kron para remover os elementos nulos das diagonais. Antes da redução é realizado um reordenamento da matriz jacobiana para facilitar a eliminação de Kron.

Primeiramente, considere que o número de barras PQ do sistema é dado por N_{PQ} e o número de barras PV por N_{PV}. Cada barra possui uma numeração, que vai de 1 até $N_{PQ}+N_{PV}+1$, pois existe a barra swing. Considerando a barra swing como a barra número 1, o vetor ref_{PQ} possui a numeração das barras PQ, subtraída uma unidade e o vetor ref_{PV} possui a numeração das barras PV também subtraída uma unidade,

pois a barra swing não é considerada na montagem do jacobiano. Por exemplo considere um sistema com 7 barras numeradas de acordo com a Tabela 5.18.

Tabela 5.18: Exemplo 7 barras

Numeração	Tipo de Barra
1	Swing
2	PQ
3	PQ
4	PV
5	PQ
6	PQ
7	PV

Logo, os vetores ref_{PQ} e ref_{PV} indicarão novas numerações dadas por:

$$ref_{PQ} = \begin{bmatrix} 2 & 3 & 5 & 6 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 4 & 5 \end{bmatrix} \tag{5.11}$$

$$ref_{PV} = \begin{bmatrix} 4 & 7 \end{bmatrix} - \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 6 \end{bmatrix} \tag{5.12}$$

Considere:

$$(n_1, n_2, \cdots, n_{pq}) \quad \epsilon \quad ref_{PQ} \tag{5.13}$$

$$(k_1, k_2, \cdots, k_{pv}) \quad \epsilon \quad ref_{PV} \tag{5.14}$$

A estrutura da matriz $\mathbf{J_I}$, como mostrado na Subseção 3.3.3, é dada pela equação (5.15).

$$\mathbf{J_I} = \left[\begin{array}{cccccc|cccccc|c}
B'_{11} & B_{12} & \cdots & B_{1k} & \cdots & B_{1n} & G'_{11} & G_{12} & \cdots & G_{1k} & \cdots & G_{1n} & 0 \\
B_{21} & B'_{22} & \cdots & B_{2k} & \cdots & B_{2n} & G_{21} & G'_{22} & \cdots & G_{2k} & \cdots & G_{2n} & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
B_{k1} & B_{k2} & \cdots & B'_{kk} & \cdots & B_{kn} & G_{k1} & G_{k2} & \cdots & G'_{kk} & \cdots & G_{kn} & \frac{V_{r_k}}{V_k^2} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
B_{n1} & B_{n2} & \cdots & B_{nk} & \cdots & B'_{nn} & G_{n1} & G_{n2} & \cdots & G_{nk} & \cdots & G'_{nn} & 0 \\
\hline
G''_{11} & G_{12} & \cdots & G_{1k} & \cdots & G_{1n} & B''_{11} & -B_{12} & \cdots & -B_{1k} & \cdots & -B_{1n} & 0 \\
G_{21} & G''_{22} & \cdots & G_{2k} & \cdots & G_{2n} & -B_{21} & B''_{22} & \cdots & -B_{2k} & \cdots & -B_{2n} & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G_{k1} & G_{k2} & \cdots & G''_{kk} & \cdots & G_{kn} & -B_{k1} & -B_{k2} & \cdots & B''_{kk} & \cdots & -B_{kn} & -\frac{V_{imag_k}}{V_k^2} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G_{n1} & G_{n2} & \cdots & G_{nk} & \cdots & G''_{nn} & -B_{n1} & -B_{n2} & \cdots & -B_{nk} & \cdots & B''_{nn} & 0 \\
\hline
0 & 0 & \cdots & \frac{V_{r_k}}{V_k} & \cdots & 0 & 0 & 0 & \cdots & \frac{V_{imag_k}}{V_k} & \cdots & 0 & 0
\end{array}\right] \tag{5.15}$$

Inicialmente, faz-se a transposição de todas as colunas das barras *PV* com elementos V_{r_k}/V_k e a transposição de todas as linhas das barras *PV* com elementos V_{r_k}/V_k^2, trocando-as pelas primeiras linhas e colunas de 1 até k_{pv}. Dessa forma, obtem-se a matriz $\mathbf{J_{I(reord)}}$ dada pela equação (5.16).

$$\mathbf{J_{I(reord)}} = \left[\begin{array}{c|ccc|ccc|ccc|c}
B'_{kk} & B_{k1} & \cdots & B_{kn} & G_{k1} & G_{k2} & \cdots & G'_{kk} & \cdots & G_{kn} & \frac{V_{r_k}}{V_k^2} \\
\hline
B_{1k} & B'_{11} & \cdots & B_{1n} & G'_{11} & G_{12} & \cdots & G_{1k} & \cdots & G_{1n} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
B_{nk} & B_{n1} & \cdots & B'_{nn} & G_{n1} & G_{n2} & \cdots & G_{nk} & \cdots & G'_{nn} & 0 \\
\hline
G_{1k} & G''_{11} & \cdots & G_{1n} & B''_{11} & -B_{12} & \cdots & -B_{1k} & \cdots & -B_{1n} & 0 \\
G_{2k} & G_{21} & \cdots & G_{2n} & -B_{21} & B''_{22} & \cdots & -B_{2k} & \cdots & -B_{2n} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G''_{kk} & G_{k1} & \cdots & G_{kn} & -B_{k1} & -B_{k2} & \cdots & B''_{kk} & \cdots & -B_{kn} & -\frac{V_{imag_k}}{V_k^2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G_{nk} & G_{n1} & \cdots & G''_{nn} & -B_{n1} & -B_{n2} & \cdots & -B_{nk} & \cdots & B''_{nn} & 0 \\
\hline
\frac{V_{r_k}}{V_k} & 0 & \cdots & 0 & 0 & 0 & \cdots & \frac{V_{imag_k}}{V_k} & \cdots & 0 & 0
\end{array}\right] \tag{5.16}$$

A transposição de linhas também modifica o vetor dos mismatches de injeção de potência e a transposição de colunas modifica o vetor de inscrementos das magnitudes das tensões reais e imaginárias. Logo, o novo sistema será dado por:

$$
\left[\begin{array}{c}
\frac{V_{imag_k}}{V_k^2}\Delta P_k \\
\Delta I_{imag_1} \\
\Delta I_{imag_2} \\
\vdots \\
\Delta I_{imag_n} \\
\hline
\Delta I_{r_1} \\
\Delta I_{r_2} \\
\vdots \\
\frac{V_{r_k}}{V_k^2}\Delta P_k \\
\vdots \\
\Delta I_{r_n} \\
\hline
\Delta V_k
\end{array}\right]
=-
\left[\begin{array}{c|ccc|ccc|ccc|c}
B'_{kk} & B_{k1} & \cdots & B_{kn} & G_{k1} & G_{k2} & \cdots & G'_{kk} & \cdots & G_{kn} & \frac{V_{r_k}}{V_k^2} \\
\hline
B_{1k} & B'_{11} & \cdots & B_{1n} & G'_{11} & G_{12} & \cdots & G_{1k} & \cdots & G_{1n} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
B_{nk} & B_{n1} & \cdots & B'_{nn} & G_{n1} & G_{n2} & \cdots & G_{nk} & \cdots & G'_{nn} & 0 \\
\hline
G_{1k} & G''_{11} & \cdots & G_{1n} & B''_{11} & -B_{12} & \cdots & -B_{1k} & \cdots & -B_{1n} & 0 \\
G_{2k} & G_{21} & \cdots & G_{2n} & -B_{21} & B''_{22} & \cdots & -B_{2k} & \cdots & -B_{2n} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G''_{kk} & G_{k1} & \cdots & G_{kn} & -B_{k1} & -B_{k2} & \cdots & B''_{kk} & \cdots & -B_{kn} & -\frac{V_{imag_k}}{V_k^2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G_{nk} & G_{n1} & \cdots & G''_{nn} & -B_{n1} & -B_{n2} & \cdots & -B_{nk} & \cdots & B''_{nn} & 0 \\
\hline
\frac{V_{r_k}}{V_k} & 0 & \cdots & 0 & 0 & 0 & \cdots & \frac{V_{imag_k}}{V_k} & \cdots & 0 & 0
\end{array}\right]
\left[\begin{array}{c}
\Delta V_{r_k} \\
\Delta V_{r_1} \\
\Delta V_{r_2} \\
\vdots \\
\Delta V_{r_n} \\
\hline
\Delta V_{imag_1} \\
\Delta V_{imag_2} \\
\vdots \\
\Delta V_{imag_k} \\
\vdots \\
\Delta V_{imag_n} \\
\hline
\Delta Q_k
\end{array}\right]
\tag{5.17}
$$

A matriz $\mathbf{J_{I(reord)}}$ pode ser escrita em função de suas submatrizes como:

$$
\mathbf{J_{I(reord)}} = \left[\begin{array}{c|c|c|c}
B_{kk} & B_{prim_{kh}} & G_{prim_{kh}} & D_2 \\
\hline
B_{prim_{kv}} & B_{prim_{mod}} & G_{prim_{mod}} & 0 \\
\hline
G_{sec_{kv}} & G_{sec_{mod}} & B_{sec} & V_{imagv} \\
\hline
D_1 & 0 & V_{imagh} & 0
\end{array}\right]
\tag{5.18}
$$

As submatrizes de $\mathbf{J_{I(reord)}}$ podem ser definidas como:

$$
B_{kk} = \begin{bmatrix}
B'_{k_1,k_1} & \cdots & B_{k_1,k_{pv}} \\
\vdots & \ddots & \vdots \\
B_{k_{pv},k_1} & \cdots & B'_{k_{pv},k_{pv}}
\end{bmatrix}
\tag{5.19}
$$

$$
B_{prim_{kh}} = \begin{bmatrix}
B_{k_1,n_1} & \cdots & B_{k_1,n_{pq}} \\
\vdots & \ddots & \vdots \\
B_{k_{pv},n_1} & \cdots & B_{k_{pv},n_{pq}}
\end{bmatrix}
\tag{5.20}
$$

$$
B_{prim_{kv}} = \begin{bmatrix}
B_{n_1,k_1} & \cdots & B_{n_1,k_{pv}} \\
\vdots & \ddots & \vdots \\
B_{n_{pq},k_1} & \cdots & B_{n_{pq},k_{pv}}
\end{bmatrix}
\tag{5.21}
$$

$$
B_{prim_{mod}} = \begin{bmatrix}
B'_{n_1,n_1} & \cdots & B_{n_1,n_{pq}} \\
\vdots & \ddots & \vdots \\
B_{n_{pq},n_1} & \cdots & B'_{n_{pq},n_{pq}}
\end{bmatrix}
\tag{5.22}
$$

$$
G_{prim_{kh}} = \begin{bmatrix}
G_{k_1,1} & \cdots & G_{k_1,n} \\
\vdots & \ddots & \vdots \\
G_{k_{pv},1} & \cdots & G_{k_{pv},n}
\end{bmatrix}
\tag{5.23}
$$

$$G_{prim_{mod}} = \begin{bmatrix} G'_{n_1,1} & \cdots & G_{n_1,n} \\ \vdots & \ddots & \vdots \\ G_{n_{pq},1} & \cdots & G''_{n_{pq},n} \end{bmatrix} \tag{5.24}$$

$$G_{sec_{kv}} = \begin{bmatrix} G_{1,k_1} & \cdots & G_{1,k_{pv}} \\ \vdots & \ddots & \vdots \\ G_{n,k_1} & \cdots & G_{n,k_{pv}} \end{bmatrix} \tag{5.25}$$

$$G_{sec_{mod}} = \begin{bmatrix} G'_{1,n_1} & \cdots & G_{1,n_{pq}} \\ \vdots & \ddots & \vdots \\ G_{n,1} & \cdots & G''_{n,n_{pq}} \end{bmatrix} \tag{5.26}$$

$$D_1 = \begin{bmatrix} \frac{V_{r_{k_1}}}{V_{k_1}} & 0 & \cdots & 0 \\ 0 & \frac{V_{r_{k_2}}}{V_{k_2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{V_{r_{k_{pv}}}}{V_{k_{pv}}} \end{bmatrix} \tag{5.27}$$

$$D_2 = \begin{bmatrix} \frac{V_{r_{k_1}}}{V_{k_1}^2} & 0 & \cdots & 0 \\ 0 & \frac{V_{r_{k_2}}}{V_{k_2}^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{V_{r_{k_{pv}}}}{V_{k_{pv}}^2} \end{bmatrix} \tag{5.28}$$

O reordenamento foi feito para facilitar a execução da redução de Kron [21]. A matriz $\mathbf{J_{I(reord)}}$ ainda apresenta elementos nulos na diagonal. Após o reordenamento, é feita a redução da matriz. Inicialmente, são utilizadas as matrizes auxiliares J_1, J_2, J_3 e J_4:

$$\mathbf{J_{I(reord)}} = \left[\begin{array}{c|c} J_1 & J_2 \\ \hline J_3 & J_4 \end{array}\right] \tag{5.29}$$

$$J_1 = \begin{bmatrix} B_{kk} \\ \hline B_{prim_{kv}} \\ \hline G_{sec_{kv}} \end{bmatrix} \tag{5.30}$$

$$J_2 = \left[\begin{array}{c|c|c} B_{prim_{kh}} & G_{primkh} & D_2 \\ \hline B_{prim_{mod}} & G_{prim_{mod}} & 0 \\ \hline G_{sec_{mod}} & B_{sec} & V_{imagv} \end{array}\right] \tag{5.31}$$

$$J_3 = \Big[D1 \Big] \tag{5.32}$$

$$J_4 = \left[\begin{array}{c|c|c} 0 & V_{imagh} & 0 \end{array} \right] \tag{5.33}$$

A redução utilizando a eliminação de Kron é feita em duas etapas. Na primeira etapa, eliminam-se as linhas e colunas adjacentes à submatriz D_1. Logo, são eliminadas as matrizes J_1, J_3 e J_4. Pelo método de Kron, é necessário modificar todos os elementos da matriz J_2 remanescente, mas utilizando a matriz J_4 como pivo, apenas os elementos das colunas adjacentes à V_{imagh} serão modificados, pois os outros serão multiplicados pelos elementos nulos de J_4. A primeira etapa da redução é feita utilizando a seguinte equação matricial:

$$J_2^{new} = J_2 - J_1[(J_3)^{-1} J_4] \tag{5.34}$$

J_2^{new} passa a ser uma matriz com dimensões $(2N_{PQ} + 2N_{PV}) \times (2N_{PQ} + 2N_{PV})$ dada por:

$$J_2^{new} = \left[\begin{array}{c|c|c} B_{prim_{kh}} & G_{prim_{kh}}^{new} & D_2 \\ \hline B_{prim_{mod}} & G_{prim_{mod}}^{new} & 0 \\ \hline G_{sec_{mod}} & B_{sec}^{new} & V_{imagv} \end{array} \right] \tag{5.35}$$

Também deve ser feito a redução da matriz dos mismatches, dada por:

$$b_{I_{reord}} = \left[\begin{array}{c} \frac{V_{imag_k}}{V_k^2} \Delta P_k \\ \Delta I_{imag_1} \\ \Delta I_{imag_2} \\ \vdots \\ \Delta I_{imag_n} \\ \hline \Delta I_{r_1} \\ \Delta I_{r_2} \\ \vdots \\ \frac{V_{r_k}}{V_k^2} \Delta P_k \\ \vdots \\ \Delta I_{r_n} \\ \hline \Delta V_k \end{array} \right] \tag{5.36}$$

Pode-se dividir a matriz de mismatches em submatrizes B_1 e B_2 dadas por:

$$b_{I_{reord}} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \tag{5.37}$$

$$B_1 = \begin{bmatrix} \frac{V_{imag_k}}{V_k^2}\Delta P_k \\ \Delta I_{imag_1} \\ \Delta I_{imag_2} \\ \vdots \\ \Delta I_{imag_n} \\ \hline \Delta I_{r_1} \\ \Delta I_{r_2} \\ \vdots \\ \frac{V_{r_k}}{V_k^2}\Delta P_k \\ \vdots \\ \Delta I_{r_n} \end{bmatrix} \tag{5.38}$$

$$B_2 = \left[\Delta V_k\right] \tag{5.39}$$

A eliminação de Kron é feita utilizando a seguinte equação matricial:

$$B_1^{new} = B_1 - J_1[(J_3)^{-1}B_2] \tag{5.40}$$

B_1^{new} passa a ser uma matriz com dimensões $(2N_{PQ} + 2N_{PV}) \times (1)$. Na segunda etapa da redução, eliminam-se as linhas e colunas adjacentes à submatriz D_2. Pode-se subdividir a matriz J_2^{new} em submatrizes auxiliares J_5, J_6, J_7 e J_8:

$$J_2^{new} = \left[\begin{array}{c|c} J_5 & J_6 \\ \hline J_7 & J_8 \end{array}\right] \tag{5.41}$$

$$J_5 = \left[\begin{array}{c|c} B_{prim_{kh}} & G_{prim_{kh}}^{new} \end{array}\right] \tag{5.42}$$

$$J_6 = \left[\begin{array}{c} D_2 \end{array}\right] \tag{5.43}$$

$$J_7 = \left[\begin{array}{c|c} B_{prim_{mod}} & G_{prim_{mod}}^{new} \\ \hline G_{sec_{mod}} & B_{sec}^{new} \end{array}\right] \tag{5.44}$$

$$J_8 = \left[\begin{array}{c} 0 \\ \hline V_{imagv} \end{array}\right] \tag{5.45}$$

Serão eliminadas as matrizes J_5, J_6 e J_8. Pelo método de Kron, é necessário modificar todos os elementos da matriz J_7 remanescente, mas utilizando a matriz J_8 como pivô. Apenas os elementos das linhas adjacentes à V_{imagv} serão modificados, pois os outros serão multiplicados pelos elementos nulos de J_8. A segunda etapa da redução é feita utilizando a seguinte equação matricial:

$$J_7^{new} = J_7 - J_8[(J_6)^{-1}J_5] \tag{5.46}$$

J_7^{new} é a nova matriz jacobiana por injeção de corrente reordenada e reduzida $\mathbf{J_{I_{red}}}$. Ela passa a ser uma matriz com dimensões $(2N_{PQ}+N_{PV}) \times (2N_{PQ}+N_{PV})$ dada por:

$$\mathbf{J_{I_{red}}} = \left[\begin{array}{c|c} B_{prim_{mod}} & G_{prim_{mod}}^{new} \\ \hline G_{sec_{mod}}^{new} & B_{sec}^{new_2} \end{array}\right] \tag{5.47}$$

Também deve ser feita a redução da matriz dos mismatches B_1^{new}. Pode-se dividir a matriz B_1^{new} em submatrizes B_3 e B_4 dadas por:

$$B_1^{new} = \begin{bmatrix} B_3 \\ B_4 \end{bmatrix} \tag{5.48}$$

$$B_3 = \left[\frac{V_{imag_k}}{V_k^2}\Delta P_k\right] \tag{5.49}$$

$$B_4 = \left[\begin{array}{c} \Delta I_{imag_1} \\ \Delta I_{imag_2} \\ \vdots \\ \Delta I_{imag_n} \\ \hline \Delta I_{r_1} \\ \Delta I_{r_2} \\ \vdots \\ \frac{V_{r_k}}{V_k^2}\Delta P_k \\ \vdots \\ \Delta I_{r_n} \end{array}\right] \tag{5.50}$$

A segunda parte da eliminação de Kron é feita utilizando a seguinte equação matricial:

$$B_4^{new} = B_4 - J_8[(J_6)^{-1}B_3] \tag{5.51}$$

B_4^{new} é a nova matriz de mismatches por injeção de corrente reordenada e reduzida $b_{I_{red}}$. Ela passa a ser uma matriz com dimensões $(2N_{PQ}+N_{PV}) \times (1)$ dada por:

$$b_{I_{red}} = \begin{bmatrix} \Delta I_{imag_1}^{new} \\ \Delta I_{imag_2}^{new} \\ \vdots \\ \Delta I_{imag_n}^{new} \\ \hline \Delta I_{r_1}^{new} \\ \Delta I_{r_2}^{new} \\ \vdots \\ \left(\frac{V_{r_k}}{V_k^2}\Delta P_k\right)^{new} \\ \vdots \\ \Delta I_{r_n}^{new} \end{bmatrix} \tag{5.52}$$

O vetor das variáveis também é modificado. Com a eliminação das colunas na matriz Jacobiana, os elementos ΔV_{r_k} e ΔQ_k são eliminados. Logo, $x_{I_{red}}$ é dado por:

$$x_{I_{red}} = \begin{bmatrix} \Delta V_{r_{n_1}} \\ \Delta V_{r_{n_1}} \\ \vdots \\ \Delta V_{r_{npq}} \\ \hline \Delta V_{imag_1} \\ \Delta V_{imag_2} \\ \vdots \\ \Delta V_{imag_k} \\ \vdots \\ \Delta V_{imag_n} \end{bmatrix} \tag{5.53}$$

Divide-se o vetor a matriz x em duas submatrizes X_1 e X_2 dadas por:

$$x_{I_{red}} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{5.54}$$

$$X_1 = \begin{bmatrix} \Delta V_{r_{n_1}} \\ \Delta V_{r_{n_1}} \\ \vdots \\ \Delta V_{r_{npq}} \end{bmatrix} \tag{5.55}$$

$$X_2 = \begin{bmatrix} \Delta V_{imag_1} \\ \Delta V_{imag_2} \\ \vdots \\ \Delta V_{imag_k} \\ \vdots \\ \Delta V_{imag_n} \end{bmatrix} \tag{5.56}$$

Os incrementos para o novo sistema podem ser encontrados como:

$$x_{I_{red}} = -(\mathbf{J_{I_{red}}})^{-1} b_{I_{red}} \tag{5.57}$$

Percebe-se que os valores de ΔQ_k e ΔV_{r_k} não são encontrados diretamente pela equação (5.57). Não é necessário deteminar ΔQ_k, pois são apenas variáveis auxiliares que não interferem na solução das partes reais e imaginárias das tensões nas barras. No entanto, é necessário obter o valor de ΔV_{r_k} para encontrar a solução completa do sistema. Esse valor pode ser obtido por:

$$\Delta V_{r_k} = -(J_3)^{-1} V_{imagh} X_2 + B_2 \tag{5.58}$$

O método mostrado acima permite resolver o PFC por injeção de corrente utilizando métodos iterativos, pois a matriz Jacobiana reordenada reduzida resultante $\mathbf{J_{I_{red}}}$ não possui elementos nulos na diagonal principal. Além disso, sua dimensão $(2N_{PQ} + N_{PV}) \times (2N_{PQ} + N_{PV})$ é a mesma da matriz por coordenadas polares.

Para ilustrar os método acima, pode-se observar as estruturas da matriz jacobiana para o caso com 30 barras, com 24 barras PQ e 5 barras PV. A dimensão da matriz $\mathbf{J_I}$ será de $(2N_{PQ} + 3N_{PV}) \times (2N_{PQ} + 3N_{PV}) = (63) \times (63)$. Inicialmente, a estrutura da matriz $\mathbf{J_I}$, apresentada na equação (5.15), é dada pela Figura 5.17. Vê-se a presença de elementos nulos na diagonal principal. Após o reordenamento, a estrutura da matriz $\mathbf{J_{I(reord)}}$, apresentada na equação (5.18), é dada pela Figura 5.18, na qual pode-se observar a formação das duas matrizes diagonais D_1 e D_2, apresentadas nas equações (5.27) e (5.28).

Em seguida, é realizada a primeira etapa da eliminação de Kron descrita anteriormente. A estrutura da matriz J_2^{new}, apresentada na equação (5.35), é dada pela Figura 5.19. Pode-se observar que foram eliminadas as submatrizes J_5, J_6 e J_8 apresentadas na equação (5.41).

Por fim, é realizada a segunda etapa da eliminação de Kron, resultando na matriz $\mathbf{J_{I_{red}}}$ dada pela equação (5.47) e apresentada na Figura 5.20. Vê-se que foram eliminadas as submatrizes J_5, J_6 e J_8 apresentadas na equação (5.29). Observa-se que não há elementos nulos na diagonal principal. Além disso, o número de elementos não nulos da matriz passou de 420 para 359, porém ela ainda possui um alto grau de esparsidade. As novas dimensões da matriz são dadas por $(2N_{PQ} + N_{PV}) \times (2N_{PQ} + N_{PV}) = (53) \times (53)$.

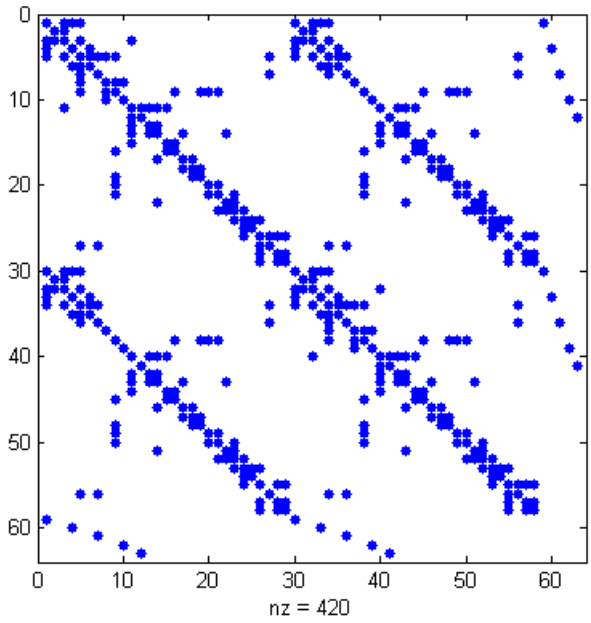


Figura 5.17: Estrutura da matriz jacobiana por injeção de corrente para caso com 30 barras

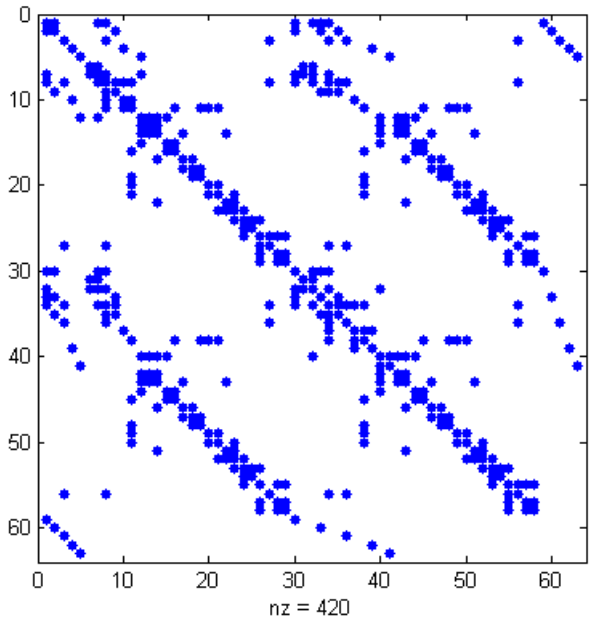


Figura 5.18: Estrutura da matriz jacobiana por injeção de corrente para caso com 30 barras após o reordenamento

5.7.2 Reordenamento sem Redução da Matriz Jacobiana

É proposta outra solução para eliminar os elementos nulos da diagonal principal da matriz jacobiana por injeção de corrente. Nesse método será utilizado apenas o reordenamento, sem ser necessário realizar a eliminação de Kron na matriz.

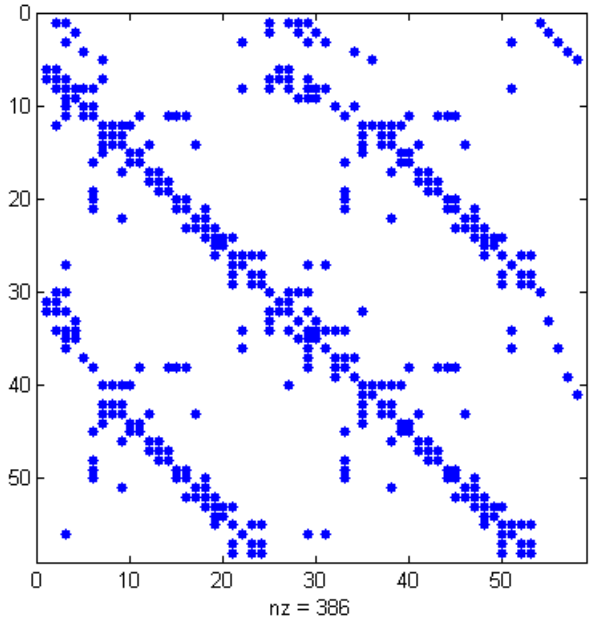


Figura 5.19: Estrutura da matriz jacobiana por injeção de corrente para caso com 30 barras após o reordenamento e primeira etapa da redução

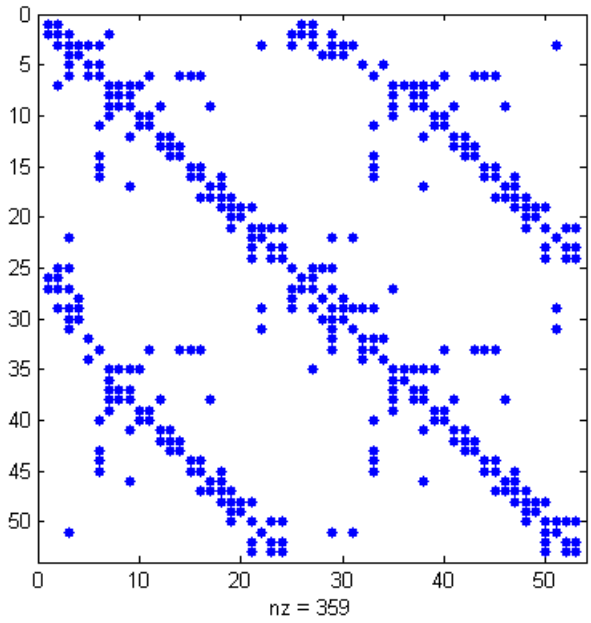


Figura 5.20: Estrutura da matriz jacobiana por injeção de corrente para caso com 30 barras reordenada e reduzida

Considere novamente a matriz $\mathbf{J_I}$ dada em (5.15). As colunas com os elementos V_{r_k}/V_k das barras *PV* são transpostas com as colunas correspondentes onde há elemetos V_{r_k}/V_k^2. Dessa forma, obtém-se a matriz dada na equação (5.59).

$$\mathbf{J_{I_{reord}}} = \left[\begin{array}{ccccccc|ccccc|c}
B'_{11} & B_{12} & \cdots & 0 & \cdots & B_{1n} & G'_{11} & G_{12} & \cdots & G_{1k} & \cdots & G_{1n} & B_{1k} \\
B_{21} & B'_{22} & \cdots & 0 & \cdots & B_{2n} & G_{21} & G'_{22} & \cdots & G_{2k} & \cdots & G_{2n} & B_{2k} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
B_{k1} & B_{k2} & \cdots & \frac{V_{r_k}}{V_k^2} & \cdots & B_{kn} & G_{k1} & G_{k2} & \cdots & G'_{kk} & \cdots & G_{kn} & B'_{kk} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
B_{n1} & B_{n2} & \cdots & 0 & \cdots & B'_{nn} & G_{n1} & G_{n2} & \cdots & G_{nk} & \cdots & G'_{nn} & B_{nk} \\
\hline
G''_{11} & G_{12} & \cdots & 0 & \cdots & G_{1n} & B''_{11} & -B_{12} & \cdots & -B_{1k} & \cdots & -B_{1n} & G_{1k} \\
G_{21} & G''_{22} & \cdots & 0 & \cdots & G_{2n} & -B_{21} & B''_{22} & \cdots & -B_{2k} & \cdots & -B_{2n} & G_{2k} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G_{k1} & G_{k2} & \cdots & -\frac{V_{imag_k}}{V_k^2} & \cdots & G_{kn} & -B_{k1} & -B_{k2} & \cdots & B''_{kk} & \cdots & -B_{kn} & G''_{kk} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G_{n1} & G_{n2} & \cdots & 0 & \cdots & G''_{nn} & -B_{n1} & -B_{n2} & \cdots & -B_{nk} & \cdots & B''_{nn} & G_{nk} \\
\hline
0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & \frac{V_{imag_k}}{V_k} & \cdots & 0 & \frac{V_{r_k}}{V_k}
\end{array}\right] \tag{5.59}$$

Como foram alteradas apenas as colunas da matriz jacobiana, a matriz dos mismatches não é alterada. Porém, a matriz dos incrementos deve ser reordenada. Dessa forma, o sistema final fica:

$$\left[\begin{array}{c}
\Delta I_{imag_1} \\
\Delta I_{imag_2} \\
\vdots \\
\frac{V_{imag_k}}{V_k^2}\Delta P_k \\
\vdots \\
\Delta I_{imag_n} \\
\hline
\Delta I_{r_1} \\
\Delta I_{r_2} \\
\vdots \\
\frac{V_{r_k}}{V_k^2}\Delta P_k \\
\vdots \\
\Delta I_{r_n} \\
\hline
\Delta V_k
\end{array}\right] = -\left[\begin{array}{ccccccc|ccccc|c}
B'_{11} & B_{12} & \cdots & 0 & \cdots & B_{1n} & G'_{11} & G_{12} & \cdots & G_{1k} & \cdots & G_{1n} & B_{1k} \\
B_{21} & B'_{22} & \cdots & 0 & \cdots & B_{2n} & G_{21} & G'_{22} & \cdots & G_{2k} & \cdots & G_{2n} & B_{2k} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
B_{k1} & B_{k2} & \cdots & \frac{V_{r_k}}{V_k^2} & \cdots & B_{kn} & G_{k1} & G_{k2} & \cdots & G'_{kk} & \cdots & G_{kn} & B'_{kk} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
B_{n1} & B_{n2} & \cdots & 0 & \cdots & B'_{nn} & G_{n1} & G_{n2} & \cdots & G_{nk} & \cdots & G'_{nn} & B_{nk} \\
\hline
G''_{11} & G_{12} & \cdots & 0 & \cdots & G_{1n} & B''_{11} & -B_{12} & \cdots & -B_{1k} & \cdots & -B_{1n} & G_{1k} \\
G_{21} & G''_{22} & \cdots & 0 & \cdots & G_{2n} & -B_{21} & B''_{22} & \cdots & -B_{2k} & \cdots & -B_{2n} & G_{2k} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G_{k1} & G_{k2} & \cdots & -\frac{V_{imag_k}}{V_k^2} & \cdots & G_{kn} & -B_{k1} & -B_{k2} & \cdots & B''_{kk} & \cdots & -B_{kn} & G''_{kk} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
G_{n1} & G_{n2} & \cdots & 0 & \cdots & G''_{nn} & -B_{n1} & -B_{n2} & \cdots & -B_{nk} & \cdots & B''_{nn} & G_{nk} \\
\hline
0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & \cdots & \frac{V_{imag_k}}{V_k} & \cdots & 0 & \frac{V_{r_k}}{V_k}
\end{array}\right] \left[\begin{array}{c}
\Delta V_{r_1} \\
\Delta V_{r_2} \\
\vdots \\
\Delta Q_k \\
\vdots \\
\Delta V_{r_n} \\
\hline
\Delta V_{imag_1} \\
\Delta V_{imag_2} \\
\vdots \\
\Delta V_{imag_k} \\
\vdots \\
\Delta V_{imag_n} \\
\hline
\Delta V_{r_k}
\end{array}\right] \tag{5.60}$$

Consequentemente, os elementos V_{r_k}/V_k^2 e V_{r_k}/V_k estarão sempre na diagonal principal da nova matriz Jacobiana, eliminando os elementos nulos que geravam problemas nos métodos iterativos.

Para ilustrar os método acima, pode-se observar as estruturas da matriz jacobiana para o caso com 30 barras, com 24 barras PQ e 5 barras PV. A estrutura da matriz $\mathbf{J_I}$, apresentada na equação (5.15), é dada pela Figura 5.17, onde percebe-se a existência de elementos nulos na diagonal principal. Após o reordenamento descrito anteriormente, a matriz $\mathbf{J_{I_{reord}}}$, apresentada na equação (5.59), terá a estrutura dada

pela Figura 5.21. Observa-se que não existem mais elementos nulos na diagonal principal e a dimensão da matriz não se altera.

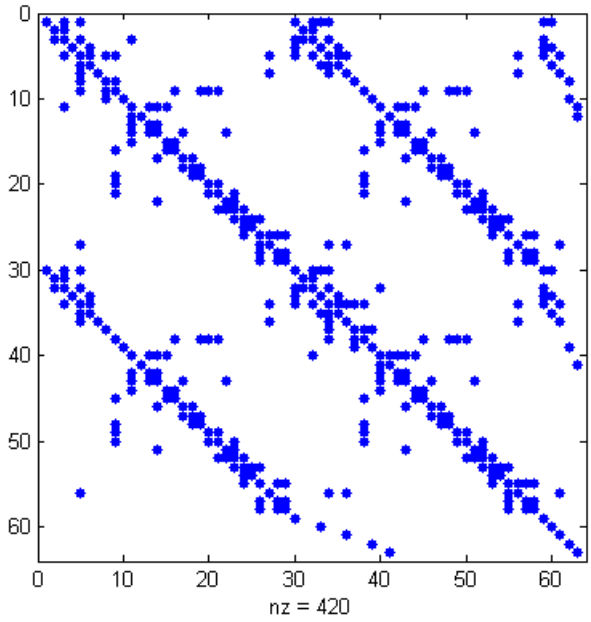


Figura 5.21: Estrutura Matriz Jacobiana por injeção de corrente para Caso com 30 barras reordenada sem redução

5.7.3 Métodos Diretos

Foram analisados quatro diferentes métodos diretos, sendo o primeiro utilizando o cálculo pela inversa do jacobiano, o segundo utilizando o método LU, em seguida utilizando o método ILU apresentado na Subseção 2.4.3.1 com o parâmetros droptol=0, thresh=0, milu='off' e type='ilutp' e por fim utilizando a expansão linear apresentada na Subseção 3.3.4. O método por expansão linear não convergiu utilizando reordenamento com redução da matriz $\mathbf{J_I}$. Logo, ele foi analisado apenas utilizando o reordenamento sem redução da matriz $\mathbf{J_I}$. Para a aplicação dos métodos LU e ILU, foram utilizados os reordenamentos RCM e AMD. Para a expansão linear, foi utilizado o reordenamento AMD. Os resultados obtidos podem ser observados na Tabela 5.19.

Observa-se que, da mesma forma como para coordenadas polares e retangulares, o método direto utilizando ILU e AMD para o reordenamento sem redução foi o com melhor desempenho. Os métodos utilizando RCM tiveram os piores desempenhos. O reordenamento sem redução apresentou em geral melhores resultados do que o reordenamento com redução. Um dos motivos pode ser que apesar da matriz J_I ter maiores dimensões para o reordenamento sem redução, os cálculos para realização da redução de Kron são evitados, reduzindo o tempo computacional.

Tabela 5.19: Tempos médios para métodos diretos de solução do PFC por injeção de corrente

Modificação $\mathbf{J_I}$	Método	Reordenamento	Tempo Solução PFC[s]
-	Direto	-	0,1269
-	LU	RCM	0,3774
		AMD	0,1568
Reordenamento com Redução	ILU	RCM	0,2143
		AMD	0,1083
Reordenamento sem Redução	ILU	RCM	0,2788
		AMD	**0,0918**
	Expansão Linear	AMD	0,1197

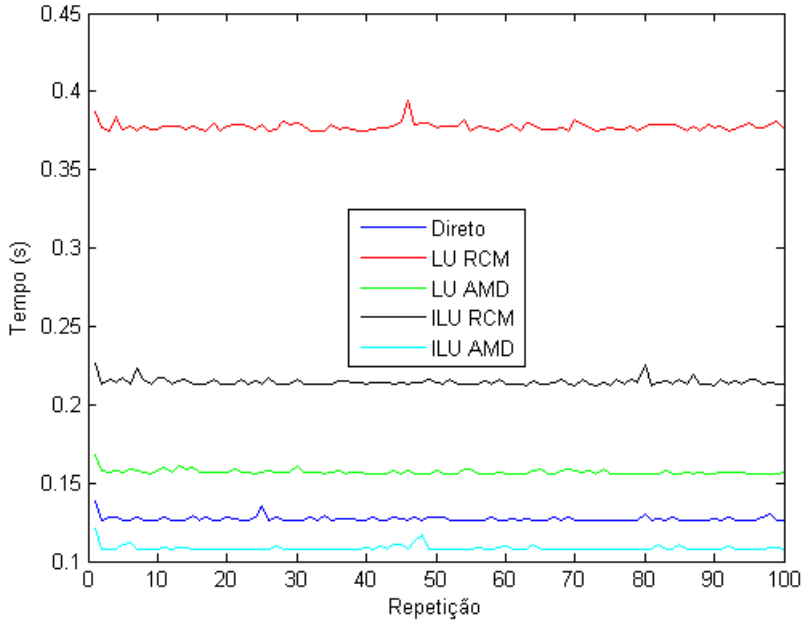


Figura 5.22: Tempos computacionais para métodos diretos por injeção de corrente com reordenamento e redução

5.7.4 Métodos Iterativos

Foi analisado o PFC com 3375 barras utilizando diferentes métodos iterativos para injeção de corrente, apresentados na Tabela 5.9.

Foram testados diferentes tipos de pré-condicionadores e parâmetros da função ILU presente no MATLAB. Os resultados obtidos serão apresentados e discutidos a seguir.

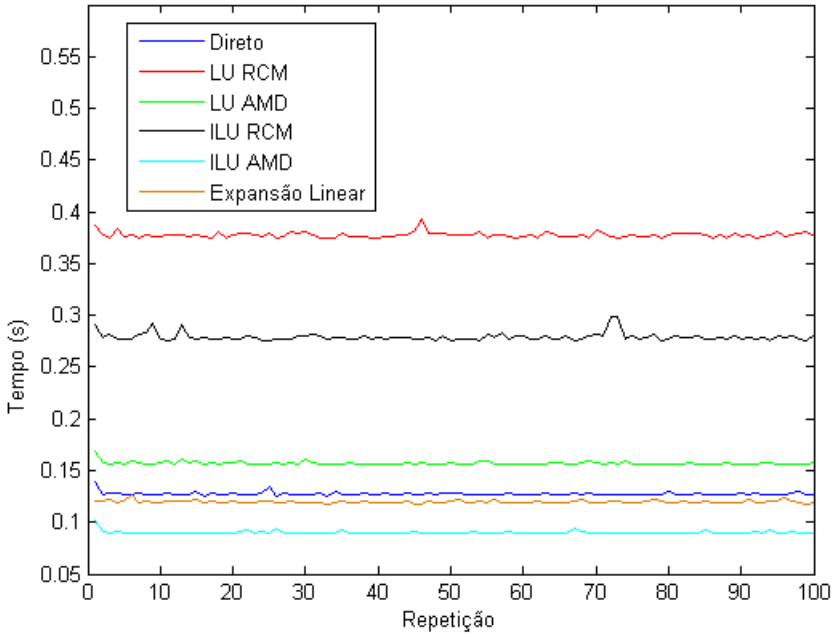


Figura 5.23: Tempos computacionais para métodos diretos por injeção de corrente com reordenamento sem redução

5.7.4.1 Pré-Condicionador Variável

Nessa seção são apresentados os resultados obtidos por injeção de corrente utilizando o pré-condicionador variável baseado na matriz jacobiana $\mathbf{J_I}$ completa pelo método ILUTP modificado apresentado na Subseção 2.4.3.1. O reordenamento e o pré-condicionador foram efetuados a cada nova iteração após a obtenção da nova matriz jacobiana. Foram analisados os quatro métodos iterativos apresentados na Tabela 5.9.

Os parâmetros utilizados para a função ILU do MATLAB foram droptol=0, thresh=0, milu='off' e type='ilutp'.

Tabela 5.20: Tempos médios para métodos iterativos de solução do PFC por injeção de corrente com pré-condicionador variável

Modificação $\mathbf{J_I}$	Reordenamento	Método	Tempo Solução PFC[s]
-	-	Direto	0,1269
Reordenamento com Redução	RCM	GMRES	0,2082
		BiCGStab	0,2171
	AMD	GMRES	**0,1003**
		BiCGStab	0,1143
Reordenamento sem Redução	RCM	GMRES	0,2293
		BiCGStab	0,2484
	AMD	GMRES	0,1514
		BiCGStab	0,1792

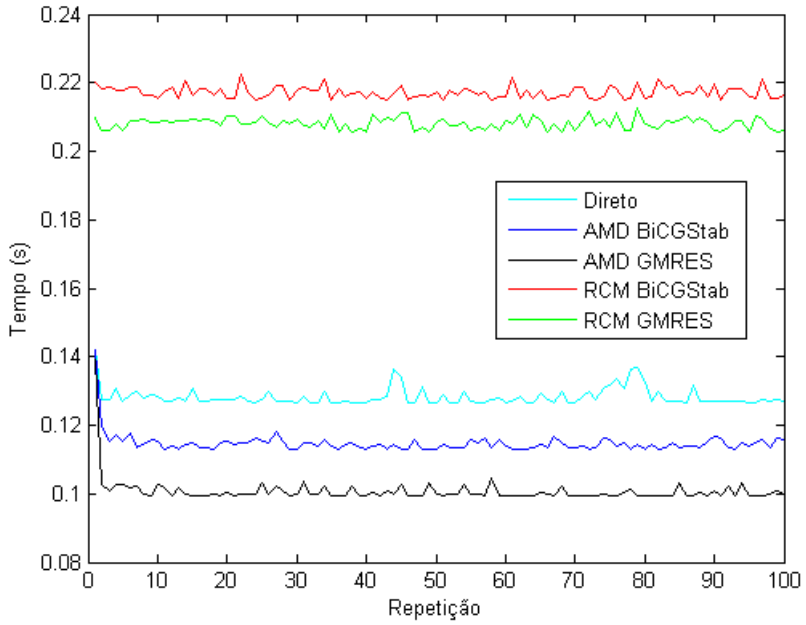


Figura 5.24: Tempos computacionais para métodos injeção de corrente reordenado e reduzido com pré-condicionador variável

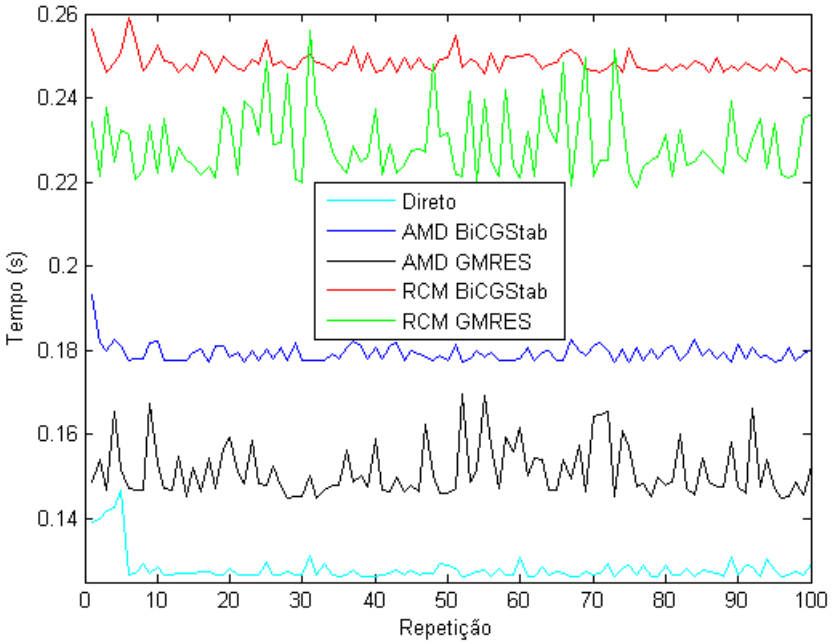


Figura 5.25: Tempos computacionais para métodos injeção de corrente reordenado sem redução com pré-condicionador variável

Utilizando o pré-condicionador variável, o reordenamento com redução apresentou melhores resultados do que o reordenamento sem redução. Os resultados utilizando GMRES foram melhores do que os utilizando BiCGStab. O reordenamento AMD apresentou desempenho superior ao reordenamento RCM.

O melhor desempenho foi para o reordenamento com redução utilizando AMD e GMRES.

5.7.4.2 Pré-Condicionador Fixo

Nessa seção são apresentados os resultados obtidos por injeção de corrente utilizando o pré-condicionador fixo baseado na matriz jacobiana $\mathbf{J_I}$ completa pelo método ILUTP modificado apresentado na Subseção 2.4.3.1. O reordenamento e o pré-condicionador foram calculados apenas na primeira iteração e utilizados nas iterações subsequentes. Foram analisados os quatro métodos iterativos apresentados na Tabela 5.9.

Os parâmetros utilizados para a função ILU do MATLAB foram droptol=0, thresh=0, milu='off' e type='ilutp'.

Tabela 5.21: Tempos médios para métodos iterativos de solução do PFC por injeção de corrente com pré-condicionador fixo

Modificação $\mathbf{J_I}$	Reordenamento	Método	Tempo Solução PFC[s]
-	-	Direto	0,1269
Reordenamento com Redução	RCM	GMRES	0,1979
		BiCGStab	0,2147
	AMD	GMRES	0,1441
		BiCGStab	0,1510
Reordenamento sem Redução	RCM	GMRES	0,1559
		BiCGStab	0,1679
	AMD	GMRES	**0,0908**
		BiCGStab	0,0987

Os resultados para o reordenamento sem redução tiveram melhores desempenhos utilizando o pré-condicionador fixo. No entanto, os resultados para o reordenamento com redução tiveram piores desempenhos em comparação com o pré-condicionador variável. O melhor resultado obtido para os métodos por injeção de corrente foi para o reordenamento sem redução utilizando pré-condicionador fixo, AMD e GMRES. Os motivos podem ser a menor quantidade de cálculos internos para modificação da matriz $\mathbf{J_I}$ e o pré-condicionador fixo ter garantido a convergência dos método iterativos, sem ser necessário utilizar tempo computacional para calcular novos valores para o pré-condicionador nas iterações subsequentes. Para o reordenamento com redução, o pré-condicionador fixo não apresentou resultados satisfatórios, possivelmente devido a quantidade de modificações realizada na estrutura original da matriz $\mathbf{J_I}$.

5.7.4.3 Outros Tipos de Pré-Condicionadores

Nos métodos iterativos utilizando coordenadas polares apresentados anteriormente, foram utilizados pré-condicionadores gerados pelo método ILU baseados na matriz Jacobiana $\mathbf{J_I}$ completa. Foram analisados outras três opções de pré-condicionadores, realizando o método ILU na matriz $\mathbf{J_I}$, apresentada na equação (3.45), modificada como:

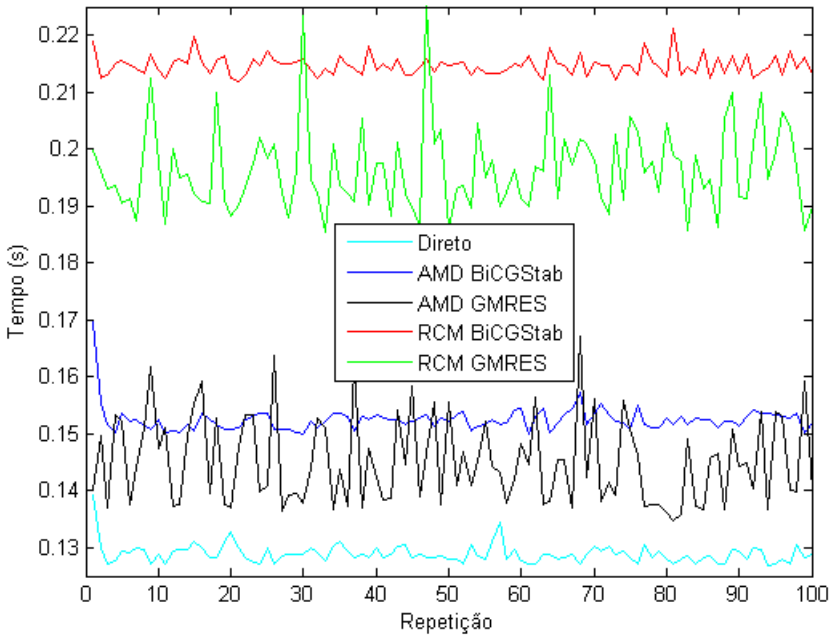


Figura 5.26: Tempos computacionais para métodos injeção de corrente reordenado e reduzido com pré-condicionador fixo

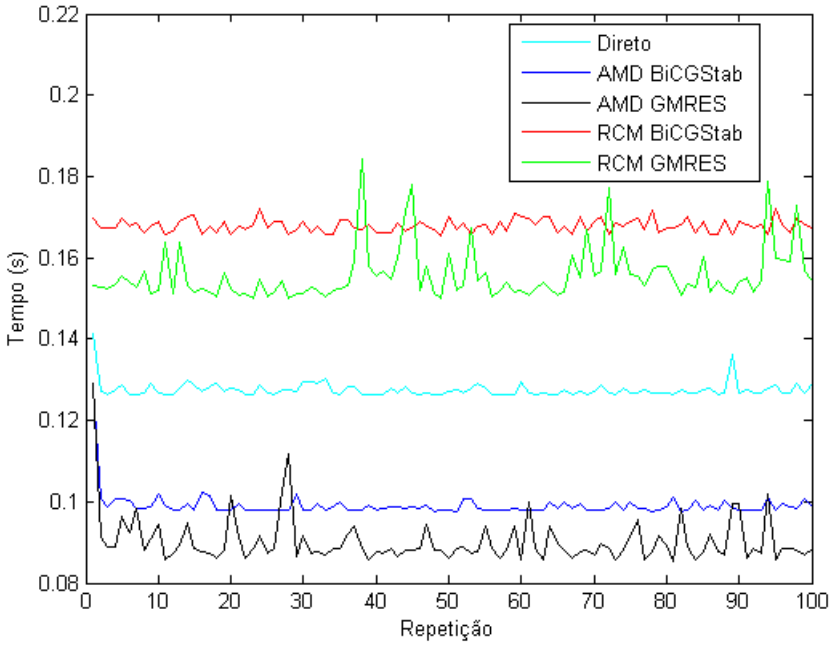


Figura 5.27: Tempos computacionais para métodos injeção de corrente reordenado sem redução com pré-condicionador fixo

1. T_1 zerando as duas submatrizes G_{prim} e G_{sec}, logo:

$$T_1 = \left[\begin{array}{c|c|c} B_{prim} & 0 & V_{rv} \\ \hline 0 & B_{sec} & V_{imagv} \\ \hline V_{rh} & V_{imagh} & 0 \end{array}\right] \tag{5.61}$$

2. T_2 zerando a submatriz G_{prim}, logo:

$$T_2 = \left[\begin{array}{c|c|c} B_{prim} & 0 & V_{rv} \\ \hline G_{sec} & B_{sec} & V_{imagv} \\ \hline V_{rh} & V_{imagh} & 0 \end{array}\right] \tag{5.62}$$

3. T_3 zerando a submatriz G_{sec}, logo:

$$T_3 = \left[\begin{array}{c|c|c} B_{prim} & G_{prim} & V_{rv} \\ \hline 0 & B_{sec} & V_{imagv} \\ \hline V_{rh} & V_{imagh} & 0 \end{array}\right] \tag{5.63}$$

Os pré-condicionadores e os parâmetros do reordenamento foram calculados apenas na primeira iteração e foram fixados nas iterações subsequentes. Os parâmetros utilizados para a função ILU do MATLAB foram droptol=0, thresh=0, milu='off' e type='ilutp'.

Tabela 5.22: Tempos médios dos métodos para solução do PFC por MNR por injeção de corrente para diferentes pré-condicionadores

Modificação $\mathbf{J_I}$	Reordenamento	Método	Tempo Solução PFC[s]			
			$\mathbf{J_I}$	T_1	T_2	T_3
Reordenamento com redução	RCM	GMRES	0,2082	0,2663	0,2089	0,2075
		BiCGStab	0,2171	0,3728	0,2661	0,2707
	AMD	GMRES	0,1003	0,1805	0,1407	0,1441
		BiCGStab	0,1143	0,2470	0,1879	0,1890
Reordenamento sem redução	RCM	GMRES	0,1559	0,2454	0,1783	0,2050
		BiCGStab	0,1679	0,2914	0,2110	0,2037
	AMD	GMRES	**0,0908**	0,1479	0,1033	0,1229
		BiCGStab	0,0987	0,1639	0,1254	0,1209

Os resultados obtidos para os pré-condicionadores T_1, T_2, e T_3 foram piores do que os obtidos utilizando a matriz J_I completa. Os melhores resultados entre os diferentes tipos de pré-condicionadores foram para o pré-condicionador T_2.

5.8 Comparação dos Resultados

Nessa seção, são selecionados e comparados os melhores resultados obtidos pelos diferentes métodos de resolução do PFC para o caso com 3775 barras. Também são analisados sistemas de ordem superior ao de 3375 barras para avaliar seus desempenhos, além de análises de sensibilidade para o parâmetro *droptol*.

5.8.1 Melhores Resultados

Foram analisados os melhores resultados obtidos para cada modelagem do PFC, considerando os métodos diretos e os métodos iterativos. Para os métodos iterativos foram utilizandos pré-condicionadores fixos baseados nas matrizes jacobianas completas. Os melhores resultados estão apresentados na Tabela 5.23. Também é mostrado o resultado obtido utilizando o método por coordenadas polares direto pela inversa do jacobiano, pois é o método de referência utilizado no MATPOWER.

Tabela 5.23: Tempos médios para os melhores resultados

Resolução do PFC	Reordenamento	Método	Tempo Solução PFC[s]	Redução Percentual
Coordenadas Polares	-	Direto	0,0935	-
	-	ILU Direto	0,0414	55,72%
	AMD	BiCGStab	**0,0330**	64,71%
Coordenadas Retangulares	-	ILU Direto	0,0796	14,87%
	AMD	GMRES	0,0787	15,83%
Injeção de Corrente	-	ILU Direto	0,0918	1,82%
Reordenado sem redução	AMD	GMRES	0,0908	2,9%

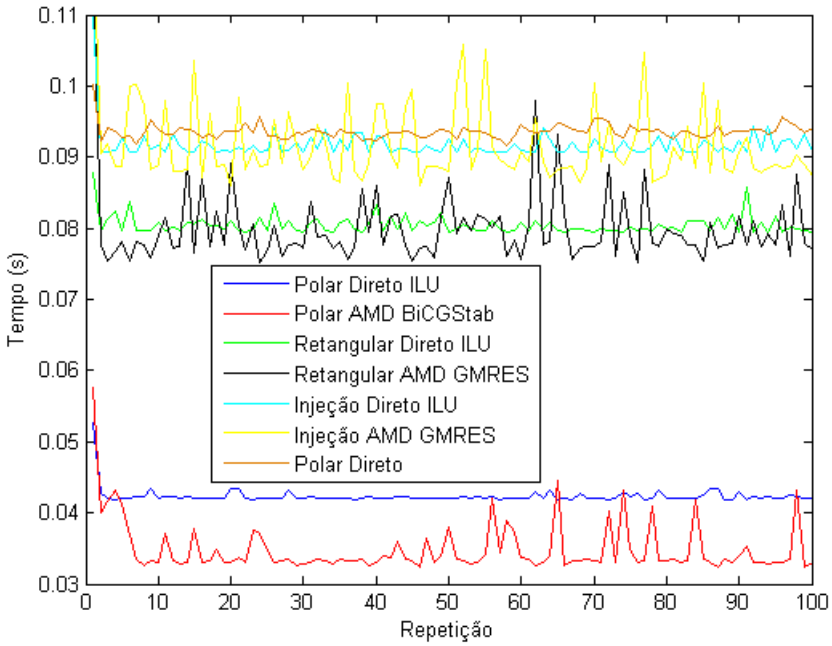


Figura 5.28: Melhores resultados obtidos para a resolução do PFC

Observa-se que todos os métodos obtiveram desempenho superior ao obtido pelo método original do MATPOWER. Os métodos que utilizam coordenadas polares tiveram os melhores desempenhos, tendo uma redução de até 64,71% do tempo obtido pelo MATPOWER, seguidos pelos métodos por coordenadas retangulares com uma redução de até 15,83% e por fim os métodos por injeção de corrente com uma redução de até 2,9%. Os métodos iterativos tiveram desempenhos superiores aos dos métodos diretos para

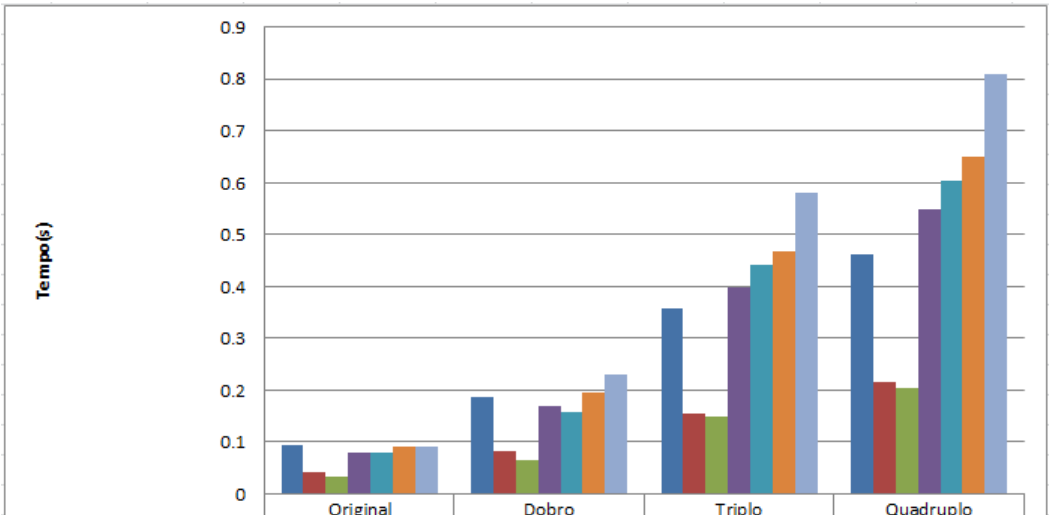
cada forma de resolução do PFC. O melhor desempenho para todos os métodos foi obtido utilizando coordenadas polares, reordenamento AMD e método iterativo BiCGStab, com pré-condicionador fixo baseado na matriz $\mathbf{J_P}$ gerado pela fatoração ILU com os parâmetros droptol=0, thresh=0, milu='off' e type='ilutp'.

Os métodos diretos apresentaram tempos mais estáveis do que os métodos iterativos. O método iterativo com o melhor desempenho, coordenadas polares e AMD BiCGStab, apresentou variações no tempo de CPU de até aproximadamente 30% de seu valor médio. Já o método por coordenadas polares direto ILU apresentou o segundo melhor desempenho e tempos estáveis durante as simulações realizadas, como pode ser observado na Figura 5.28. Conclui-se que ela pode ser interessante em aplicações que demandem maior estabilidade.

5.8.2 Testes com Outros Sistemas

Foram realizados testes utilizando os métodos com os melhores desempenhos para sistemas com maiores quantidades de barras. Os resultados foram testados para sistemas com o dobro, o triplo e o quádruplo de barras do caso com 3375 barras, apresentados na Subseção 4.3.3.

Observa-se na Tabela 5.24 e na Figura 5.29 que o método por coordenadas polares utilizando reordenamento AMD e BiCGStab apresentou o melhor desempenho para todos os sistemas, seguido pelo método por coordenadas polares ILU direto.



	Original	Dobro	Triplo	Quadruplo
Polar Direto	0.0935	0.1869	0.3569	0.4611
Polar ILU Direto	0.0414	0.0831	0.1557	0.2168
Polar AMD BiCGStab	0.033	0.0645	0.1483	0.2028
Retangular ILU Direto	0.0796	0.1687	0.3986	0.5498
Retangular AMD GMRES	0.0787	0.1579	0.4417	0.604
Injeção ILU Direto	0.0918	0.1963	0.4676	0.6497
Injeção AMD GMRES	0.0908	0.2314	0.5803	0.8096

Figura 5.29: Resultados obtidos para a resolução do PFC com diferentes quantidades de barras

Em relação aos métodos que utilizam coordenadas retangulares e injeção de corrente, o método iterativo AMD GMRES apresentou desempenho superior ao do método ILU direto no sistema original. Porém,

Tabela 5.24: Tempos médios para os melhores resultados para diferentes quantidades de barras

Quantidade de Barras	Resolução do PFC	Reordenamento	Método	Tempo Solução PFC[s]
Original, 3375 barras	Coordenadas Polares	-	Direto	0,0935
		-	ILU Direto	0,0414
		AMD	BiCGStab	**0,0330**
	Coordenadas Retangulares	-	ILU Direto	0,0796
		AMD	GMRES	0,0787
	Injeção de Corrente	-	ILU Direto	0,0918
	Reordenado sem Redução	AMD	GMRES	0,0908
Dobro, 6750 barras	Coordenadas Polares	-	Direto	0,1869
		-	ILU Direto	0,0831
		AMD	BiCGStab	**0,0645**
	Coordenadas Retangulares	-	ILU Direto	0,1687
		AMD	GMRES	0,1579
	Injeção de Corrente	-	ILU Direto	0,1963
	Reordenado sem Redução	AMD	GMRES	0,2314
Triplo, 10125 barras	Coordenadas Polares	-	Direto	0,3569
		-	ILU Direto	0,1557
		AMD	BiCGStab	**0,1483**
	Coordenadas Retangulares	-	ILU Direto	0,3986
		AMD	GMRES	0,4417
	Injeção de Corrente	-	ILU Direto	0,4676
	Reordenado sem Redução	AMD	GMRES	0,5803
Quadruplo, 13500 barras	Coordenadas Polares	-	Direto	0,4611
		-	ILU Direto	0,2168
		AMD	BiCGStab	**0,2028**
	Coordenadas Retangulares	-	ILU Direto	0,5498
		AMD	GMRES	0,6040
	Injeção de Corrente	-	ILU Direto	0,6497
	Reordenado sem Redução	AMD	GMRES	0,8096

nos sistemas maiores, o método ILU direto apresentou desempenhos superiores ao do método iterativo.

O método por injeção de corrente sem redução utilizando AMD e GMRES demanda modificações na matriz jacobiana para evitar erros de pivoteamento na aplicação do ILUTP. Com o aumento da ordem do sistema, essas operações passam a demandar tempos elevados, de forma que o método ILU direto por injeção de corrente passou a apresentar desempenhos superiores ao do método iterativo.

O método por coordenadas polares direto pela inversa do jacobiano, utilizado como padrão no MATPOWER, apresentou desempenho inferior ao dos outros métodos para o sistema original. Para sistemas de ordem superior, esse método teve desempenho superior ao dos métodos por injeção de corrente sem redução. A partir do sistema com o triplo de barras, o método por coordenadas polares direto também

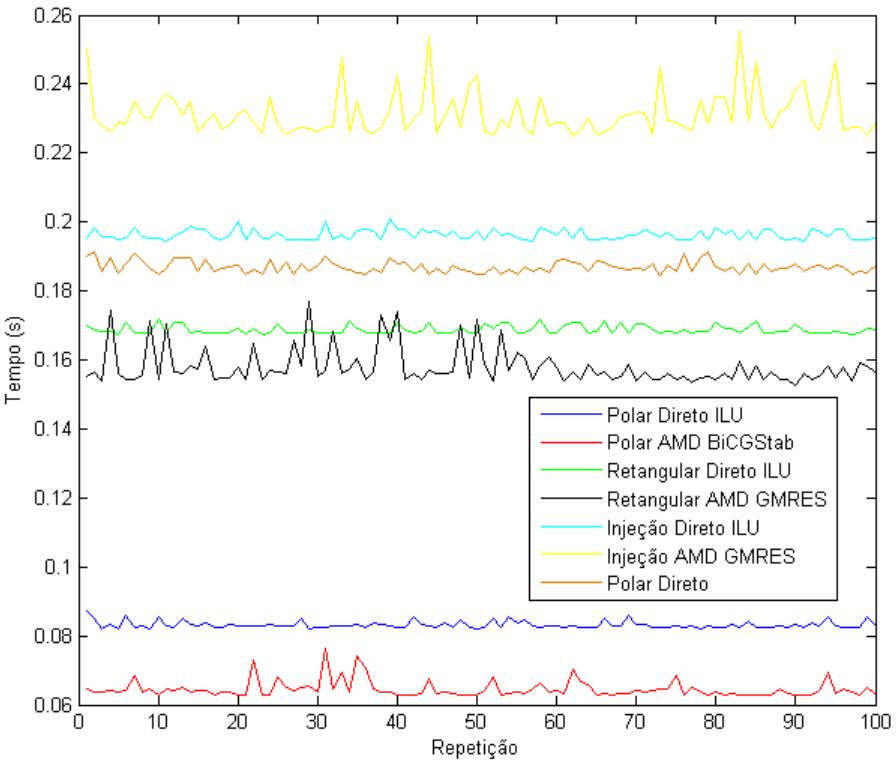


Figura 5.30: Resultados obtidos para a resolução do PFC com o dobro de barras

apresentou desempenho superior ao do método por coordenadas retangulares.

Os melhores resultados do método por coordenadas polares podem ser atribuídos ao fato de as matrizes jacobianas dos métodos por coordenadas retangular e por injeção de corrente terem ordem superior à da matriz jacobiana do método por coordenadas polares. Para o sistema com o quádruplo de barras (13500 barras), o jacobiano em coordenadas polares tem ordem de 25426x25426, em retangulares 26990x26990 e o por injeção de corrente sem redução 28554x28554. Conclui-se que o jacobiano por retangular tem 81978624 mais elementos do que o polar e o por injeção tem 168849440 mais elementos do que o polar.

Pela Figura 5.33, percebe-se que os tempos computacionais demandados para resolução do PFC apresentam um crescimento exponencial com o aumento da quantidade de barras do sistema. Analisando os métodos iterativos com melhores resultados para cada iteração e o método direto utilizado no MATPOWER, perbece-se que o método por coordenadas polares, BiCGStab e reordenamento AMD apresenta uma tendência de crescimento do tempo consideravelmente inferior ao dos outros métodos, de forma que quanto maior o sistema, melhor será seu desempenho em comparação com os outros estudados.

5.8.3 Análise de sensibilidade droptol ILU

Na Subseção 2.4.3.1, foi apresentada a função ILU do MATLAB. O parâmetro *droptol* afeta diretamente os resultados dos tempos de solução do PFC pelo MNR por coordenadas polares, coordenadas retangulares e injeção de corrente. A sensibilidade para o parâmetro *droptol* será analisada para os méto-

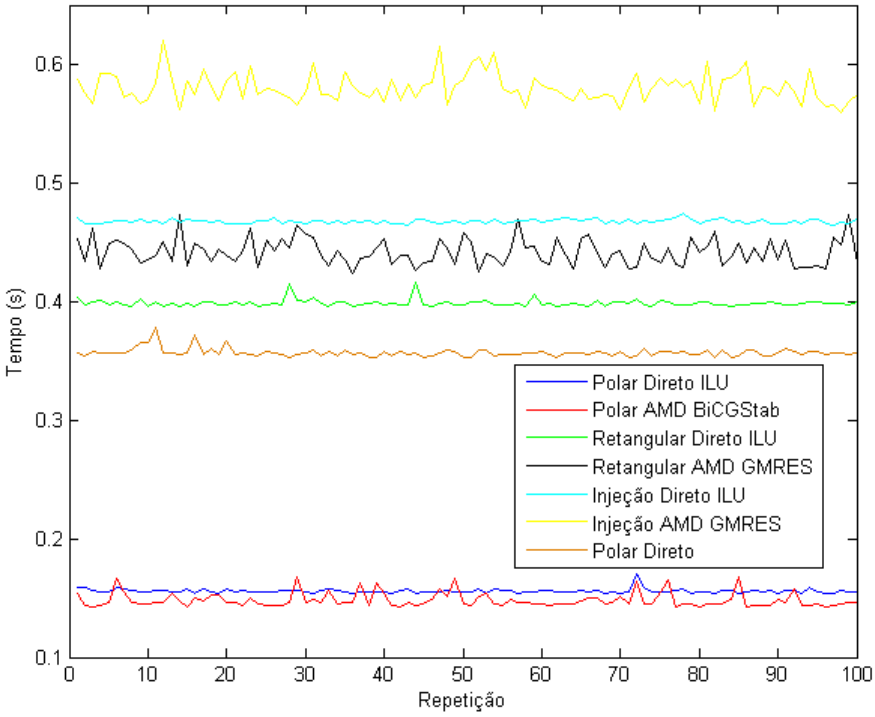


Figura 5.31: Resultados obtidos para a resolução do PFC com o triplo de barras

dos iterativos com os melhores resultados para coordenadas polares, retangulares e por injeção de corrente sem redução. A configuração do método ILU foi thresh=0, milu='off', type='ilutp' e *droptol* variando de 0 a 1×10^{-4}.

Observa-se nas Figuras 5.34, 5.35 e 5.36 que o tempo mínimo é obtido para droptol=0. Para valores de *droptol* maiores do que 0, o tempo computacional demandado aumenta consideravelmente para coordenadas polares, coordenadas retangulares e injeção de corrente. Um dos possíveis motivos é que, ao utilizar o parâmetro thresh=0 e type='ilutp', o pivoteamento pela diagonal combinado com a fatoração iILUTP permitem encontrar matrizes $\tilde{L}$ e $\tilde{U}$ praticamente equivalentes às matrizes L e U geradas pela fatoração LU completa, porém com um tempo de CPU inferior como foi mostrado na Seção 5.3. Dessa forma, a convergência dos métodos se torna mais rápida. Quando aumentamos o valor de *droptol*, as matrizes $\tilde{L}$ e $\tilde{U}$ se diferenciam cada vez mais das matrizes L e U, exigindo mais iterações dos métodos iterativos e maior tempo de CPU.

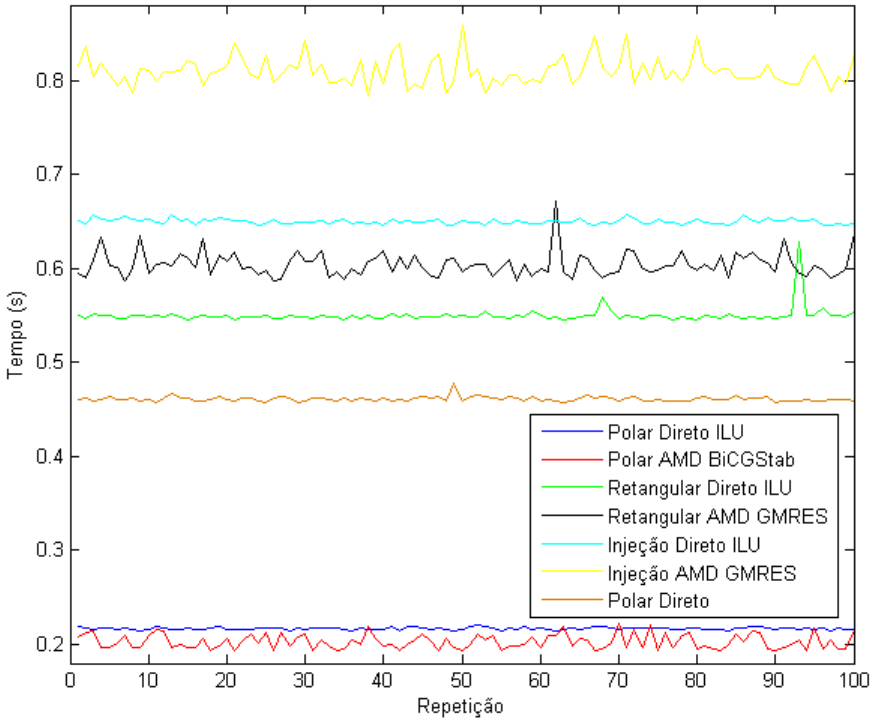


Figura 5.32: Resultados obtidos para a resolução do PFC com o quadruplo de barras

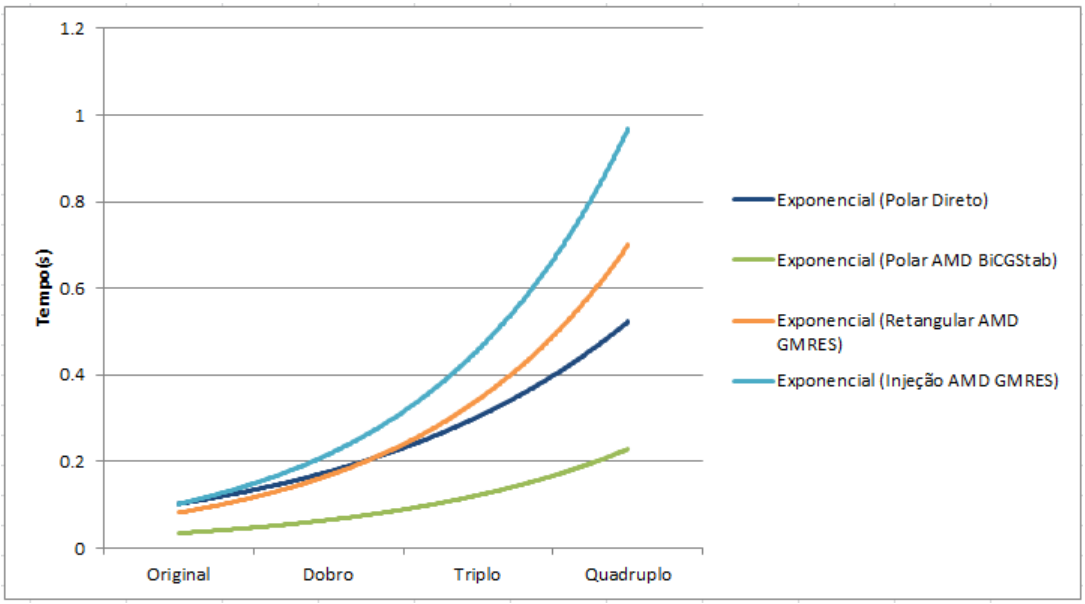


Figura 5.33: Gráficos exponenciais para o tempo médio de diferentes formulações do PFC utilizando método iterativos e o método do MATPOWER

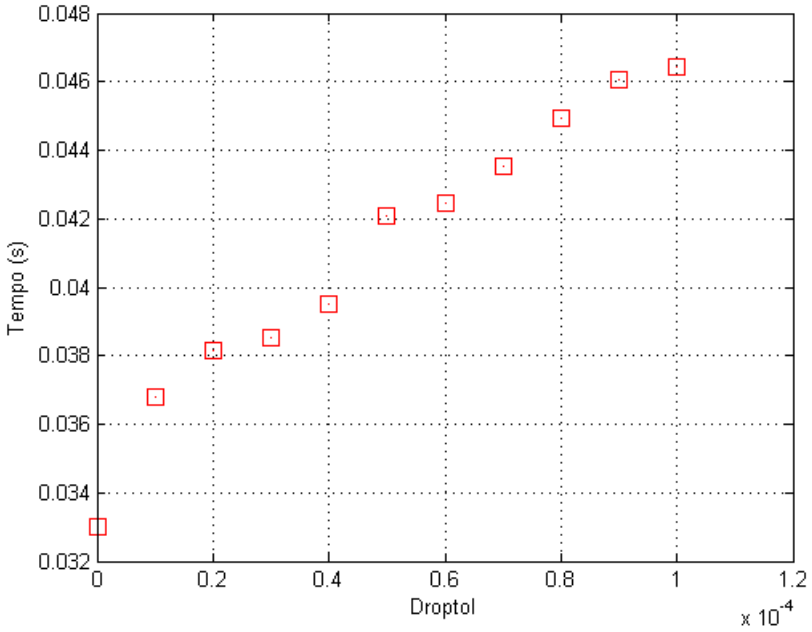


Figura 5.34: Sensibilidade dos tempos médios de CPU ao *droptol* para o método com coordenadas polares por AMD BiCGStab

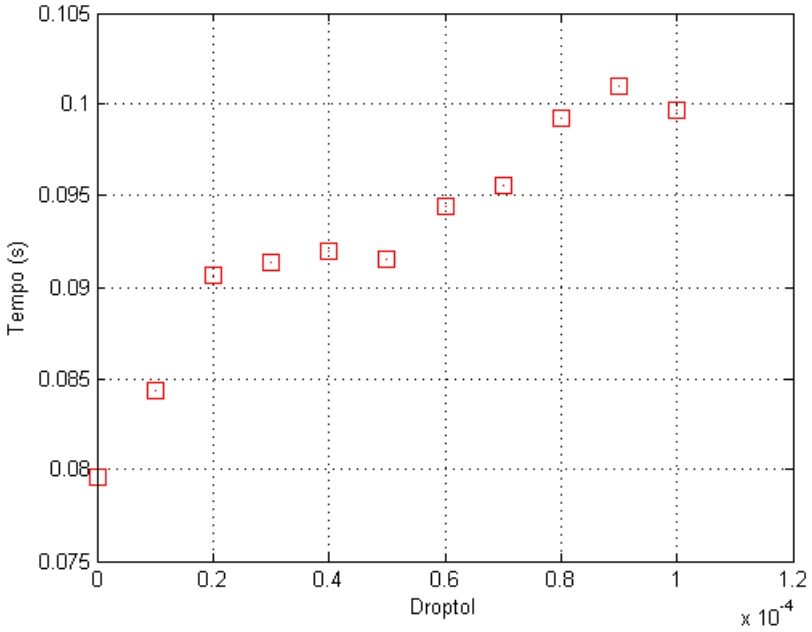


Figura 5.35: Sensibilidade dos tempos médios de CPU ao *droptol* para o método com coordenadas retangulares por AMD GMRES

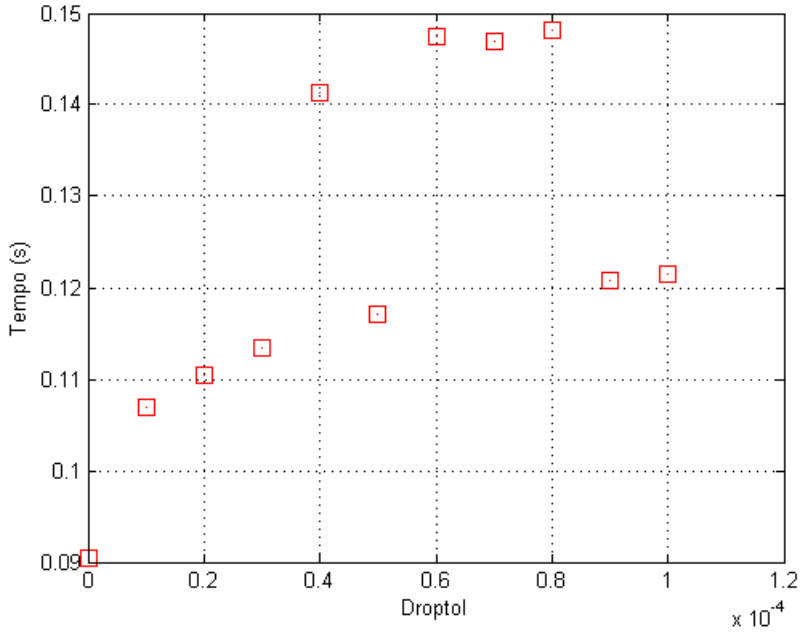


Figura 5.36: Sensibilidade dos tempos médios de CPU ao *droptol* para o método com injeção de corrente reordenado sem redução por AMD GMRES

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Foram realizados testes com o método iterativo GMRES para comparar o desempenho computacional da função *built-in* do MATLAB com uma função a partir do Algoritmo 2.1. Verificou-se que a função *built-in* possui um desempenho superior e maior estabilidade. Portanto, foram utilizadas funções *built-in* para os métodos iterativos nas simulações seguintes por já estarem otimizadas.

Realizou-se testes com as fatorações LU e ILU do MATLAB para definir qual a função e quais valores dos parâmetros geravam as matrizes L e U no menor tempo de CPU a partir do jacobiano do caso de 3375 barras utilizando coordenadas polares. O melhor tempo foi obtido utilizando o método ILU com os parâmetros type='ílutp', milu='off', thresh=0 e droptol=0.

Em seguida, observou-se a ocorrência do preenchimento das matrizes L, U e M na fatoração LU de matrizes esparsas de grande porte. Verificou-se que a utilização do reordenamento e da fatoração ILU reduz consideravelmente o preenchimento e o tempo de CPU demandado.

Foram realizados estudos e simulações utilizando-se métodos diretos e iterativos para a resolução de problemas de fluxo de carga. Os principais conceitos para o entendimento dos problemas de fluxo de carga e dos métodos de resolução foram abordados e diferentes formulações matemáticas foram estudadas e analisadas. Um novo método direto para resolução do subproblema linear do MNR na solução do PFC utilizando expansão linear pela série de Taylor foi apresentado. Os resultados simulados foram analisados e comparados para definir os métodos com melhores desempenhos. Após serem definidos, esses métodos foram testados em diferentes sistemas teste para verificar os desempenhos. Análises de sensibilidade de parâmetros das funções utilizadas no MATLAB foram realizadas.

A formulação do PFC por coordenadas polares foi testada para métodos diretos e iterativos. Dentre os métodos diretos testados, o melhor desempenho foi obtido pela fatoração ILUTP e reordenamento AMD da matriz jacobiana, com um tempo médio de resolução do PFC para o caso de 3375 barras de 0,0414. O método por expansão linear apresentou o segundo melhor resultado dos métodos diretos, com tempo médio de 0,0537. Em seguida foram testados os métodos iterativos GMRES e BiCGStab com reordenamento AMD ou RCM utilizando pré-condicionadores variáveis e fixos. Os pré-condicionadores fixos apresentaram me-

lhores desempenhos em relação aos variáveis. Diferentes tipos de pré-condicionadores para coordenadas polares foram testados e a sensibilidade ao parâmetro droptol da fatoração ILU em relação aos resultados foi analisada. Verificou-se que para $droptol = 6,5 \times 10^{-5}$, o pré-condicionadores T_3 obteve desempenho superior ao jacobiano completo, com tempo médio de 0,0374 em comparação com 0,0393 para o jacobiano completo. No entanto, o melhor resultado geral foi obtido para droptol=0, método BiCGStab, reordenamento AMD e pré-condicionador fixo baseado na matriz jacobiana $\mathbf{J_P}$ completa, com um tempo médio de 0,033.

Foi realizado um estudo do número de iterações dos métodos iterativos para cada iteração do MNR para coordenadas polares. Observou-se que, utilizando o pré-condicionador fixo baseado na $\mathbf{J_P}$ completa, na primeira iteração do MNR, o método iterativo BiCGStab converge em 0,5 iterações. Como o pré-condicionador é mantido fixo, a quantidade de iterações necessária para a convergência cresce para 1,5 e 2 nas iterações seguintes do MNR. Os melhores métodos direto e iterativo por coordenadas polares, com tempos médios de 0,0414 e 0,033, respectivamente, obtiveram desempenho consideravelmente superior em comparação com o método pela inversa do jacobiano por coordenadas polares, comumente utilizado pelo MATPOWER, que obteve um tempo médio de 0,0935.

Testaram-se métodos diretos e iterativos para o PFC pela formulação por coordenadas retangulares. Dentre os métodos diretos testados, o melhor desempenho, assim como para coordenadas polares, foi obtido pela fatoração ILUTP e reordenamento AMD da matriz jacobiana $\mathbf{J_R}$ com tempo médio de 0,0796. O método por expansão linear apresentou o segundo melhor resultado dos métodos diretos, com tempo médio de 0,1075. Foram testados os métodos iterativos GMRES e BiCGStab com reordenamento AMD ou RCM utilizando pré-condicionadores variáveis e fixos. O melhor resultado geral por coordenadas retangulares foi obtido para o método GMRES, reordenamento AMD e pré-condicionador fixo baseado na matriz jacobiana $\mathbf{J_R}$ completa com tempo médio de 0,0787. Outros tipo de pré-condicionadores foram testados, porém apresentaram tempos muito elevados. Os melhores métodos direto e iterativo por coordenadas polares, com tempos médios de 0,0796 e 0,0787, respectivamente, obtiveram desempenho superior em comparação com o método pela inversa do jacobiano por coordenadas polares, comumente utilizado pelo MATPOWER, que obteve um tempo médio de 0,0935.

Em seguida, testaram-se métodos diretos e iterativos pela modelagem do PFC por injeção de correntes. Foram utilizadas duas formas de modificação da matriz $\mathbf{J_I}$ para remover elementos nulos da diagonal e evitar erros de pivoteamento: reordenamento de linhas e colunas com redução de Kron da matriz e reordenamento de linhas e colunas sem redução. O método direto com melhor desempenho foi obtido para o reordenamento sem redução da matriz $\mathbf{J_I}$, fatoração ILUTP e reordenamento AMD da matriz jacobiana $\mathbf{J_{I_{reord}}}$, com tempo médio de 0,0918. Para os métodos iterativos, o melhor desempenho foi obtido para o reordenamento sem redução da matriz $\mathbf{J_I}$, método GMRES, reordenamento AMD e pré-condicionador fixo baseado na matriz jacobiana $\mathbf{J_{I_{reord}}}$ completa, com tempo médio de 0,0908. Outros tipo de pré-condicionadores foram testados, porém não apresentaram resultados satisfatórios. Os melhores métodos direto e iterativo por injeção de corrente, com tempos médios de 0,0918 e 0,0908, respectivamente, obtiveram desempenhos um pouco superiores em comparação com o método pela inversa do jacobiano por coordenadas polares, comumente utilizado pelo MATPOWER, que obteve um tempo médio de 0,0935.

Foram analisados o melhor método direto e o melhor método iterativo de cada formulação do PFC,

além do método direto utilizado pelo MATPOWER, em sistemas de ordem superior ao caso de 3375 barras. O método iterativo BiCGStab por coordenadas polares, reordenamento AMD e pré-condicionador fixo baseado na matriz completa teve o melhor desempenho em todas as simulações. O segundo melhor desempenho em todas as simulações foi obtido para o método direto por coordenadas polares, fatoração ILUTP e reordenamento AMD. Os melhores métodos por coordenadas retangulares tiveram desempenho superior aos métodos por injeção de corrente. Para o sistema original de 3375 barras, os melhores métodos por coordenadas retangulares e por injeção de corrente foram superiores ao método utilizado no MATPOWER. Para sistemas de ordem superior, o método utilizado pelo MATPOWER apresentou desempenho superior aos métodos por injeção de corrente e desempenho superior aos métodos por coordenadas retangulares a partir do sistema com o triplo de barras do caso 3375. Isso se deve ao fato de que os jacobianos dos métodos retangular e injeção de corrente terem ordem superior ao jacobiano do método polar, de forma que quanto maior o sistema, maior a diferença de tamanho, o que piora o desempenho computacional.

A sensibilidade ao parâmetro *droptol*, que define a tolerância do ILUTP, foi analisada e verificou-se que ele altera de maneira significativa os pré-condicionadores e, consequentemente, o desempenho dos métodos iterativos. Para todos os métodos iterativos e formulações utilizadas, com os parâmetros da função ILUTP definidos como type='ílutp', milu='off' e thresh=0, a redução do *droptol* melhorou o desempenho computacional. Os melhores resultados dos métodos iterativos foram obtidos utilizando droptol=0.

Em síntese, método iterativo BiCGStab apresentou os melhores resultados para a forma polar. Já o método iterativo GMRES teve desempenho superior para a forma retangular e para a modelagem por injeção de corrente. O método direto utilizando expansão linear da série de Taylor do jacobiano não apresentou resultados satisfatórios, porém o método deve ser melhor estudado para encontrar formas de otimizar a obtenção das variações do jacobiano sem exigir um grande esforço computacional. No geral, o BiCGStab apresentou o melhor desempenho, em conjunto com a forma polar e o reodernamento AMD. Os pré-condicionadores fixos, calculados apenas na primeira iteração, tiveram desempenho superior ao dos pré-condicionadores variáveis, calculados a cada iteração. O reordenamento AMD apresentou desempenho consideravelmente superior ao reordenamento RCM em todos os testes realizados.

O trabalho apresentou contribuições ao tema estudado. Foram desenvolvidas técnicas utilizando a redução de Kron ou apenas o reordenamento para a matriz jacobiana pela modelagem por injeção de corrente que eliminam elementos nulos da diagonal principal e permitem a aplicação da fatoração ILU utilizando pivoteamento pela diagonal. Também foi proposto um novo método de resolução do PFC pelo MNR utilizando expansão linear pela série de Taylor. O método apresentou resultados consistentes e possui potencial para obter resultados melhores caso seja otimizado. O método proposto utilizando coordenadas polares, BiCGStab e reordenamento AMD apresentou resultados consideravelmente melhores do que o resultado obtido pelo MATPOWER para sistemas de ordem superior.

6.2 Trabalhos Futuros

Diversos estudos podem ser realizados nessa área em trabalhos futuros. Algumas sugestões são:

- Analisar diferentes opções de pré-condicionadores para as formulações por coordenadas polares,

coordenadas retangulares e para a modelagem por injeção de corrente

- Verificar a sensibilidade dos desempenhos computacionais dos métodos iterativos em relação a outros parâmetros utilizados nas funções do MATLAB
- Analisar o desempenhos dos métodos em sistemas com perturbações
- Estudar maneiras de otimizar os métodos utilizando coordenadas retangulares, modelagem por injeção de corrente e expansão linear da série de Taylor
- Realizar estudos do fluxo de potência continuado para o método por injeção de corrente

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MONTICELLI, A. J. *Fluxo de Carga em Redes de Energia Elétrica*. [S.l.]: Editora Edgard Blücher Ltda., 1983.

[2] FERREIRA, C. A. *Novas Aplicações da Formulação de Injeção de Corrente em Sistemas Elétricos de Potência*. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2003.

[3] FERNANDES, A. A. *Uma Estrutura de Solucionador Iterativo Linear com Aplicação à Solução de Equações do Problema de Fluxo de Carga*. Tese (Doutorado) — Universidade de Brasília, 2014.

[4] FERNANDES, A. A.; FREITAS, F. D.; ISHIHARA, J. Y. *Preconditioning Based on Decoupling for Non-symmetric Matrices with Application to Power Systems*. Universidade de Brasilia, 2013.

[5] LUGON, B. A. *Algoritmos de reordenamento de matrizes esparsas aplicados a precondicionadores ILU(p)*. XLV SBPO - Simpósio Brasileiro de Pesquisa Operacional, Universidade Federal do Espírito Santo, 2013.

[6] POMA, C. E. P. *Um Solucionador Iterativo Para Sistemas-Lineares: Aplicação no Problema do Fluxo de Carga*. Tese (Doutorado) — Pontifícia Universidade Católica, 2010.

[7] PAZ, A. R. A. *Aplicação do Método GMRES na Solução de Problemas de Estabilidade em Sistemas de Energia Elétrica*. Tese (Doutorado) — Pontifícia Universidade Católica, 2012.

[8] ZIMMERMAN, R. *"Powerweb", Power Systems Engineering Research Center*. Acessado em maio de 2014. Disponível em: <http://www.pserc.cornell.edu:8082/powerweb/>.

[9] IDEMA, R. et al. Scalable newton-krylov solver for very large power flow problems. *IEEE Transactions on Power Systems*, vol 27 nº 1 pp 390–396, 2012.

[10] TOPSAKAL, E. et al. Evaluation of the bicgstab(i) algorithm for the finite-elemental boundary-integral method. *IEEE Antennas and Propagation Magazine*, 2001.

[11] TU, F.; FLUECK, A. J. *A Message-Passing Distributed-Memory Newton-GMRES Parallel Power Flow Algorithm*. Department of Electrical and Computer Engineering, Illinois Institute of Technology, Power Engineering Society Summer Meeting, 2002, IEEE vol. 3, 25–25 July pp 1477–1482 Chicago, IL, USA.

[12] GOLUB, G. H.; LOAN, C. F. V. *Matrix Computations*. [S.l.]: The Johns Hopkins University Press, 1996.

[13] CHAN, T. F.; VORST, H. A. V. D. *Approximate and Incomplete Factorization*. University of California, 2013.

[14] SAAD, Y. *Iterative Methods for Sparse Linear Systems*. [S.l.]: Society for Industrial and Applied Mathematics, 2003.

[15] BARRETT, R.; VORST, H. V. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. [S.l.]: Siam, 1993.

[16] SPHAIER, L. A.; ALVES, L. S. B. *Métodos Computacionais com o Mathematica*. [S.l.]: E-Papers Serviços Editoriais LTDA, 2007.

[17] STEVENSON, W. D. *Elementos de Análise de Sistemas de Potência*. [S.l.]: McGraw-Hill, Ltda, 1986.

[18] FRANCO, P. C. *Análise de Sistemas de Potência*. Universidade de Brasilia, 2014.

[19] LEITE, L. C. G.; COSTA, V. M. Fluxo de potência continuado via equações de injeção de corrente. *Revista Controle & Automação*, vol. 14, nº 4, out/nov/dez/2003, pp 430–432.

[20] ZIMMERMAN, R.; MURILLO-SANCHEZ, C. *Matpower 4.1 User´s Manual*. Power Systems Engineering Research Center, 2011.

[21] EL, H. R.; BOUCHEKARA, H. *The Admittance Model and Network Calculations*. Electrical Power System Analysis, Umm Al-Qura University, 2012.

7. ANEXOS

7.1 Funções do Programa de Resolução do PFC

I - runpf_geral(): script geral para seleção do método

Método Polar: script para resolução do PFC pelo MNR utilizando coordenadas polares

- pol_newton_direto(): métodos diretos
- pol_newton_expansao(): método por expansão linear da série de Taylor
  - pol_expansão_linear(): resolução por expansão linear da série de Taylor
- pol_newton_iterativo(): métodos iterativos
  - pol_iterativo_reordenamento(): reordenamento e escolha do tipo de método iterativo

Método Retangular: script para resolução do PFC pelo MNR utilizando coordenadas retangulares

- ret_newton_direto(): métodos diretos
- ret_newton_expansao(): expansão linear da série de Taylor
  - ret_expansão_linear(): resolução por expansão linear da série de Taylor
- ret_newton_iterativo(): métodos iterativos
  - ret_newton_iterativo_reordenamento_fast(): reordenamento e método iterativo com melhor desempenho
  - ret_newton_iterativo_reordenamento_pre(): reordenamento e escolha do tipo de método iterativo

Método Injeção de Corrente: script para resolução do PFC pelo MNR utilizando injeção de corrente Reordenamento com redução da matriz $\mathbf{J_I}$

- inj_newton_direto(): métodos diretos
- inj_newton_expansao(): expansão linear da série de Taylor
  - inj_expansão_linear(): resolução por expansão linear da série de Taylor
- inj_newton_iterativo(): métodos iterativos
  - inj_reducao_iterativo_fast(): reordenamento e redução da matriz $\mathbf{J_I}$ com melhor desempenho
    - inj_iterativo_reordenamento_fast(): reordenamento e método iterativo com melhor desempenho
  - inj_reducao_iterativo_pre(): reordenamento e redução da matriz $\mathbf{J_I}$
    - inj_iterativo_reordenamento_pre(): reordenamento e método iterativo

Reordenamento sem redução da matriz $\mathbf{J_I}$

- inj2_newton_iterativo(): métodos iterativos
  - inj2_reordcol_iterativo(): reordenamento sem redução da matriz $\mathbf{J_I}$
    - inj2_iterativo_reordenamento_fast(): reordenamento e método iterativo com melhor desempenho

7.2 Códigos

runpf_geral.m

```
function [results, success, et] = ...
                runpf_geral(casedata, mpopt, fname, solvedcase)
%% Dados Iniciais
rep=1; %quantidade de repetições
testes=0; %contagem de testes realizados
t_teste=tic;
legenda=[];
plotar_tempos=0; %1-plota graficos dos tempos para cada alg    0-não plota ...
    gráficos
plotar_droptol=0;

%% Programa de resolução do problema de Fluxo de Cargas por Diferentes Métodos ...
    Direto e Iterativos
for alg=[4] %Define o método de resolução
%% Métodos alg
%alg=1; %coordernadas polares direto
%alg=2 ;%coordenadas retangulares direto
%alg=3 ;%injeção de corrente direto
%alg=4 ;%coordenadas polares iterativo
%alg=5 ;%coordenadas retangulares iterativo
%alg=6 ;%injeção de corrente reduzida iterativo
%alg=7 ;%injeção de corrente reordenado sem redução

%% Tempos
tempo_pf=[];
tempo_total=[];
tilu=[];

%% Não Alterar - Dados Pré-Definidos
pr=0;re=0;me=0;dropt=0;md=0;

%% Pre definir método direto, método iterativo, ...
    reordenamento,pre-condicionadores e droptol
%md- Método Direto-  1-Inversa    2-LU    3-ILU    4-Expansão Linear
%pr- Pre Condicionador- 1-Pre1    2-Pre2    3-Pre3   4-Completa
%re- Reordenamento 1-AMD    2-RCM
%me- Método Iterativo 1-BiCGSTAB    2-GMRES
%dropt- Droptolerance
```

```
36      %% Métodos Diretos
37      if alg==1|alg==2|alg==3
38          % Polar Direto
39          if alg==1
40              md=[1,2,3,4]; %1-Inversa    2-LU    3-ILU    4-Expansão Linear
41
42          % Retangular Direto
43          elseif alg==2
44              md=[1,2,3,4]; %1-Inversa    2-LU    3-ILU    4-Expansão Linear
45
46          %Injeção Direto
47          elseif alg==3
48              md=[1,2,3,4,5]; %1-Inversa    2-LU    3-ILU Reordenado sem Redução ...
                      4-Expansão Linear Reordenado sem Redução    5-ILU Reordenado ...
                    com Redução
49
50          end
51
52      %% Métodos Iterativos
53      %Polar Iterativo
54      elseif alg==4
55          pr=[3,4]; %Pre-Condicionadores 1-T1    2-T2    3-T3     4-Completa
56          re=[1];%Reordenamento   1-AMD    2-RCM
57          me=[1];%Método Iterativo    BiCGSTAB-1    GMRES-2
58          dropt=[0];
59          %dropt=[0:1e-5:6.5e-5];%DropTolerance
60
61      %Retangular Iterativo
62      elseif alg==5
63          pr=[4]; %Pre-Condicionadores 1-Pre1    2-Pre2    3-Pre3   4-Completa
64          re=[1];%AMD-1    RCM-2
65          me=[2];%BiCGSTAB-1    GMRES-2
66          dropt=[0];%DropTolerance
67
68      %Injeção Reduzido
69      elseif alg==6
70          pr=[4:1]; %Pre-Condicionadores 1-Pre1    2-Pre2    3-Pre3   4-Completa
71          re=[1,2];%AMD-1    RCM-2
72          me=[1,2];%BiCGSTAB-1    GMRES-2
73          dropt=0;%DropTolerance
74
75      %Injeção sem Redução
76      elseif alg==7
77          pr=[4]; %Pre-Condicionadores 1-Pre1    2-Pre2    3-Pre3   4-Completa
78          re=[1];%AMD-1    RCM-2
79          me=[2];%BiCGSTAB-1    GMRES-2
80          dropt=0;%DropTolerance
81      end
82
83  %% Loop para o tipo de reordenamento, método direto, método iterativo, ...
        pré-condicionador e droptol
84
```

```
for droptol=dropt;
for reordenamento=re;
for metodo=me;
for precond=pr;
for metodo_direto=md;
testes=testes+1;
for k=1:rep;
%fprintf('\n %%%%%%%%%% Repetição %d %%%%%%%%%% \n', k);
t1=tic; %tempo total

%% define named indices into bus, gen, branch matrices
[PQ, PV, REF, NONE, BUS_I, BUS_TYPE, PD, QD, GS, BS, BUS_AREA, VM, ...
    VA, BASE_KV, ZONE, VMAX, VMIN, LAM_P, LAM_Q, MU_VMAX, MU_VMIN] = idx_bus;
[F_BUS, T_BUS, BR_R, BR_X, BR_B, RATE_A, RATE_B, RATE_C, ...
    TAP, SHIFT, BR_STATUS, PF, QF, PT, QT, MU_SF, MU_ST, ...
    ANGMIN, ANGMAX, MU_ANGMIN, MU_ANGMAX] = idx_brch;
[GEN_BUS, PG, QG, QMAX, QMIN, VG, MBASE, GEN_STATUS, PMAX, PMIN, ...
    MU_PMAX, MU_PMIN, MU_QMAX, MU_QMIN, PC1, PC2, QC1MIN, QC1MAX, ...
    QC2MIN, QC2MAX, RAMP_AGC, RAMP_10, RAMP_30, RAMP_Q, APF] = idx_gen;

%Contagem tempo método polar
%t2=tic;

%% default arguments
if nargin < 4
    solvedcase = '';                % %% don't save solved case
    if nargin < 3
        fname = '';                 % %% don't print results to a file
        if nargin < 2
            mpopt = mpoption;       % %% use default options
            if nargin < 1
                casedata = 'case30'; %% default data file is 'case9.m'
            end
        end
    end
end

%% options
verbose = mpopt(31);
qlim = mpopt(6);                    % %% enforce Q limits on gens?
dc = mpopt(10);                     % %% use DC formulation?

mpc=loadcase(casedata); %(tt);   %colocar o nome do arquivo do Matpower

%% add zero columns to branch for flows if needed
if size(mpc.branch,2) < QT
  mpc.branch = [ mpc.branch sparse(size(mpc.branch, 1), QT-size(mpc.branch,2)) ];
end

%% convert to internal indexing
mpc = ext2int(mpc);
[baseMVA, bus, gen, branch] = deal(mpc.baseMVA, mpc.bus, mpc.gen, mpc.branch);
```

```
137
138 %% get bus index lists of each type of bus
139 [ref, pv, pq] = bustypes(bus, gen);
140 ref;
141 pv;
142 pq ;
143
144 %% generator info
145 on = find(gen(:, GEN_STATUS) > 0);      %% which generators are on?
146 gbus = gen(on, GEN_BUS);                %% what buses are they at?
147
148 %%-----  run the power flow  -----
149 t0 = clock;
150 if verbose > 0
151     v = mpver('all');
152 %    fprintf('\nMATPOWER Version %s, %s', v.Version, v.Date);
153 end
154 if dc                                   %% DC formulation
155     if verbose > 0
156 %       fprintf(' -- DC Power Flow\n');
157     end
158     %% initial state
159     Va0 = bus(:, VA) * (pi/180);
160
161     %% build B matrices and phase shift injections
162     [B, Bf, Pbusinj, Pfinj] = makeBdc(baseMVA, bus, branch);
163
164     %% compute complex bus power injections (generation - load)
165     %% adjusted for phase shifters and real shunts
166     Pbus = real(makeSbus(baseMVA, bus, gen)) - Pbusinj - bus(:, GS) / baseMVA;
167
168     %% "run" the power flow
169     Va = dcpf(B, Pbus, Va0, ref, pv, pq);
170     pv;
171     pq;
172
173     %% update data matrices with solution
174     branch(:, [QF, QT]) = sparse(size(branch, 1), 2);
175     branch(:, PF) = (Bf * Va + Pfinj) * baseMVA;
176     branch(:, PT) = -branch(:, PF);
177     bus(:, VM) = ones(size(bus, 1), 1);
178     bus(:, VA) = Va * (180/pi);
179     %% update Pg for slack generator (1st gen at ref bus)
180     %% (note: other gens at ref bus are accounted for in Pbus)
181     %%      Pg = Pinj + Pload + Gs
182     %%      newPg = oldPg + newPinj - oldPinj
183     refgen = sparse(size(ref));
184     for k = 1:length(ref)
185         temp = find(gbus == ref(k));
186         refgen(k) = on(temp(1));
187     end
188     gen(refgen, PG) = gen(refgen, PG) + (B(ref, :) * Va - Pbus(ref)) * baseMVA;
```

```
189
190     success = 1;
191 else
192
193 %%   Definindo saida do fprint
194     if verbose > 0
195         if alg == 1
196             solver = 'Polar Direto ';
197
198         elseif alg == 2
199             solver = 'Retangular Direto ';
200
201         elseif alg == 3
202             solver = 'Injeção de Corrente Direto ';
203
204         elseif alg == 4
205             solver = 'Polar Iterativo ';
206
207         elseif alg == 5
208             solver = 'Retangular Iterativo ';
209
210         elseif alg == 6
211             solver = 'Injeção Reduzido Iterativo ';
212
213         elseif alg == 7
214             solver = 'Injeção Sem Redução Iterativo ';
215
216         else
217             solver = 'unknown ';
218         end
219
220         if reordenamento==1;
221             reord = 'AMD ';
222         elseif reordenamento==2;
223             reord = 'RCM ';
224         else
225             reord = '';
226         end
227
228         if metodo==1;
229             met = 'BiCGSTAB ';
230         elseif metodo==2;
231             met = 'GMRES ';
232         else
233             met = '';
234         end
235
236         if metodo_direto==1;
237             met_d = 'Inversa \ ';
238         elseif metodo_direto==2;
239             met_d = 'LU ';
240         elseif metodo_direto==3;
```

```
241             if alg==3
242                 met_d = 'Reordenado sem Redução ILU ';
243             else
244                 met_d = 'ILU ';
245             end
246         elseif metodo_direto==4;
247             if alg==3
248                 met_d = 'Reordenado sem Redução Expansão Linear ';
249             else
250                 met_d= 'Expansão Linear ';
251             end
252         elseif metodo_direto==5;
253             met_d = 'Reordenado com Redução ILU ';
254         else
255             met_d='';
256         end
257 
258         if precond==1;
259             prec='Precond 1 ';
260         elseif precond==2;
261             prec='Precond 2 ';
262         elseif precond==3;
263             prec='Precond 3 ';
264         elseif precond==4;
265             prec='Precond Completo ';
266         else
267             prec='';
268         end
269     end
270 
271 %% AC formulation
272 
273     % Estado Inicial
274     % V0    = ones(size(bus, 1), 1);            %% flat start
275     V0  = bus(:, VM) .* exp(sqrt(-1) * pi/180 * bus(:, VA));
276     V0(gbus) = gen(on, VG) ./ abs(V0(gbus)).* V0(gbus);
277 
278     if qlim
279         ref0 = ref;                         %% save index and angle of
280         Varef0 = bus(ref0, VA);             %%   original reference bus(es)
281         limited = [];                       %% list of indices of gens @ Q lims
282         fixedQg = sparse(size(gen, 1), 1);   %% Qg of gens at Q limits
283     end
284     repeat = 1;
285     V0;
286     while (repeat)
287         %% Construção da Matriz Admitância Ybus
288         [Ybus, Yf, Yt] = makeYbus(baseMVA, bus, branch);
289 
290         %% Cálculo da Potência Complexa Injetada(Geração - Carga)
291         Sbus = makeSbus(baseMVA, bus, gen);
292         Yb=full(Ybus);
```

```
293         Sbus;
294
295         %% Rodar o Fluxo de Carga
296 t2=tic; %tempo power flow
297         if alg == 1
298             if metodo_direto==4
299                 [V, success, iterations] = pol_newton_expansao(Ybus, Sbus, V0, ...
                        ref, pv, pq, mpopt);
300             else
301                 [V, success, iterations] = pol_newton_direto(Ybus, Sbus, V0, ...
                        ref, pv, pq, mpopt,metodo_direto);
302             end
303
304         elseif alg == 2
305            if metodo_direto==4
306                 [V, success, iterations] = ret_newton_expansao(Ybus, Sbus, V0, ...
                        pv, pq);
307            else
308                 [V, success, iterations] = ret_newton_direto(Ybus, Sbus, V0, ...
                        pv, pq, metodo_direto);
309            end
310
311         elseif alg == 3
312             if metodo_direto==4
313                 [V, success, iterations] = inj_newton_expansao(Ybus, Sbus, V0, ...
                        pv, pq,reordenamento);
314             else
315                 [V, success, iterations] = inj_newton_direto(Ybus, Sbus, V0, ...
                        pv, pq, metodo_direto);
316             end
317
318         elseif alg == 4
319             [V, success, iterations,iterations_iter,t_iter] = ...
                    pol_newton_iterativo(Ybus, Sbus, V0, ref, pv, pq, ...
                    mpopt,reordenamento,metodo,precond,droptol,rep);
320             linha_res_pre=0;
321
322         elseif alg == 5
323             [V, success, iterations] = ret_newton_iterativo(Ybus, Sbus, V0, ...
                    pv, pq,k,reordenamento,metodo,precond,droptol);
324             linha_res_pre=5;
325
326         elseif alg == 6
327             [V, success, iterations,tilu] = inj_newton_iterativo(Ybus, Sbus, ...
                    V0, pv, pq,k,tilu,reordenamento,metodo,precond,droptol);
328             linha_res_pre=10;
329
330         elseif alg == 7
331             [V, success, iterations] = inj2_newton_iterativo(Ybus, Sbus, V0, ...
                    pv, pq,k,tilu,reordenamento,metodo,precond,droptol);
332             linha_res_pre=15;
333
```

```
334         else
335             error('Não foi selecionado nenhum método');
336         end
337
338 tempo_pf(k)=toc(t2); %tempo power flow
339
340
341 if alg==4
342     t_iteracao1(k)=t_iter;
343 end
344
345         %% Atualização das matrizes de dados com as soluções
346         [bus, gen, branch] = pfsoln(baseMVA, bus, gen, branch, Ybus, Yf, Yt, ...
                V, ref, pv, pq);
347
348         if qlim             %% enforce generator Q limits
349             %% find gens with violated Q constraints
350             mx = find( gen(:, GEN_STATUS) > 0 & gen(:, QG) > gen(:, QMAX) );
351             mn = find( gen(:, GEN_STATUS) > 0 & gen(:, QG) < gen(:, QMIN) );
352
353             if ~isempty(mx) || ~isempty(mn)  %% we have some Q limit violations
354                 if isempty(pv)
355                     if verbose
356                         if ~isempty(mx)
357                             fprintf('Gen %d (only one left) exceeds upper Q ...
                                    limit : INFEASIBLE PROBLEM\n', mx);
358                         else
359                             fprintf('Gen %d (only one left) exceeds lower Q ...
                                    limit : INFEASIBLE PROBLEM\n', mn);
360                         end
361                     end
362                     success = 0;
363                     break;
364                 end
365
366                 %% one at a time?
367                 if qlim == 2    %% fix largest violation, ignore the rest
368                     [junk, k] = max([gen(mx, QG) - gen(mx, QMAX);
369                                      gen(mn, QMIN) - gen(mn, QG)]);
370                     if k > length(mx)
371                         mn = mn(k-length(mx));
372                         mx = [];
373                     else
374                         mx = mx(k);
375                         mn = [];
376                     end
377                 end
378
379                 if verbose && ~isempty(mx)
380                     fprintf('Gen %d at upper Q limit, converting to PQ bus\n', ...
                            mx);
381                 end
```

```
382                 if verbose && ~isempty(mn)
383                     fprintf('Gen %d at lower Q limit, converting to PQ bus\n', ...
                            mn);
384                 end
385
386                 %% save corresponding limit values
387                 fixedQg(mx) = gen(mx, QMAX);
388                 fixedQg(mn) = gen(mn, QMIN);
389                 mx = [mx;mn];
390
391                 %% convert to PQ bus
392                 gen(mx, QG) = fixedQg(mx);      %% set Qg to binding limit
393                 gen(mx, GEN_STATUS) = 0;        %% temporarily turn off gen,
394                 for i = 1:length(mx)            %% (one at a time, since
395                     bi = gen(mx(i), GEN_BUS);   %%  they may be at same bus)
396                     bus(bi, [PD,QD]) = ...      %% adjust load accordingly,
397                         bus(bi, [PD,QD]) - gen(mx(i), [PG,QG]);
398                 end
399                 if length(ref) > 1 && any(bus(gen(mx, GEN_BUS), BUS_TYPE) == REF)
400                     error('Sorry, MATPOWER cannot enforce Q limits for slack ...
                            buses in systems with multiple slacks.');
401                 end
402                 bus(gen(mx, GEN_BUS), BUS_TYPE) = PQ;   %% & set bus type to PQ
403
404                 %% update bus index lists of each type of bus
405                 ref_temp = ref;
406                 [ref, pv, pq] = bustypes(bus, gen);
407                 if verbose && ref ~= ref_temp
408                     fprintf('Bus %d is new slack bus\n', ref);
409                 end
410                 limited = [limited; mx];
411             else
412                 repeat = 0; %% no more generator Q limits violated
413             end
414         else
415             repeat = 0;     %% don't enforce generator Q limits, once is enough
416         end
417     end
418     if qlim && ~isempty(limited)
419         %% restore injections from limited gens (those at Q limits)
420         gen(limited, QG) = fixedQg(limited);    %% restore Qg value,
421         for i = 1:length(limited)               %% (one at a time, since
422             bi = gen(limited(i), GEN_BUS);      %%  they may be at same bus)
423             bus(bi, [PD,QD]) = ...              %% re-adjust load,
424                 bus(bi, [PD,QD]) + gen(limited(i), [PG,QG]);
425         end
426         gen(limited, GEN_STATUS) = 1;               %% and turn gen back on
427         if ref ~= ref0
428             %% adjust voltage angles to make original ref bus correct
429             bus(:, VA) = bus(:, VA) - bus(ref0, VA) + Varef0;
430         end
431     end
```

```
end

mpc.et = etime(clock, t0);
mpc.success = success;

%%-----  output results  -----

%% convert back to original bus numbering & print results
[mpc.bus, mpc.gen, mpc.branch] = deal(bus, gen, branch);
results = int2ext(mpc);

%% zero out result fields of out-of-service gens & branches
if ~isempty(results.order.gen.status.off)
  results.gen(results.order.gen.status.off, [PG QG]) = 0;
end
if ~isempty(results.order.branch.status.off)
  results.branch(results.order.branch.status.off, [PF QF PT QT]) = 0;
end

%% save solved case
if solvedcase
    savecase(solvedcase, results);
end

if nargout == 1 || nargout == 2
    MVAbase = results;
    bus = success;
elseif nargout > 2
    [MVAbase, bus, gen, branch, et] = ...
        deal(results.baseMVA, results.bus, results.gen, results.branch, ...
            results.et);
% else  %% don't define MVAbase, so it doesn't print anything
end
tempo_total(k)=toc(t1);
end

%% Print do Método de Resolução
legenda=[solver,reord,met,prec,met_d];
fprintf('%%%%%%%%%%%%%%%%%%%%%%%%% %s%s%s%s%s%%%%%%%%%%%%%%%%%%%%%%%%%\n', ...
    solver,reord,met,prec,met_d);
alg;
precond;

%% Cálculo dos Tempos
if exist('tempos_pf_tot.mat', 'file')==0
    save tempos_pf_tot
end

if length(tempo_pf)>5
    tempo_pf_medio=mean(tempo_pf(5:end))
    tempo_total_medio=mean(tempo_total(5:end));

```

```
    if alg==4
        tempo_1_iteracao=mean(t_iteracao1(5:end))
    end
else
    tempo_pf_medio=mean(tempo_pf)
    tempo_total_medio=mean(tempo_total);

    if alg==4
        tempo_1_iteracao=mean(t_iteracao1)
    end
end

%Organização dos tempos salvos e gráficos
if metodo_direto==0
    if reordenamento==2 & metodo==2 %RCM e GMRES
        i=1;
        color='green';
    elseif reordenamento==2 & metodo==1 %RCM e BiCGSTAB
        i=2;
        color='red';
    elseif reordenamento==1 & metodo==2 %AMD e GMRES
        i=3;
        color='black';
    elseif reordenamento==1 & metodo==1 %AMD e BiCGSTAB
        i=4;
        color='blue';
    end
else
    if metodo_direto==1;
        i=1;
        color='cyan';
    elseif metodo_direto==2;
        i=2;
        color='red';
    elseif metodo_direto==3;
        i=3;
        color='black';
    elseif metodo_direto==4;
        i=4;
        color='blue';
    end
end

if precond==0
    load tempos_pf_tot
    t_pf(i,alg)=tempo_pf_medio;
    t_tot(i,alg)=tempo_total_medio;
    save tempos_pf_tot t_pf t_tot t_precond
elseif precond==4
    load tempos_pf_tot
    t_pf(i,alg)=tempo_pf_medio;
    t_tot(i,alg)=tempo_total_medio;
```

```
534     t_precond(linha_res_pre+i,1)=tempo_pf_medio;
535     save tempos_pf_tot t_pf t_tot t_precond
536 elseif precond==1|precond==2|precond==3
537     load tempos_pf_tot
538     t_precond(linha_res_pre+i,precond+1)=tempo_pf_medio;
539     save tempos_pf_tot t_pf t_tot t_precond
540 end
541 
542 %% Plotar Gráficos
543 
544 if plotar_tempos==1
545     figure(alg)
546     plot(tempo_pf,color,'DisplayName',legenda)
547     xlabel('Repetição')
548     ylabel('Tempo (seg)')
549     hold on
550 end
551 
552 %% Plocar Gráficos Tempos barra
553 % figure(10)
554 % bar(testes,tempo_pf_medio)
555 % hold on
556 
557 %% Plotar Gráficos Droptol
558 if plotar_droptol==1
559     if alg==4|alg==5|alg==6|alg==7
560         figure(alg+7)
561         if alg==4 & droptol==0
562             plot(droptol,0.033,'--rs','MarkerSize',10)
563         else
564             plot(droptol,tempo_pf_medio,'--rs','MarkerSize',10)
565         end
566         xlabel('Droptol')
567         ylabel('Tempo (seg)')
568         hold on
569     end
570 end
571 
572 %% Verificar resultados obtidos
573 V_norma=norm(V)
574 iterations
575 droptol
576 %iterations_iter
577 end
578 end
579 end
580 end
581 end
582 
583 if plotar_tempos==1
584     legend('show')
585 end
```

```
586
587 end
588 testes
```