UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UTILIZAÇÃO DE MATRIZ SENSIBILIDADE E PROGRAMAÇÃO LINEAR PARA CORRIGIR OS NÍVEIS DE TENSÃO E OTIMIZAR A ALOCAÇÃO DE REATIVOS EM SISTEMAS DE POTÊNCIA

MARCOS ANTONIO PAES REZENDE

ORIENTADORA: ALESSANDRA MACEDO DE SOUZA

TRABALHO DE CONCLUSÃO DE CURSO DE ENGENHARIA ELÉTRICA

BRASÍLIA/DF: NOVEMBRO – 2008

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UTILIZAÇÃO DE MATRIZ SENSIBILIDADE E PROGRAMAÇÃO LINEAR PARA CORRIGIR OS NÍVEIS DE TENSÃO E OTIMIZAR A ALOCAÇÃO DE REATIVOS EM SISTEMAS DE POTÊNCIA

MARCOS ANTONIO PAES REZENDE

TRABALHO SUBMETIDO AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

APROVADA POR:

_____

Profª Alessandra Macedo de Souza, Dra (ENE-UnB)
(Orientadora)

_____

Prof. Francisco Damasceno Freitas, Dr (ENE-UnB)
(Examinador Interno)

_____

Prof. Luis Filomeno de Jesus Fernandes, Dr
(Examinador Externo)

BRASÍLIA/DF, 14 DE NOVEMBRO DE 2008

FICHA CATALOGRÁFICA

REZENDE, MARCOS ANTONIO PAES

Utilização de matriz sensibilidade e programação linear para corrigir os níveis de tensão e otimizar a alocação de reativos em sistemas de potência.

Publicação ENE-2/08, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 49p.

REFERÊNCIA BIBLIOGRÁFICA

REZENDE, M. A. P. (2008). Utilização De Matriz Sensibilidade E Programação Linear Para Corrigir Os Níveis De Tensão E Otimizar A Alocação De Reativos Em Sistemas De Potência. Trabalho de Conclusão de Curso, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 49p.

CESSÃO DE DIREITOS

AUTOR: Marcos Antonio Paes Rezende.

ORIENTADORA: Alessandra Macedo de Souza.

TÍTULO: Utilização de matriz sensibilidade e programação linear para corrigir os níveis de tensão e otimizar a alocação de reativos em sistemas de potência.

ANO: 2008

DEDICATÓRIA

A toda minha família

AGRADECIMENTOS

A Deus, por tudo que aconteceu em minha vida.

Aos meu pais pelo apoio durante todo curso.

Ao meu irmão pela amizade.

A Cristiane pela ajuda com português.

A professora e orientadora Alessandra pela paciência e compreensão.

RESUMO

Alocação ótima de reativos em sistemas de potência utilizando matriz sensibilidade e otimização via programação linear no MATLAB

AUTOR: Marcos Antonio Paes Rezende.

ORIENTADORA: Alessandra Macedo de Souza.

Brasília – Outubro de 2008

Este trabalho apresenta um método computacional para correção dos níveis de tensão de um sistema elétrico. As tensões são corrigidas com a alocação de reativos, usando a matriz sensibilidade do sistema para calcular a quantidade de reativos que deverão ser alocados e definir onde eles serão alocados.

O programa foi desenvolvido no MATLAB, utilizando a função *linprog*, que resolve problemas de otimização utilizando programação linear. Para a resolução do fluxo de carga foi adotado o método de Newton.

O programa foi testado em 3 sistemas conhecidos de 6, 14 e 30 barras e os resultados encontrados foram satisfatórios. É possível escolher os limites de tensão desejados e o método de otimização. Também podem ser escolhidas as tolerâncias desejadas para a resolução do fluxo de carga e para a alocação de reativos.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1
INTRODUÇÃO

1.1 – Aspectos gerais

Com o aumento dos sistemas de potência impulsionado pelo crescimento dos centros consumidores, torna-se necessário um cuidadoso planejamento para o fornecimento de energia de qualidade. Este planejamento envolve, entre outros assuntos, a alocação de potência reativa nos barramentos para a manutenção das tensões dentro de limites aceitáveis.

É desejável, para que os custos da implantação não sejam grandes, que seja alocado o mínimo possível de reativos. Métodos utilizados para a alocação de reativos são utilizados desde a década de 50. Eles utilizavam métodos analíticos a partir de simplificações do modelo do sistema de potência. Com a popularização dos computadores, métodos numéricos começaram a ser utilizados a partir da década de 70, (SOUSA, 2003).

Hoje, para o controle de tensão, existem métodos bem sofisticados que utilizam inteligência artificial através de Redes Neurais Artificiais (RNA) ou Lógica *Fuzzy* para a decisão, em tempo real, na atuação nos equipamentos de controle de tensão do sistema de potência (LIMA, 2007). Também existem técnicas de controle que incluem restrições dinâmicas aos geradores para um melhor planejamento dos locais e valores dos suportes de potência reativa, (OLIVEIRA, 2008).

Existe também outro método, que utiliza a alocação de reativos não só para obter tensões dentro de limites desejados, mas também para maximizar a capacidade de transferência de potência no sistema elétrico, (GOH, 2006).

Outro método com motivação mais econômica para a alocação de reativos em sistemas de potência com vários participantes não propõe uma solução de custo mínimo global, mas alocações de modo que cada participante cubra as suas perdas, (CHICCO, 2002).

No trabalho apresentado, para a otimização da alocação de reativos nos sistemas de potência, foi proposta uma metodologia que utiliza métodos numéricos, baseada na utilização de uma

matriz que relaciona o impacto nas tensões do sistema com a alocação de potência reativa em uma barra. Essa matriz é chamada matriz sensibilidade. Este método foi baseado no descrito por SOUZA em 1997. A matriz sensibilidade é calculada no ponto de solução do problema de fluxo de carga. Tanto a matriz sensibilidade como o fluxo de carga foram calculados utilizando o programa MATLAB. Para a alocação ótima de reativos, foi usada a função *linprog* do MATLAB, que permite escolher dois métodos distintos de otimização. O programa de otimização utiliza as mesmas variáveis do programa de fluxo de carga e estão ambos embutidos no arquivo maloca.m mostrado no apêndice B.

O programa foi testado utilizando três sistemas: Ward&Hale de 6 barras, IEEE-14 e IEEE-30. Os dados desses sistemas estão disponíveis no apêndice A.

1.2 – Organização do trabalho

O trabalho foi dividido em 5 capítulos. No primeiro, foi feita uma breve introdução ao problema de alocação de reativos e a apresentação dos objetivos do trabalho.

No segundo capítulo são apresentadas as formulações matemáticas necessárias para a resolução do fluxo de carga e para a alocação ótima de reativos

No capítulo seguinte é mostrado como a otimização será realizada no MATLAB.

Os resultados obtidos nos três sistemas testados utilizando o programa desenvolvido são apresentados no capítulo 4.

As conclusões do trabalho são apresentadas no capítulo final.

Nos apêndices são mostrados os dados dos sistemas utilizados, bem como todo código implementado.

CAPÍTULO 2
FLUXO DE CARGA COM ALOCAÇÃO DE REATIVOS

2.1 – Introdução

O problema de fluxo de carga fornece como solução o ponto de operação de um sistema de potência, isto é, a magnitude e a fase da tensão para todas as barras da rede elétrica. O fluxo de carga é obtido da aplicação da lei dos nós no sistema de potência. Para cada barra podem ser determinadas duas equações:

$$P_k = \sum_{m \in \Omega_k} P_{km}(V_k, V_m, \theta_k, \theta_m) \tag{2.1}$$

$$Q_k + Q_k^{sh}(V_k) = \sum_{m \in \Omega_k} Q_{km}(V_k, V_m, \theta_k, \theta_m) \tag{2.2}$$

Em que:

- Ω_k é o conjunto das barras vizinhas a barra k;
- V_k é a magnitude do fasor de tensão na barra k;
- θ_k é o ângulo do fasor de tensão na barra k;
- V_m é a magnitude do fasor de tensão na barra m;
- θ_m é o ângulo do fasor de tensão na barra m;
- P_k é a geração líquida de potência ativa na barra k (geração menos carga);
- Q_k é a injeção líquida de potência reativa na barra k;
- Q_k^{sh} é a injeção de potência reativa na barra k devido a um elemento shunt conectado a barra;

- P_{km} é o fluxo de potência ativa da barra k para a barra m;

- Q_{km} é o fluxo de potência reativa da barra k para a barra m.

Os fluxos P_{km} e Q_{km} podem ser obtidos a partir do modelo π utilizado para as linhas de transmissão e para os transformadores, que são dados pelas expressões 2.3 e 2.4 (MONTICELLI, 1983). Para as linhas com um transformador o modelo adotado é o mostrado na Figura 2.1.

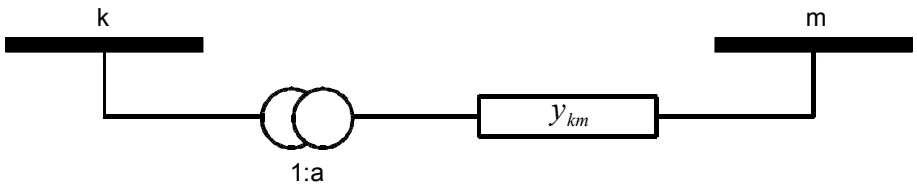

Figura 2.1 – Modelo de linha com transformador

$$P_{km} = (a_{km}V_k)^2 g_{km} - a_{km}V_k V_m g_{km} \cos\theta_{km} - a_{km}V_k V_m b_{km} sen\,\theta_{km} \quad (2.3)$$

$$Q_{km} = -(a_{km}V_k)^2 (b_{km} + b_{km}^{sh}) + a_{km}V_k V_m b_{km} \cos\theta_{km} - a_{km}V_k V_m g_{km} sen\,\theta_{km} \quad (2.4)$$

Em que:

- a_{km} é a relação de transformação do transformador, conectado na barra k, da linha de transmissão que vai da barra k para m, se não há transformador na barra k $a_{km}=1$;

- g_{km} é a condutância série da linha de transmissão que liga as barras k e m. É a parte real da admitância da linha (y_{km});

- b_{km} é a susceptância série da linha de transmissão que liga as barras k e m. É a parte imaginária da admitância da linha (y_{km});

- θ_{km} é a diferença angular entra as barras k e m ($\theta_k - \theta_m$);

Combinando as equações 2.3 e 2.4 com as equações 2.1 e 2.2 obtemos as duas equações básicas do fluxo de carga mostradas a seguir:

$$P_k = V_k \sum_{m \in K} V_m (G_{km} \cos\theta_{km} + B_{km} sen\theta_{km}) \quad (2.5)$$

$$Q_k = V_k \sum_{m \in K} V_m (G_{km} sen\theta_{km} - B_{km} \cos\theta_{km}) \quad (2.6)$$

Em que o conjunto K é formado pelo conjunto das barras adjacentes à barra k (Ω_k) mais a própria barra k. As matrizes representadas por G e B são formadas pelas componentes reais e imaginárias da matriz admitância e representam a condutância e a susceptância, respectivamente.

Para cada barra temos duas equações e quatro incógnitas (P, Q, V e θ). Sendo assim, seria impossível encontrar uma solução única para o problema. Acontece que toda barra tem, pelo menos, duas dessas incógnitas determinadas pelas características delas no sistema elétrico. Serão considerados três tipos de barras:

- Barras PQ, também conhecidas como barras de carga, são as barras onde, geralmente, estão concentradas as unidades consumidoras. Nelas, o fasor de tensão não é fixado, mas a carga consumida é conhecida através de estudos prévios. Dados P e Q, calculados V e θ.

- Barras PV, também chamadas de barras geradoras ou de tensão controlada, são as barras onde estão conectados os geradores. A potência gerada por um gerador e sua tensão é determinada pelo operador do sistema. Dados P e V, calculados Q e θ.

- Barras Vθ, também conhecidas como barras de folga ou *slack*, são as barras onde é especificado um ângulo de referência e a tensão é fixa. A potência gerada por ela não é fixada, pois ela é a barra que irá suprir a potência consumida por perdas de transmissão. Dados V e θ, calculados P e Q.

Utilizando estes três tipos de barra, é possível eliminar algumas incógnitas das equações 2.5 e 2.6. O sistema elétrico de n barras fica, então, com 2n equações e 2n variáveis, existindo, assim, uma solução única para o sistema de equações.

2.2 – Resolução do problema de fluxo de carga pelo método de Newton

Uma vez obtido o sistema com 2n equações e 2n incógnitas, o método de resolução utilizado se baseia em dividir estas equações em dois subsistemas: um onde serão obtidos os fasores de tensões das barras e outro onde será obtida a potência complexa consumida ou fornecida em cada barra. O primeiro subsistema será resolvido utilizando o método de Newton. O segundo, utilizando as expressões de injeção de potência das equações 2.5 e 2.6. O primeiro subsistema exige mais esforço computacional, pois é um método iterativo que requer uma inversão matricial, procedimento muito dispendioso em termos computacionais, a cada iteração.

2.2.1 – O Método de Newton

Para a resolução do sistema foi usado o método de Newton para sistemas de equações não-lineares. Este método pode ser considerado uma ampliação do método Newton-Raphson, que resolve equações a partir de uma aproximação linear e iterativa, (RUGGIERO, 2004).

O método de Newton-Raphson é utilizado para encontrar zeros de funções da seguinte maneira:

Seja a equação f(x) = 0.

1. Dados iniciais necessários: aproximação inicial x_0 e critério de parada ε, k=0;

2. Verificar se $|f(x_k)| \leq \varepsilon$, se sim, o método convergiu para x_k com a precisão desejada, se não, ir para o passo 3

3. Atualizar o valor de x a partir da forma de iteração:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{2.7}$$

4. k=k+1

5. Voltar ao passo 2.

Utilizando este procedimento, haverá convergência em um intervalo em torno do zero da equação se $f(x)$, $f^{'}(x)$ e $f^{''}(x)$ forem contínuas. O procedimento de resolução do sistema de equações não-lineares é mostrado a seguir:

Seja um sistema de equações dado por:

$$F(\bar{x}) = \begin{pmatrix} f_1(x_1, x_2, \cdots, x_n) \\ f_2(x_1, x_2, \cdots, x_n) \\ \vdots \\ f_n(x_1, x_2, \cdots, x_n) \end{pmatrix} \tag{2.8}$$

Deseja-se encontrar:

$$F(\bar{x}) = 0 \tag{2.9}$$

O procedimento de resolução é similar. Uma diferença é que, ao invés da utilização de uma derivada simples, é utilizado o gradiente das equações. A matriz formada pelos gradientes das equações é denominada matriz Jacobiana e é mostrada a seguir:

$$J(\bar{x}) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix} \tag{2.10}$$

O método de Newton para encontrar $\bar{x}$ tal que F($\bar{x}$)=0 consiste nos seguintes passos:

1. Dados iniciais necessários: aproximação inicial $\overline{x_0}$ e critério de parada ε, k=0;

2. Calcule $F(\overline{x_k})$

3. Verificar se $\left|F(\overline{x_k})\right| \leq \varepsilon$, se sim o método convergiu para $\bar{x} = \overline{x_k}$ com a precisão desejada, se não ir para o passo 4;

4. Calcular a matriz Jacobiana ($J(\overline{x_k})$);

5. Obter o vetor de correções $\Delta x = -[J(x)]^{-1} F(x)$, solução do sistema linear: $J(\overline{x_k})\, \Delta x = - F(\overline{x_k})$;

6. Calcular o novo ponto $x_{k+1} = x_k + \Delta x$;

7. k=k+1;

8. Voltar para o passo 2

2.2.2 – O método de Newton aplicado ao problema de Fluxo de Carga

O subsistema 1, onde será encontrado V e θ é composto pelas equações:

$$P_k^{esp} - V_k \sum_{m \in K} V_m (G_{km} \cos\theta_{km} + B_{km} sen\theta_{km}) = 0 \tag{2.11}$$

$$Q_k^{esp} + V_k \sum_{m \in K} V_m (G_{km} sen\theta_{km} - B_{km} \cos\theta_{km}) = 0 \tag{2.12}$$

Em que:

- P_k^{esp} é a potência ativa gerada ou consumida na barra k. Esta potência é conhecida nas barras PV e PQ;

- Q_k^{esp} é a potência reativa consumida na barra k. Esta potência é conhecida PQ;

Sendo que para as barras PQ são resolvidas as equações 2.11 e 2.12 e para as barras PV somente a equação 2.11, resultando num sistema com 2NPQ + NPV equações. O sistema de equações será representado por:

$$F(\bar{x}) = \begin{bmatrix} P_i^{esp} - Vi\sum_{m\in K} V_m(G_{im}\cos\theta_{im} + B_{im}sen\theta_{im}) \\ Q_j^{esp} + V_j\sum_{m\in K} V_m(G_{jm}sen\theta_{jm} - B_{jm}\cos\theta_{jm}) \end{bmatrix} \tag{2.13}$$

Em que $\bar{x} = \begin{bmatrix} \theta_i \\ V_j \end{bmatrix}$, e i é o conjunto formado pelas barras PV e PQ e j o conjunto das barras PQ.

Para a resolução do sistema de equações, aplica-se o método de Newton, como mostrado na seção anterior. As aproximações iniciais são feitas iguais a 1 p.u.para as tensões e 0 para os ângulos. Esta aproximação é razoável porque os valores no sistema de potência não diferem muito desses. Nesse caso específico, o Jacobiano $J(\bar{x})$ é determinado por:

$$J(\bar{x}) = -\begin{bmatrix} \frac{\partial\overline{P_i}}{\partial\overline{\theta_i}} & \frac{\partial\overline{P_i}}{\partial\overline{V_j}} \\ \frac{\partial\overline{Q_j}}{\partial\overline{\theta_i}} & \frac{\partial\overline{Q_j}}{\partial\overline{V_j}} \end{bmatrix} \tag{2.14}$$

Para leitura mais fácil, as submatrizes da matriz Jacobiana são representadas por:

$$J(\bar{x}) = \begin{bmatrix} H & N \\ M & L \end{bmatrix} \tag{2.15}$$

Existem expressões gerais para as matrizes H, M, N e L, sendo que não é necessário aplicar a derivada em toda iteração. Nesta situação deve-se utilizar as equações de 2.16 a 2.23 a seguir:

Submatriz H:

$$H_{km} = V_k V_m(G_{km}sen\theta_{km} - B_{km}\cos\theta_{km}) \tag{2.16}$$

$$H_{kk} = -Q_k - V_k^2 B_{kk} \tag{2.17}$$

Submatriz M

$$M_{km} = -V_k V_m(G_{km}\cos\theta_{km} + B_{km}sen\theta_{km}) \tag{2.18}$$

$$M_{kk} = P_k - V_k^2 G_{kk} \tag{2.19}$$

Submatriz N

$$N_{km} = V_k(G_{km}\cos\theta_{km} + B_{km}sen\theta_{km}) \tag{2.20}$$

$$N_{kk} = V_k^{-1}(P_k + V_k^2 G_{kk}) \tag{2.21}$$

Submatriz L

$$L_{km} = V_k(G_{km}sen\theta_{km} - B_{km}\cos\theta_{km}) \tag{2.22}$$

$$L_{kk} = V_k^{-1}(Q_k - V_k^2 B_{kk}) \tag{2.23}$$

Após aplicar o método de Newton, são obtidos todos os fasores de tensão do sistema. Com V e θ determinados, as injeções de potência em cada barra podem ser calculados pelas equações 2.5 e 2.6. Uma vez determinadas todas as variáveis, o problema do fluxo de carga está resolvido.

2.3 – Alocando reativos de maneira ótima

Após a resolução do problema do fluxo de carga, algumas tensões podem estar fora de limites para adequados para prejudiquem o bom funcionamento do sistema. Este trabalho propõe uma metodologia para a correção das tensões alocando reativos nas barras. Os reativos são alocados de maneira a se otimizar quantidade de reativos para a correção das tensões dentro dos limites desejados.

2.3.1 – Motivação

Caso uma barra se encontre com tensão muito elevada, equipamentos da subestação e/ou do consumidor final podem ser danificados. Por outro lado, se a tensão estiver baixa alguns componentes de eletrônica de potência e motores podem não funcionar corretamente. Por

estes motivos, é necessário que as tensões em todas as barras do sistema de potência estejam dentro de limites aceitáveis.

Uma solução para a correção dos níveis de tensão é alocar fontes de reativos nas barras. Como as magnitudes das tensões estão ligadas fortemente à transferência de potência reativa entre as barras, a alocação de reativos em uma barra altera o fluxo de potência reativa necessário e, portanto, as tensões nas barras. Estes reativos podem ser alocados com capacitores e reatores em paralelo com a barra, compensadores síncronos e estáticos ou atuando nos próprios geradores, (GOH, 2006), (CHICCO, 2002).

2.3.2 – Cálculo dos reativos a serem alocadas a partir da matriz sensibilidade

Usualmente, para corrigir as tensões fora dos limites, os reativos são alocados nas barras onde as tensões estão abaixo do limite desejado. Esta metodologia corrige os níveis de tensão, mas não garante que será alocado o mínimo de reativos para isto. O método proposto utiliza uma matriz sensibilidade que relaciona, para o ponto resultante do fluxo de carga, quanto uma variação na injeção de reativos em uma barra gera de variação de tensão nas outras barras. Por simplicidade, só serão consideradas injeções em barras PQ.

A matriz sensibilidade utilizada ($\frac{\partial V}{\partial Q}$) pode ser obtida calculando a inversa da submatriz L do jacobiano ($\frac{\partial Q}{\partial V}$). A submatriz L será calculada utilizando as equações 2.22 e 2.23 e será invertida para a obtenção da matriz sensibilidade desejada. Com esta matriz, é possível saber onde a alocação de reativos é mais sensível, resultando em alocações nas barras que mais afetam as tensões a serem corrigidas.

Assim resolve-se o seguinte problema de otimização, também chamado de problema de programação linear (PL):

$$
\begin{aligned}
&\textit{Minimizar} && \sum_{j} \Delta Q \\
&\textit{Sujeito a} && S * \Delta Q \geq V_j^{\min} - V_j \\
& && S * \Delta Q \leq V_j^{\max} - V_j \\
& && \Delta Q \geq 0
\end{aligned}
\tag{2.24}
$$

Em que:

- j representa as barras onde podem ser alocados reativos;
- S é a matriz sensibilidade;
- ΔQ é um vetor com todas alocações feitas nas barras j.

Resolvendo este problema, obtemos as injeções de potência reativa ΔQ_j que minimiza a potência a ser alocada para manter o nível de tensão entre $V^{\min}$ e $V^{\max}$. Com as devidas injeções de potência reativa, é necessário acrescentar essas potências no problema de fluxo de carga e novamente resolvê-lo para obter o estado resultante do sistema após as alocações de reativos.

Um problema que acontece é devido à simplificação feita anteriormente de calcular S somente para o ponto desejado. Como S não considera as não-linearidades do sistema e foi calculada para um ponto específico, caso seja necessário a injeção de muitos reativos, o comportamento do sistema poderá mudar. Entretanto, no problema de otimização, a matriz sensibilidade é considerada a mesma. Portanto, é possível que, após a resolução do fluxo de carga, algumas tensões estejam fora dos limites, mesmo sendo uma restrição do problema de otimização que as tensões devem estar dento dos limites desejados.

Este problema foi solucionado recalculando a matriz sensibilidade para o novo estado do sistema resultante da resolução do fluxo de carga com as alocações obtidas na resolução do PL. O novo estado do sistema, embora ainda fora dos limites desejados, encontra-se mais próximo do estado onde as tensões estão dentro dos limites, pois já foi resolvido um PL visando corrigir as tensões do sistema. A cada iteração, as tensões ficam mais próximas dos limites desejados e há a convergência para um estado onde as tensões estejam dentro dos

limites. Como a convergência é lenta, é possível parar as iterações caso se esteja próximo o suficiente das tensões desejadas.

A metodologia proposta é representada pelo fluxograma, representado na Figura 2.2.

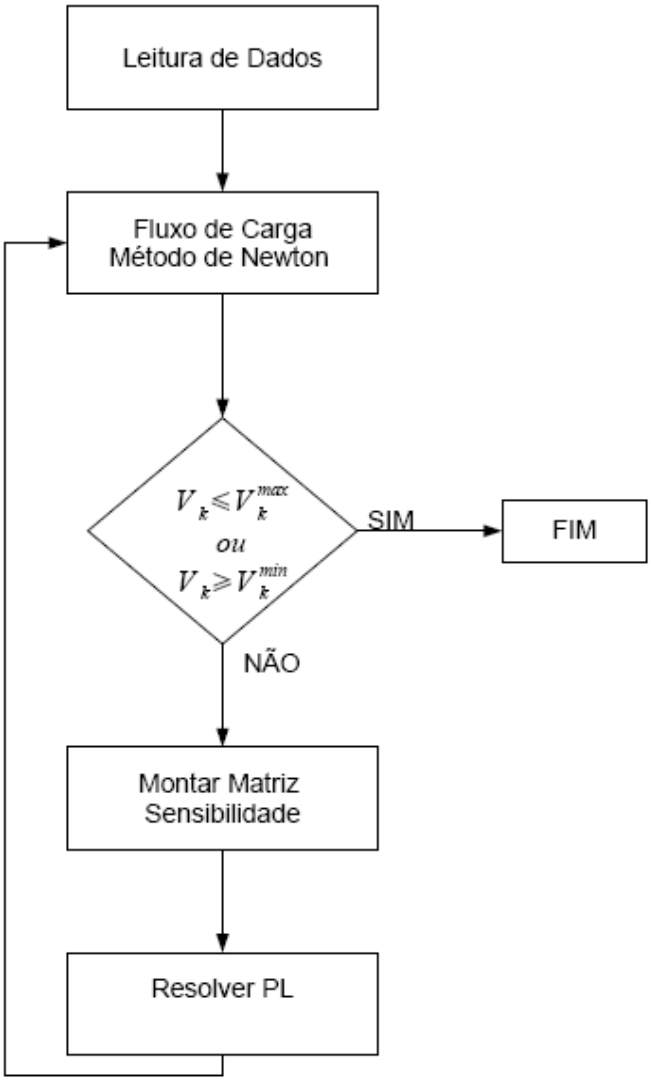

Figura 2.2 – Diagrama de fluxo do programa

CAPÍTULO 3
OTIMIZAÇÃO NO MATLAB

3.1 – Considerações iniciais

A *toolbox* de otimização presente no MATLAB apresenta diversas funções que podem ser aplicadas em diversos problemas de otimização. Para o problema da otimização dos reativos a serem alocados, que tem uma função linear a ser minimizada (a soma dos reativos alocados), a função mais simples para ser utilizada é a *linprog*. Existem outras funções mais gerais, como por exemplo, *fmincon*, *fminunc*, *quadprog* entre outras, que geram os mesmos resultados, mas como estas exigem mais dados de entrada, será utilizada, por simplicidade, a *linprog*. A *linprog* resolve problemas do tipo:

$$\begin{aligned} &Minimizar && f(x) \\ &Sujeito\ a && A*x \geq b \\ & && Aeq*x = beq \\ & && lb \leq x \leq ub \end{aligned} \tag{3.1}$$

A função *linprog* resolve os problemas de programação linear utilizando dois algoritmos distintos. O padrão é o denominado *Large-Scale,* que utiliza o método de ponto interior Primal-Dual. Outro método que pode ser utilizado é o método simplex, um dos métodos mais conhecidos para otimização. O método simplex encontra o mínimo da função percorrendo o poliedro formado pelas restrições que são feitas à função a ser minimizada. O Método de Ponto Interior pode percorrer um caminho que contenha pontos interiores ao poliedro, incluído o próprio poliedro, sendo, assim, mais abrangente que o simplex. Mais informações sobre estes métodos podem ser encontradas em DANTZIG (1955) e MEHROTRA (1992). A escolha do método que será utilizado é feita com a função *optimset*

No programa desenvolvido, pode-se escolher qual método será utilizado. O método simplex será usado como padrão, pois resulta numa otimização mais rápida, embora não tão abrangente quanto à do método de ponto interior.

3.2 – O uso da função *linprog*

A função *linprog* deve ser chamada utilizando, pelo menos, três parâmetros:

- Os coeficientes "f" da função linear a ser utilizada;
- A matriz "A" com os coeficientes do sistema de desigualdades, esse sistema é da forma $A * x \leq b$;
- O vetor coluna "b" dos termos independentes do sistema de desigualdades.

A chamada da função é feita utilizando o seguinte comando no MATLAB:

```
>>xmin = linprog(f,A,b)
```

Dessa forma os valores de x, {x1, x2, x3, ...} que minimizam a função linear f(x), de n variáveis, determinada pelos coeficientes "f" é armazenado na variável xmin, sendo que a xmin é um vetor coluna com os valores que, aplicados à função f, a minimizam.

Adicionalmente, é possível incluir restrições de igualdade, limites superiores e inferiores para os valores de x, e incluir um ponto inicial (desprezado, se o método utilizado for o simplex), além de diversas opções sobre a execução da rotina de otimização (método de otimização, tolerância, e métodos de cálculo de diversas matrizes utilizadas na otimização). Estas novas restrições são passadas à função utilizando os seguintes parâmetros:

- A matriz "Aeq" com os coeficientes do sistema de igualdades, esse sistema é da forma $Aeq * x \leq beq$;
- O vetor coluna "beq" dos termos independentes do sistema de igualdades;
- O vetor coluna "lb", com os limites inferiores que os valores de x podem assumir;
- O vetor coluna "ub", com os limites superiores que os valores de x podem assumir;
- O vetor "x0" com o ponto inicial para o método de ponto interior, este vetor é desprezado no método simplex e, se deixado em branco na utilização do método de ponto interior, o sistema calcula um ponto inicial.

- As opções são passadas com a função *optimset.* O uso do *optimset* é feito da seguinte maneira:

  >>options=optimset("variável", "valor")

  Existem 44 variáveis que podem ter seu valor ajustados pelo *optimset*. No programa só serão utilizadas as simplex e *Large-Scale*, utilizadas para escolha do método de otimização. Esta escolha é feita com os seguintes comandos:

  >>options = optimset('LargeScale', 'off', 'Simplex', 'on'), para a utilização do método simplex.

  >>options = optimset('LargeScale', 'on', 'Simplex', 'off'), para a utilização do método *Large-Scale.*

A chamada completa para a função fica então da seguinte maneira:

>>xmin = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)

É importante notar que a ordem em que os argumentos são passados à função não pode mudar. Se não há uma restrição que é passada como argumento da função antes de outra que existe, aquela deve ser indicada com um vetor vazio.

No problema proposto (eq. 2.24), só existem restrições de desigualdades e, por isso, o uso da função *linprog* será feito apenas com os três argumentos mínimos. Caso fossem propostas mais restrições, elas deveriam ser acrescentadas utilizando a mesma lógica mostrada anteriormente.

CAPÍTULO 4
RESULTADOS OBTIDOS

4.1 – INTRODUÇÃO

Para testar o que foi proposto, foi elaborado um programa em MATLAB que implementa o que já foi discutido anteriormente neste trabalho. Este programa foi testado em variações dos sistemas WARD&HALE de 6 barras, IEEE 14 e 30 barras. Como todos estes sistemas apresentam barras de tensão controlada com tensões acima do limite de 5% da tensão nominal, foi utilizado, por coerência, o limite de tensão de ±5% da tensão nominal.

Os sistemas, nas situações de carga descritas, apresentam as tensões dentro ou bem perto do limite, sendo necessário nenhuma ou pouca potência reativa alocada para corrigir as tensões. Para mostrar melhor o comportamento do programa, as cargas consideradas foram maiores para gerar mais tensões fora do limite de 5%. Assim, podemos ver melhor a decisão, tomada pelo programa, de onde alocar os reativos.

O programa foi rodado utilizando o método simplex e o *Large-Scale*, mas, em ambos os casos, os resultados obtidos foram os mesmos.

4.2 – Resultado do sistema WARD&HALE de 6 barras

O sistema Ward&Hale de seis barras (W&H6) é o mostrado no apêndice A - 1. Esse sistema foi modificado aumentando as cargas em 50%. Após a solução do fluxo de carga foram encontradas as tensões mostradas na Tabela 4.1:

Tabela 4.1 – Tensões encontradas ao resolver o fluxo de carga do sistema W&H6

Barra	Tipo	Tensão (p.u.)
1	Vθ	1,0500
2	PV	1,1000
3	PQ	0,7184
4	PQ	0,8019
5	PQ	0,7481
6	PQ	0,7590

Com os limites propostos, as tensões devem estar entre 0,95 p.u. e 1,05 p.u. (±5%). Em todas as barras PQ as tensões estão fora do intervalo desejado. A rotina de otimização será então aplicada ao sistema. A matriz sensibilidade deste sistema tem a forma mostrada na eq. 4.1.

$$S = \begin{bmatrix} \frac{\partial V_3}{\partial Q_3} & \frac{\partial V_3}{\partial Q_4} & \frac{\partial V_3}{\partial Q_5} & \frac{\partial V_3}{\partial Q_6} \\ \frac{\partial V_4}{\partial Q_3} & \frac{\partial V_4}{\partial Q_4} & \frac{\partial V_4}{\partial Q_5} & \frac{\partial V_4}{\partial Q_6} \\ \frac{\partial V_5}{\partial Q_3} & \frac{\partial V_5}{\partial Q_4} & \frac{\partial V_5}{\partial Q_5} & \frac{\partial V_5}{\partial Q_6} \\ \frac{\partial V_6}{\partial Q_3} & \frac{\partial V_6}{\partial Q_4} & \frac{\partial V_6}{\partial Q_5} & \frac{\partial V_6}{\partial Q_6} \end{bmatrix} \tag{4.1}$$

O programa calcula a matriz sensibilidade para o ponto encontrado após a resolução do fluxo de carga mostrada na Tabela 4.1. A matriz sensibilidade encontrada é:

$$S = \begin{bmatrix} 0{,}4046 & 0{,}2625 & 0{,}1119 & 0{,}1405 \\ 0{,}2931 & 0{,}3006 & 0{,}1281 & 0{,}1609 \\ 0{,}1097 & 0{,}1126 & 0{,}5323 & 0{,}2807 \\ 0{,}1398 & 0{,}1434 & 0{,}2848 & 0{,}3577 \end{bmatrix} \tag{4.2}$$

O problema de otimização para esse caso, obtido a partir da equação 2.24, é mostrado na equação 4.3. Os valores nas colunas a direita das inequações representam as variações de tensão desejadas para as tensões ficarem acima da tensão mínima e abaixo da tensão máxima estabelecida:

$$
\begin{aligned}
&\textit{Minimizar} \quad && \Delta Q_3 + \Delta Q_4 + \Delta Q_5 + \Delta Q_6 \\
&\textit{Sujeito a} \quad && \begin{bmatrix} 0{,}4046 & 0{,}2625 & 0{,}1119 & 0{,}1405 \\ 0{,}2931 & 0{,}3006 & 0{,}1281 & 0{,}1609 \\ 0{,}1097 & 0{,}1126 & 0{,}5323 & 0{,}2807 \\ 0{,}1398 & 0{,}1434 & 0{,}2848 & 0{,}3577 \end{bmatrix} * \begin{bmatrix} \Delta Q_3 \\ \Delta Q_4 \\ \Delta Q_5 \\ \Delta Q_6 \end{bmatrix} \geq \begin{bmatrix} 0{,}2316 \\ 0{,}1481 \\ 0{,}2019 \\ 0{,}1910 \end{bmatrix} \\
& && \begin{bmatrix} 0{,}4046 & 0{,}2625 & 0{,}1119 & 0{,}1405 \\ 0{,}2931 & 0{,}3006 & 0{,}1281 & 0{,}1609 \\ 0{,}1097 & 0{,}1126 & 0{,}5323 & 0{,}2807 \\ 0{,}1398 & 0{,}1434 & 0{,}2848 & 0{,}3577 \end{bmatrix} * \begin{bmatrix} \Delta Q_3 \\ \Delta Q_4 \\ \Delta Q_5 \\ \Delta Q_6 \end{bmatrix} \leq \begin{bmatrix} 0{,}3316 \\ 0{,}2481 \\ 0{,}3019 \\ 0{,}2910 \end{bmatrix} \\
& && \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \Delta Q_3 \\ \Delta Q_4 \\ \Delta Q_5 \\ \Delta Q_6 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\end{aligned}
\tag{4.3}
$$

Para a resolução do problema no MATLAB foi usada a função *linprog* como mostrado no capítulo 3. Como as inequações para a função *linprog* são do tipo A*x ≤ b, os termos onde aparecem A*x ≥ b serão multiplicados por -1.

A matriz A que contem os coeficientes do sistema de desigualdades é então criada utilizando o seguinte comando no MATLAB:

```
>>A = [-1.*S; S; -1*eye(NPQ)];
```

Onde S é a matriz sensibilidade e a função eye retorna uma matiz identidade com dimensões NPQxNPQ.

O vetor-coluna com os termos independentes é criado utilizando o seguinte comando:

```
>>b = [vm(iq)-Vmin; Vmax-vm(iq); zeros(size(iq))];
```

A função zeros retorna uma matriz somente com zeros com a dimensão desejada. A função size(iq) retorna as dimensões da matriz iq, que é um vetor-coluna com os números das barras PQ.

A função a ser minimizada, a soma dos reativos injetados, é dada por $\Delta Q_3 + \Delta Q_4 + \Delta Q_5 + \Delta Q_6$, logo os coeficientes são todos unitários. Um vetor com estes coeficientes é criado utilizando o seguinte comando:

```
>>f=ones(size(iq));
```

A função ones retorna uma matriz somente com todos valores sendo unitários, com a dimensão passada como argumento. Para o MATLAB os coeficientes da função a ser minimizada devem ser passados como um vetor-linha.

A função *linprog* pode ser chamada utilizando o seguinte comando:

```
>>xmin = linprog(f,A,b);
```

Os valores das injeções que resolvem o problema de otimização são guardados na variável xmin. Estes valores são adicionados as potências reativas líquidas injetadas em cada barra com o comando:

```
>>QN(iq) = QN(iq) + xmin;
```

Após isso é novamente calculado o fluxo de carga, com as novas injeções de potência, para a obtenção do estado do sistema após a alocação de reativos. O estado do sistema resultante após a alocação é mostrado na Tabela 4.2, a coluna com o ΔQ representa as alocações encontradas após a otimização.

Tabela 4.2 – Tensões encontradas ao resolver o fluxo de carga após a primeira alocação de reativos do sistema W&H6

Barra	Tipo	Tensão (p.u.)	ΔQ (p.u.)
1	Vθ	1,0500	-
2	PV	1,1000	-
3	PQ	0,9334	0,4479
4	PQ	0,9930	0,0000
5	PQ	0,9327	0,1685
6	PQ	0,9606	0,2247

Neste caso, mesmo após a alocação, as tensões nas barras 3 e 5 ainda estão fora do limite de 5% desejado, mas bem mais próximo do que no caso inicial. Nessa situação, a matriz sensibilidade é calculada para o estado resultante. A matriz sensibilidade neste ponto é:

$$S = \begin{bmatrix} 0{,}2630 & 0{,}1789 & 0{,}0632 & 0{,}0845 \\ 0{,}1903 & 0{,}2179 & 0{,}0770 & 0{,}1029 \\ 0{,}0610 & 0{,}0698 & 0{,}3628 & 0{,}1794 \\ 0{,}0839 & 0{,}0960 & 0{,}1848 & 0{,}2468 \end{bmatrix} \tag{4.4}$$

O processo de otimização é novamente aplicado, utilizando a matriz sensibilidade mostrada na equação 4.4. As tensões encontradas após uma segunda alocação de reativos são mostradas na Tabela 4.3

Tabela 4.3 – Tensões encontradas ao resolver o fluxo de carga após alocação de reativos do sistema W&H6

Barra	Tipo	Tensão (p.u.)	ΔQ (p.u.)
1	Vθ	1,0500	-
2	PV	1,1000	-
3	PQ	0,9502	0,5017
4	PQ	1,0068	0,0000
5	PQ	0,9496	0,2070
6	PQ	0,9732	0,2247

Somente tensão na barra 5 ainda está fora do limite de 5%, sendo necessária mais uma alocação. Nesta última alocação foram alocados menos reativos do que na primeira, pois os níveis já estavam mais próximos do desejado. Para este novo estado do sistema a matriz sensibilidade é dada por:

$$S = \begin{bmatrix} 0{,}2554 & 0{,}1744 & 0{,}0605 & 0{,}0820 \\ 0{,}1848 & 0{,}2134 & 0{,}0741 & 0{,}1004 \\ 0{,}0584 & 0{,}0675 & 0{,}3508 & 0{,}1744 \\ 0{,}0812 & 0{,}0938 & 0{,}1787 & 0{,}2422 \end{bmatrix} \tag{4.5}$$

O processo de otimização é novamente aplicado, utilizando a matriz sensibilidade mostrada na equação 4.5. As tensões encontradas após a terceira alocação de reativos são mostradas na Tabela 4.4

Tabela 4.4 – Tensões encontradas ao resolver o fluxo de carga após alocação de reativos do sistema W&H6

Barra	Tipo	Tensão (p.u.)	ΔQ (p.u.)
1	Vθ	1,0500	-
2	PV	1,1000	-
3	PQ	0,9502	0,5017
4	PQ	1,0069	0,0000
5	PQ	0,9500	0,2081
6	PQ	0,9734	0,2247

A evolução dos estados resultantes a cada etapa da resolução é mostrada na Figura 4.1. Pode ser observado que após 3 alocações todas as tensões ficaram dentro dos limites estabelecidos.

É interessante notar que, apesar da barra 4 estar com a tensão abaixo do limite, não foi alocado nenhum reativo nesta barra. Isto pode se entendido analisando a matriz sensibilidade. No caso os termos S_{21} e S_{22}, que mostram a variação da tensão na barra 4 com uma alocação na barra e e na própria barra 4, tem quase o mesmo valor. Então, para a correção da tensão na barra 4, é mais interessante alocar reativos na barra 3, uma vez que o efeito na barra 4 será praticamente o mesmo, mas com maior efeito na barra 3.

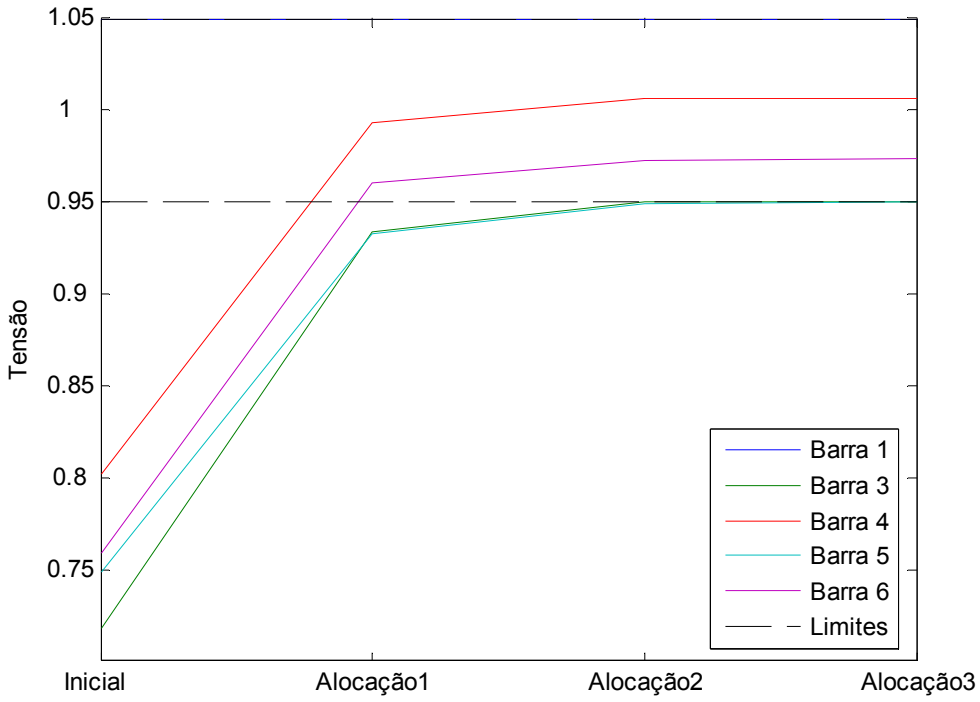

Figura 4.1 – Evolução das tensões no sistema W&H6 após sucessivas alocações de reativos.

4.3 – Resultado do sistema IEEE - 14 modificado

O sistema IEEE 14 barras está mostrado no apêndice A - 2. Ao rodar o fluxo de carga, o sistema mantinha todas as tensões dentro do limite. Para gerar uma situação com tensões abaixo do nível desejado, as cargas foram consideradas quatro vezes maiores. Para a situação de carga descrita, as tensões encontradas são mostradas na Tabela 4.5.

Tabela 4.5–Tensões encontradas ao resolver o fluxo de carga do sistema IEEE-14

Barra	Tipo	Tensão (p.u.)	Barra	Tipo	Tensão (p.u.)
1	Vθ	1,0600	8	PV	1,0900
2	PV	1,0450	9	PQ	0,7348
3	PV	1,0100	10	PQ	0,7532
4	PQ	0,7957	11	PQ	0,8923
5	PQ	0,8106	12	PQ	0,9792
6	PV	1,0700	13	PQ	0,9328
7	PQ	0,8284	14	PQ	0,7091

Podemos observar que as tensões, em geral, estão bem abaixo do limite desejado, principalmente nas barras 9 e 14. As tensões e a quantidade de reativos alocados após a utilização do procedimento de otimização são encontradas na Tabela 4.6.

Tabela 4.6–Tensões encontradas após alocação de reativos no sistema IEEE-14

Barra	Tipo	Tensão (p.u.)	ΔQ (p.u.)	Barra	Tipo	Tensão (p.u.)	ΔQ (p.u.)
1	Vθ	1,0600	-	8	PV	1,0900	-
2	PV	1,0450	-	9	PQ	0,9629	0,5205
3	PV	1,0100	-	10	PQ	0,9591	0,1337
4	PQ	0,9842	1,6628	11	PQ	0,9991	0,0000
5	PQ	0,9895	1,0987	12	PQ	1,0088	0,0000
6	PV	1,0700	-	13	PQ	0,9906	0,0000
7	PQ	0,9875	0,0000	14	PQ	0,9500	0,3682

Como esperado, as tensões ficaram dentro dos limites desejados. Todas as tensões foram corrigidas 5 alocações de reativos. Embora fosse possível a alocação em nove barras, foi alocado reativo em apenas cinco barras sendo que foram concentrados nas barras 4 e 5.

A evolução após cada alocação é mostrada na Figura 4.2:

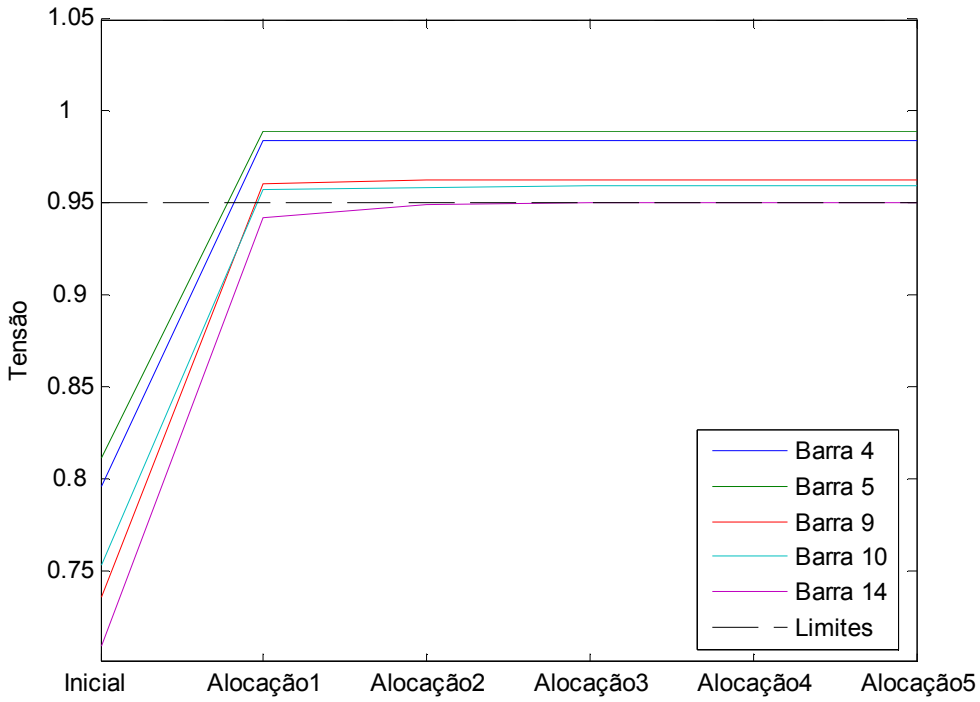

Figura 4.2 – Evolução das tensões no sistema IEEE-14 após sucessivas alocações de reativos.

4.4 – Resultado do sistema IEEE- 30 modificado

O sistema IEEE 30 barras está mostrado no apêndice A – 3. Para gerar maiores quedas de tensão, a carga foi multiplicada por 2,5. Este sistema se mostrou mais sensível ao aumento de carga, sendo que um aumento de 3 vezes já fazia o fluxo de carga não convergir.

As tensões encontradas após a resolução do problema de fluxo de carga são mostradas na Tabela 4.7.

Tabela 4.7–Tensões encontradas ao resolver o fluxo de carga do sistema IEEE-30

Barra	Tipo	Tensão (p.u.)	Barra	Tipo	Tensão (p.u.)
1	Vθ	1,0600	16	PQ	0,9255
2	PV	1,0450	17	PQ	0,9054
3	PQ	0,9466	18	PQ	0,8730
4	PQ	0,9390	19	PQ	0,8633
5	PV	1,0100	20	PQ	0,8743
6	PQ	0,9584	21	PQ	0,8794
7	PQ	0,9557	22	PQ	0,8804
8	PV	1,0100	23	PQ	0,8665
9	PQ	0,9703	24	PQ	0,8398
10	PQ	0,9175	25	PQ	0,8384
11	PV	1,0820	26	PQ	0,7816
12	PQ	0,9684	27	PQ	0,8656
13	PV	1,0710	28	PQ	0,9490
14	PQ	0,9217	29	PQ	0,7969
15	PQ	0,9053	30	PQ	0,7574

Como nos casos anteriores, as tensões nas barras PQ estão bem abaixo do patamar desejado, sendo que, nas barras 26, 29 e 30, as tensões estão abaixo de 0,8 p.u. Após a correção, as tensões encontradas e os reativos alocados são mostrados na Tabela 4.8.

Tabela 4.8–Tensões encontradas ao resolver o fluxo de carga após alocação de reativos do sistema IEEE-30

Barra	Tipo	Tensão (p.u.)	ΔQ (p.u.)	Barra	Tipo	Tensão (p.u.)	ΔQ (p.u.)
1	Vθ	1,0600	-	16	PQ	0,9708	0,0000
2	PV	1,0450	-	17	PQ	0,9613	0,0000
3	PQ	0,9581	0,0000	18	PQ	0,9507	0,0155
4	PQ	0,9527	0,0577	19	PQ	0,9503	0,1414
5	PV	1,0100	-	20	PQ	0,9546	0,0000
6	PQ	0,9703	0,0000	21	PQ	0,9516	0,0322
7	PQ	0,9629	0,0000	22	PQ	0,9550	0,0000
8	PV	1,0100	-	23	PQ	0,9501	0,0178
9	PQ	1,0032	0,0000	24	PQ	0,9514	0,1992
10	PQ	0,9769	0,0000	25	PQ	0,9687	0,0000
11	PV	1,0820	-	26	PQ	0,9500	0,0712
12	PQ	0,9681	0,0000	27	PQ	0,9875	0,0000
13	PV	1,0710	-	28	PQ	0,9694	0,0000
14	PQ	0,9681	0,0000	29	PQ	0,9562	0,0000
15	PQ	0,9626	0,0000	30	PQ	0,9520	0,1393

Como esperado, as tensões ficaram dentro dos limites desejados. A evolução das tensões após alocações sucessivas de reativos pode ser encontrada na figura 4.3. Nesse sistema só foram alocados reativos em 8 barras. Após a primeira alocação quatro barras ainda estavam fora do

limite desejado. Após a terceira alocação, as tensões não variaram muito, mas só ficaram dentro do limite desejado após 6 alocações de reativos.

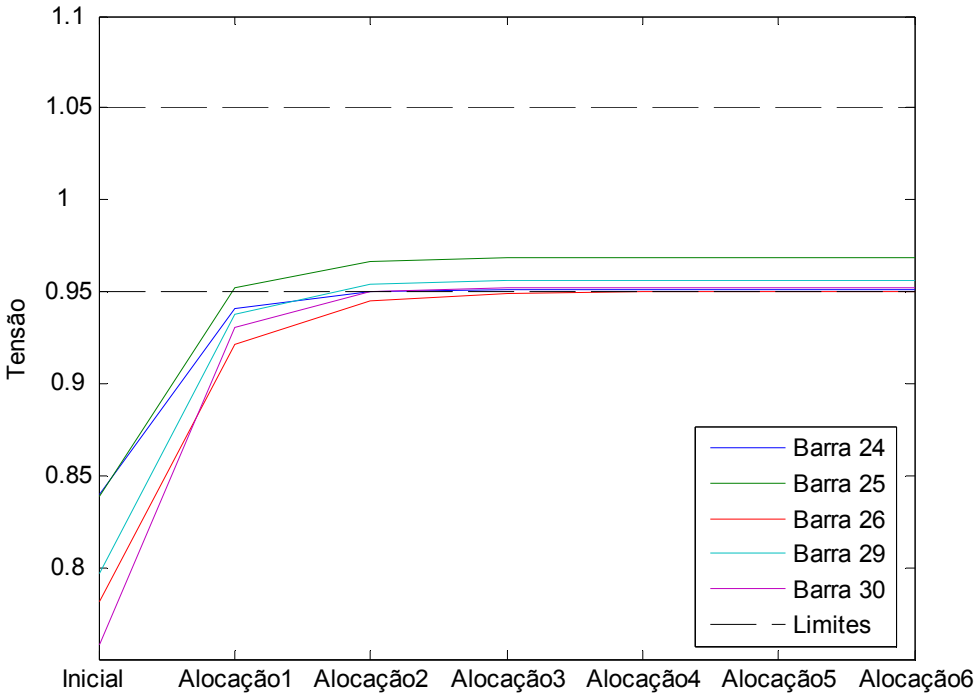

Figura 4.3 – Evolução das tensões no sistema IEEE-30 após sucessivas alocações de reativos.

CAPÍTULO 5
CONCLUSÃO

O trabalho apresentou uma metodologia de correção das magnitudes das tensões em sistemas de potência. Foi elaborado um programa que implementa a proposta do trabalho. A utilização do programa se mostrou eficaz, resultando em sistemas com as tensões dentro dos limites especificados em todos os testes realizados.

Todos os casos foram simulados utilizando-se os métodos simplex e de ponto interior. Não houve diferença de resultados utilizando o método simplex ou o de ponto interior. Isto ocorre porque o ponto de mínimo está acontecendo devido a uma restrição, então não há diferença no resultado se, ao chegar a este ponto, é percorrido um caminho interior (método de ponto interior) ou somente pelos vértices (método simplex).

Podemos perceber que nem sempre são alocados reativos nas barras com a tensão mais baixa. Por exemplo, no caso do sistema com 30 barras, a barra 29, com a terceira menor tensão, não recebeu reativos e a barra 26, com a segunda menor tensão, recebeu menos reativos que a barra 24, que tinha uma tensão maior. No teste com o sistema de 14 barras, algo parecido ocorreu. As maiores diferenças de tensão estavam nas barras 14 e 9. No entanto, foram alocados mais reativos nas barras 4 e 5.

BIBLIOGRAFIA

CHICCO, G.; GROSS, G.; TAO, S. *Allocation of the Reactive Power Support Requirements in Multitransaction* Networks - IEEE TRANSACTIONS ON POWER SYSTEMS, VOL. 17, NO. 2, MAIO 2002

DANTZIG, G.B.; ORDEN A.; WOLFE P. *Generalized Simplex Method for Minimizing a Linear from Under Linear Inequality Constraints* Pacific Journal Math., Vol. 5, pp. 183-195.

GLOVER, J. D.; SARMA, M. *Power System Analysis & Design* – Boston: PWS Publishing Company 1994.

GOH, S. H.; SAHA, T. K.; DONG, Z. Y. *Optimal Reactive Power Allocation for Power System Transfer Capability Assessment* disponível no sítio: http://ieeexplore.ieee.org/iel5/11204/36065/01709276.pdf, 2006

LIMA, R.T. *Redes neurais artificiais aplicadas no controle de tensão de sistemas elétricos de potência* – Rio de Janeiro: PUC-RIO 2007, disponível no sítio: http://www.maxwell.lambda.ele.puc-rio.br/cgi-bin/db2www/PRG_0651.D2W/SHOW?Mat=&Sys=&Nr=&Fun=&CdLinPrg=pt&CdNivTes=ME&aux=T&Cont=11488:pt

MEHROTRA, S. *On the Implementation of a Primal-Dual Interior Point Method* SIAM Journal on Optimization, Vol. 2, pp. 575-601, 1992.

MONTICELLI, A. J. *Fluxo de Carga em Redes de Energia Elétrica – São Paulo:* Editora Edgard Blücher LTDA 1983.

OLIVEIRA E. J.; FONTOURA R. M.; MARTINS N.;OLIVEIRA L. W.; PEREIRA J. L. R. *Inclusão de restrições dinâmicas no problema de planejamento de potência reativa* Sba Controle & Automação vol.19 no.1 Natal 2008

RUGGIERO, M. A. G.; LOPES, V. L. R. *Cálculo Numérico – Aspectos teóricos e Computacionais* – São Paulo: PEARSON Makron books 2004

SOUSA, T. *Estudo de planejamento de reativos em sistemas elétricos de potência* disponível no sítio: http://www.teses.usp.br/teses/disponiveis/3/3143/tde-07122006-144442/ - São Paulo USP 2003

SOUZA, A. M.; COSTA, C. E. U; COSTA, G. R. M. *Alocação ótima de reativo utilizando matriz sensibilidade* In: III Congresso Latino Americano de Geração e Transmissão de Energia Elétrica, 1997, Campos do Jordão - SP. Anais, 1997. p. 28-32.

UNIVERSITY OF WASHINGTON, College of Engineering, *14 Bus Power Flow Test Case* disponível no sítio https://www.ee.washington.edu/research/pstca/pf14/pg_tca14bus.htm/. Acesso em 13 de agosto de 2008.

UNIVERSITY OF WASHINGTON, College of Engineering, *30 Bus Power Flow Test Case* disponível no sítio https://www.ee.washington.edu/research/pstca/pf30/ieee30cdf.txt/. Acesso em 13 de agosto de 2008.

APÊNDICES

APÊNDICE A - DADOS DOS SISTEMAS UTILIZADOS

Os dados dos sistemas utilizados são mostrados a seguir. Todos os valores são dados em p.u.. O ângulo de referência é 0 nas barras Vθ.

1. Sistema WARD&HALE 6 barras:

Os dados das barras e linhas são mostrados nas Tabelas A-1 e A-2. Um diagrama do sistema é mostrado na figura A-1. Este sistema foi retirado de SOUZA (2003).

Tabela A-1–Dados das barras do sistema W&H6

Barra	Tipo	V	P Gerado	Q Gerado	P carga	Q Carga
1	Vθ	1,050	0,00	0,00	0,000	0,000
2	PV	1,100	0,50	0,00	0,000	0,000
3	PQ	-	0,00	0,00	0,825	0,195
4	PQ	-	0,00	0,00	0,000	0,000
5	PQ	-	0,00	0,00	0,450	0,270
6	PQ	-	0,00	0,00	0,750	0,075

Tabela A-2–Dados das linhas do sistema W&H6

From	To	Resistência	Reatância	Tap
1	6	0,123	0,518	-
1	4	0,080	0,370	-
4	6	0,097	0,407	-
5	6	0,000	0,300	1,025
5	2	0,282	0,640	-
2	3	0,723	1,050	-
3	4	0,000	0,133	1,1

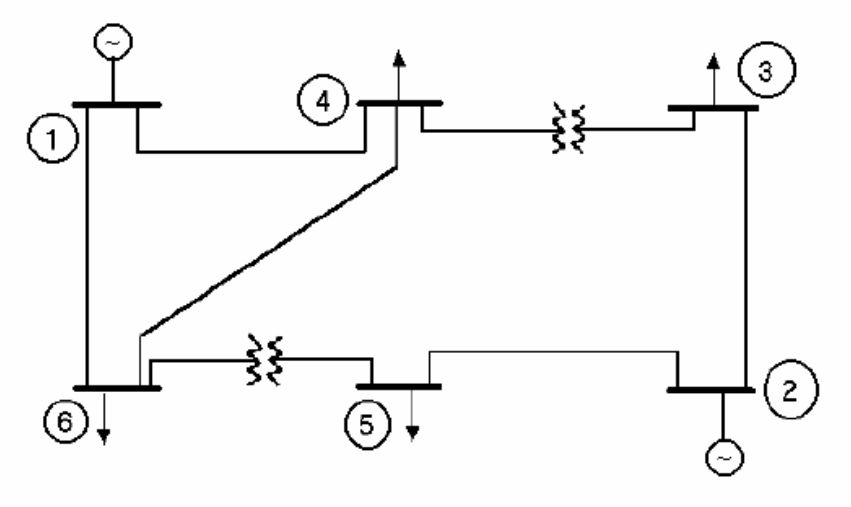

Figura A-1 – Sistema Ward&Hale de 6 barras

2. Sistema IEEE-14:

Os dados das barras e linhas são mostrados nas Tabelas A-3 e A-4. Um diagrama do sistema é mostrado na figura A-2. Os dados deste sistema foram retirados do sítio da Universidade de Washington citado na bibliografia.

Tabela A-3–Dados das barras do sistema IEEE-14

Barra	Tipo	V	P Gerado	Q Gerado	P carga	Q Carga	B^{sh}
1	PV	1,060	2,320	0,000	0,000	0,000	-
2	Vθ	1,045	0,400	-0,424	0,868	0,508	-
3	PV	1,010	0,000	0,000	3,768	0,760	-
4	PQ	-	0,000	0,000	1,912	0,000	-
5	PQ	-	0,000	0,000	0,304	0,064	-
6	PV	1,070	0,000	0,000	0,448	0,300	-
7	PQ	-	0,000	0,000	0,000	0,000	-
8	PV	1,090	0,000	0,000	0,000	0,000	-
9	PQ	-	0,000	0,000	1,180	0,664	0,190
10	PQ	-	0,000	0,000	0,360	0,232	-
11	PQ	-	0,000	0,000	0,140	0,072	-
12	PQ	-	0,000	0,000	0,244	0,064	-
13	PQ	-	0,000	0,000	0,540	0,232	-
14	PQ	-	0,000	0,000	0,596	0,200	-

Tabela A-4–Dados das linhas do sistema IEEE-14

From	To	Resistência	Reatância	Line Charging	Tap
1	2	0,01938	0,05917	0,0528	-
1	5	0,05403	0,22304	0,0492	-
2	3	0,04699	0,19797	0,0438	-
2	4	0,05811	0,17632	0,0374	-
2	5	0,05695	0,17388	0,034	-
3	4	0,06701	0,17103	0,0346	-
4	5	0,01335	0,04211	0,0128	-
4	7	0,00000	0,20912	0,0000	0,978
4	9	0,00000	0,55618	0,0000	0,969
5	6	0,00000	0,25202	0,0000	0,932
6	11	0,09498	0,19890	0,0000	-
6	12	0,12291	0,25581	0,0000	-
6	13	0,06615	0,13027	0,0000	-
7	8	0,00000	0,17615	0,0000	-
7	9	0,00000	0,11001	0,0000	-
9	10	0,03181	0,08450	0,0000	-
9	14	0,12711	0,27038	0,0000	-
10	11	0,08205	0,19207	0,0000	-
12	13	0,22092	0,19988	0,0000	-
13	14	0,17093	0,34802	0,0000	-

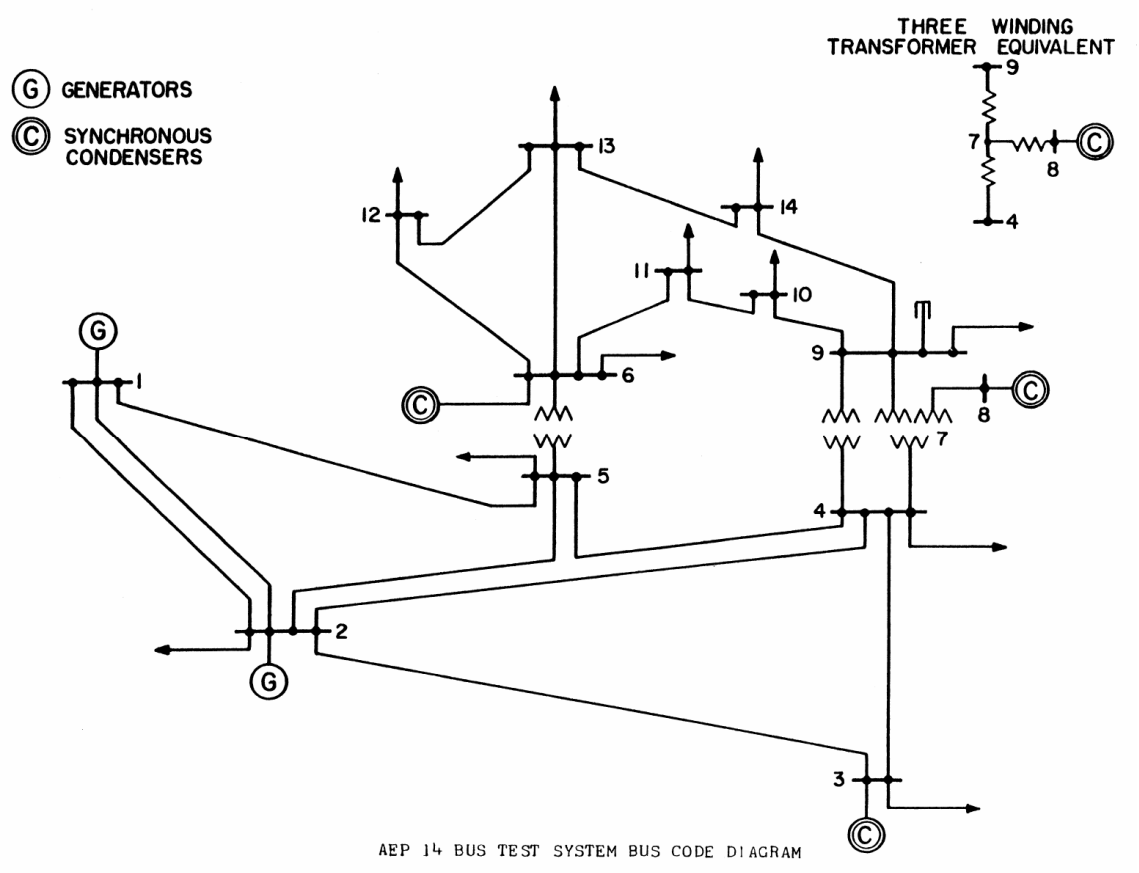

Figura A-2 – Sistema IEEE-14

3. Sistema IEEE-30:

Os dados das barras e linhas são mostrados nas Tabelas A-5 e A-6. Um diagrama do sistema é mostrado na figura A-3. Os dados deste sistema foram retirados do sítio da Universidade de Washington citado na bibliografia.

Tabela A-5–Dados das barras do sistema IEEE-30

Barra	Tipo	V	P Gerado	Q Gerado	P carga	Q Carga	B^{sh}
1	Vθ	1,060	2.602	-0.161	0.0000	0.0000	-
2	PV	1,045	0.400	0.500	0.5425	0.3175	-
3	PQ	-	0.000	0.000	0.0600	0.0300	-
4	PQ	-	0.000	0.000	0.1900	0.0400	-
5	PV	1,010	0.000	0.370	2.3550	0.4750	-
6	PQ	-	0.000	0.000	0.0000	0.0000	-
7	PQ	-	0.000	0.000	0.5700	0.2725	-
8	PV	1,010	0.000	0.373	0.7500	0.7500	-
9	PQ	-	0.000	0.000	0.0000	0.0000	-
10	PQ	-	0.000	0.000	0.1450	0.0500	0.190
11	PV	1,082	0.000	0.162	0.0000	0.0000	-
12	PQ	-	0.000	0.000	0.2800	0.1875	-
13	PV	1,071	0.000	0.106	0.0000	0.0000	-
14	PQ	-	0.000	0.000	0.1550	0.0400	-
15	PQ	-	0.000	0.000	0.2050	0.0625	-
16	PQ	-	0.000	0.000	0.0875	0.0450	-
17	PQ	-	0.000	0.000	0.2250	0.1450	-
18	PQ	-	0.000	0.000	0.0800	0.0225	-
19	PQ	-	0.000	0.000	0.2375	0.0850	-
20	PQ	-	0.000	0.000	0.0550	0.0175	-
21	PQ	-	0.000	0.000	0.4375	0.2800	-
22	PQ	-	0.000	0.000	0.0000	0.0000	-
23	PQ	-	0.000	0.000	0.0800	0.0400	-
24	PQ	-	0.000	0.000	0.2175	0.1675	0.043
25	PQ	-	0.000	0.000	0.0000	0.0000	-
26	PQ	-	0.000	0.000	0.0875	0.0575	-
27	PQ	-	0.000	0.000	0.0000	0.0000	-
28	PQ	-	0.000	0.000	0.0000	0.0000	-
29	PQ	-	0.000	0.000	0.0600	0.0225	-
30	PQ	-	0.000	0.000	0.2650	0.0475	-

Tabela A-6–Dados das linhas do sistema IEEE-30

From	To	Resistência	Reatância	Line Charging	Tap
1	2	0.0192	0.0575	0.1056	-
1	3	0.0452	0.1652	0.0816	-
2	4	0.0570	0.1737	0.0736	-
3	4	0.0132	0.0379	0.0168	-
2	5	0.0472	0.1983	0.0836	-
2	6	0.0581	0.1763	0.0748	-
4	6	0.0119	0.0414	0.0180	-
5	7	0.0460	0.1160	0.0408	-
6	7	0.0267	0.0820	0.0340	-
6	8	0.0120	0.0420	0.0180	-
6	9	0.0000	0.2080	0.0000	0.978
6	10	0.0000	0.5560	0.0000	0.969
9	11	0.0000	0.2080	0.0000	-
9	10	0.0000	0.1100	0.0000	-
4	12	0.0000	0.2560	0.0000	0.932
12	13	0.0000	0.1400	0.0000	-
12	14	0.1231	0.2559	0.0000	-
12	15	0.0662	0.1304	0.0000	-
12	16	0.0945	0.1987	0.0000	-
14	15	0.2210	0.1997	0.0000	-
16	17	0.0524	0.1923	0.0000	-
15	18	0.1073	0.2185	0.0000	-
18	19	0.0639	0.1292	0.0000	-
19	20	0.0340	0.0680	0.0000	-
10	20	0.0936	0.2090	0.0000	-
10	17	0.0324	0.0845	0.0000	-
10	21	0.0348	0.0749	0.0000	-
10	22	0.0727	0.1499	0.0000	-
21	22	0.0116	0.0236	0.0000	-
15	23	0.1000	0.2020	0.0000	-
22	24	0.1150	0.1790	0.0000	-
23	24	0.1320	0.2700	0.0000	-
24	25	0.1885	0.3292	0.0000	-
25	26	0.2544	0.3800	0.0000	-
25	27	0.1093	0.2087	0.0000	-
28	27	0.0000	0.3960	0.0000	0.968
27	29	0.2198	0.4153	0.0000	-
27	30	0.3202	0.6027	0.0000	-
29	30	0.2399	0.4533	0.0000	-
8	28	0.0636	0.2000	0.0856	-
6	28	0.0169	0.0599	0.0260	-

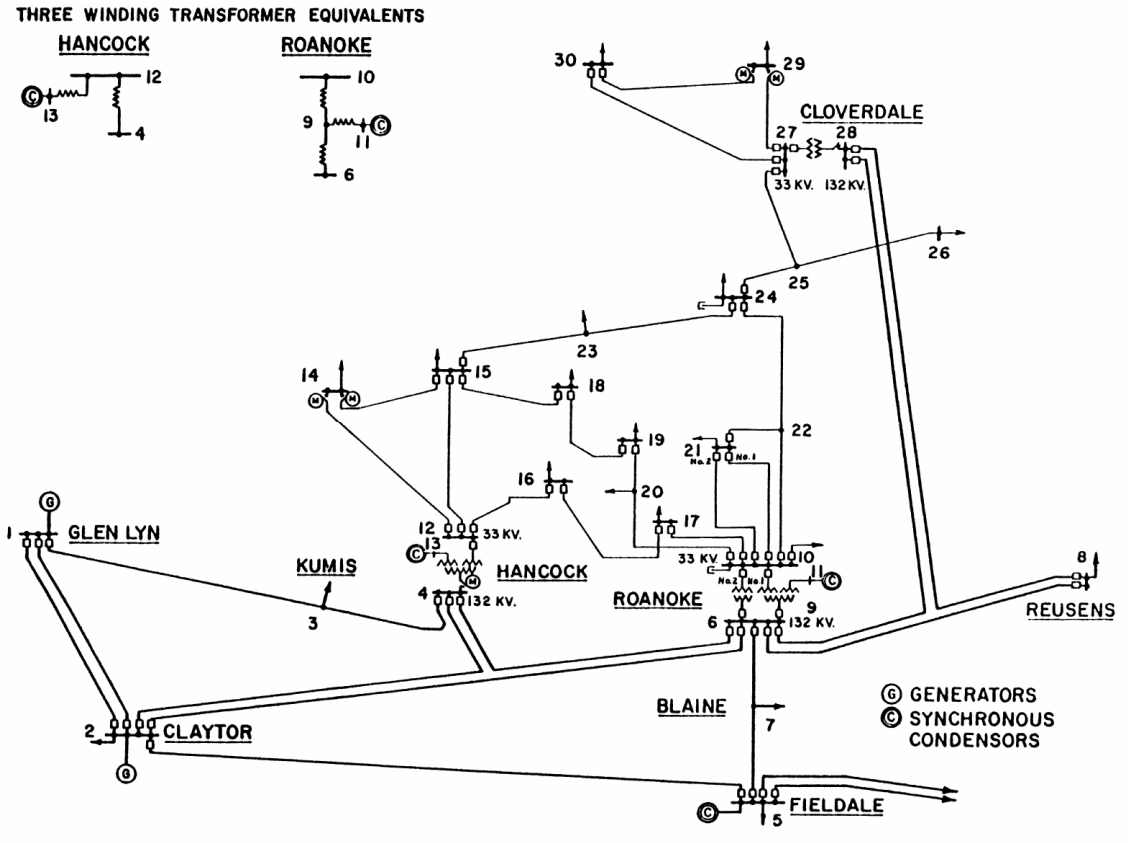

Figura A-3 –Sistema IEEE-30

APÊNDICE B - CÓDIGO DO PROGRAMA

Os comentários são precedidos de %.

Arquivo de entrada:

```
clear
clc;
global A nb nl rl xl vm va
%declara variaveis globais para serem lidas por funções auxiliares
% 1. vetor de numeros das barras de origem das linhas (vetor ifr):
ifr = [1 1 4 5 5 2 3]';
% 2. vetor de numeros das barras de destino das linhas (vetor ito):
ito = [6 4 6 6 2 3 4]';
% 3. vetor de Resistencias das Linhas:
rl = [0.123 0.080 0.097 0.000 0.282 0.723 0.000]';
% 4. vetor de Reactância das Linhas:
xl = [0.518 0.370 0.407 0.300 0.640 1.050 0.133]';
% 5. vetor de Susceptâncias shunt das Linhas
% (lembre-se de introduzir o valor b/2!!!):
bsh = [0.000 0.000 0.000 0.000 0.000 0.000 0.000]';
% 6. vetor de valor dos taps nos transformadores associados a cada linha:
transf = [1 1 1 1 1 1 1]';
tap = [1 1 1 1.025 1 1 1.1]';
rel_traf = transf.*tap;
% 7. vetor de Compensador shunt em cada barra
% - se o compensador e capacitivo, introduza o valor da susceptancia com sinal positivo
% - se o compensador e inductivo, introduza o valor da susceptancia com sinal negativo
bshb = [0.000 0.000 0.000 0.000 0.000 0.000]';
% 8. Calculo do numero de barras da rede
nb=max(max([ito,ifr]));
% 9. Calculo do numero de linhas da rede
nl=size(ifr,1);
% DADOS DO SISTEMA DE POTENCIA:
% 10. Numero da Barra de referencia (V@):
referencia = [1]';            %Não é necessário
folga = [1]';                 %Não é necessário
% 11. barras com tensão fixa (PV e V@):
iv = [1 2]';
% 12. barras PQ:
iq = [3 4 5 6]';
% 13. Numero das barras com P fixo (PQ e PV):
% Não esqueca de introduzir os valores a seguir na mesma ordem em que introduziu os numeros
das
% barras!!\n');
ip = [2 3 4 5 6]';
% Vetor com todas as barras menos a barra de referencia
```

```
ir = [2 3 4 5 6]';            %Não é necessário
% 14. Geração e demanda ATIVA especificada para cada barra:
PG = [0 0.5 0 0 0 0]'; % Valor de Geração em cada barra (dimensão nb x nb)
PD = -1.5*[0 0 0.55 0 0.3 0.5]'; % Valor de Demanda em
PN = PG + PD; % Calculo do vetor da Potencia liquida injetada em cada barra (dimensão nb x nb)
% 15. Valor de geraçao e demanda REATIVA para cada barra:
QG = [0 0 0 0 0 0]'; % Valor de Geração em cada barra (dimensão nb x nb)
QD = -1.5*[0 0 0.13 0 0.18 0.05]'; % Valor de Demanda em cada barra (dimensão nb x nb)
QN = QG + QD; % Calculo do vetor da Potencia liquida injetada em cada barra (dimensão nb x nb)
% 16. Valor de V para cada barra PV e V@:
vs = [1.05 1.1]';
maloca; % Executa a rotina de cálculo
```

Base do programa (maloca.m):

```
% Inicializa algumas variáveis internas
% A primeiras podem ser alteradas para a execução do programa de acordo com
% a necessidade
Vmin = 0.95;            % Tensão mínima desejada
Vmax = 1.05;            % Tensão máxima desejada
epslon = 0.0000000001;  % Tolerância para erro máximo no fluxo de carga
errotolerado = 0.00001; % Tolerância para alocar ou não reativos para corrigir tensão
simplex = 1;            % Se simplex=1 executa o metodo simplex se não utiliza MPI
plota = 1;              % Se plota = 1 mostra a evoluçãodas tensões a cada iteração
quant = 5;              % Define quantas barras serão mostradas no plot, será mostrado o
                        % progresso das barras com as menores tensões sem nenhum
                        % reativo alocado, se plota = 1 e quant = 0 são
                        % mostradas todas barras


% Nessas variaveis internas não mexer. No programa elas são testadas e
% alteradas se necessário, os valores iniciais para o funcionamento correto
% devem ser todos 0
flag=0;
deltaq=0;
cont=0;


NPQ=max(size(iq));
cont=cont+1
while(flag==0)

    madmitancia

    % Calculo do subsistema 1: Tensoes nas barras e Potencias injetadas
    % Define o valor de tensao = 1 pu para todas as barras do sistema
    vm = ones(nb,1);
```

```
% Substitui a tensao para as barras de tensao fixa
vm(iv) = vs;
% Define o valor do angulo = 0 para todas as barras do sistema
va = zeros(nb,1);
% Monta o vetor v (estado do sistema)
v = vm.*exp(j*va);
% Calcula as potencias complexas
s=v.*conj(Y*v);
% Calcula o erro entre os valores especificados e os valores calculados
% a partir do estado atual
m=[PN(ip)-real(s(ip)); QN(iq)-imag(s(iq))];
% Identifica o erro maximo cometido
maxm=max(abs(m));
n=1;
Iteracao=[];
% Compara o erro com o erro maximo admitido
while maxm > epslon
    %calcula a matriz jacobiana a partir da rotina jacobiano
    jacobiano;
    % Calcula o vetor de correcao
    dx = J \ m;
    % Aplica correcao aos angulos
    va(ip) = va(ip) + dx(1:nb-1);
    % Aplica correcao aos modulos
    vm(iq) = vm(iq) + dx(nb:length(dx));
    % Repete basicamente o que foi feito anteriormente de montar o
    % vetor de tensões, calcular potência e identificar o maior erro
    v = vm.*exp(j*va);
    s = v.*conj(Y*v);
    m = [PN(ip)-real(s(ip)); QN(iq)-imag(s(iq))];
    % b.16 Identifica o erro maximo cometido como anteriormente
    maxm = max(abs(m));
    Iteracao(n)=maxm;
    if(n>100)
            error('Sistema não convergiu com 100 iterações');
            break;
    end
    n=n+1;
    %acaba o loop while agora ele compara o novo maxm e se for
    %necessário executa o loop novamente, se não for necessário os
    %erros encontrados são menores que o epsilon e podemos partir para
    %a resolução do subsistema 2
end
% Guarda as tensões do sistema quando não há ainda alocação de
% reativos.

vmantigo(cont,:)=vm;
cont=cont+1;
```

```
% Calculo das tensões e correntes nas linhas
Vkm=v(ifr)-v(ito);
Vmk=v(ito)-v(ifr);

Ikm=yl.*Vkm + bsh*j.*v(ifr);
Imk=yl.*Vmk + bsh*j.*v(ito);

% Calculo das potencias de transferencia pelas linhas:
Skm=v(ifr).*conj(Ikm);
Smk=v(ito).*conj(Imk);

% Calculo das perdas totais do sistema:

PerdasTotais=sum(Skm+Smk);

% Atualiza os valores gerados em cada barra
PG=real(s)-PD;
QG=imag(s)-QD;

% Verifica se é necessário alocar reativos para corrigir a tensão
corrige=0;

r=1;
while(r<=size(iq,1))
    if(vm(iq(r))>=Vmax+errotolerado||vm(iq(r))<=Vmin-errotolerado)
        corrige=1;
        break
    end
    r=r+1;
end

% Se nao for necessário imprime um relatório sobre o estado atual do
% sistema e as tensões sem a alocação e quanto foi alocado de reativo
% em cada barra

if(corrige==0)
    if plota==1

        [r,c]=size(vmantigo);
        [b,indices]=sort(vmantigo(1,1:c));
        if quant==0
            b=1:nb;
        else
            if quant<=nb
                b=indices(1:quant);
                b=sort(b);
            else
                b=1:nb;
            end
        end
        vmantigo(:,c+1)=ones(r,1)*Vmin;
```

```
    cont=cont+1;
    vmantigo(:,c+2)=ones(r,1)*Vmax;
    clf
    for i=b
        plot(0:r-1,vmantigo(:,i))
        hold all
    end
    plot(0:r-1,vmantigo(:,c+1),'k--')
    plot(0:r-1,vmantigo(:,c+2),'k--')
    ylabel('Tensão')
    set(gca,'XTick',0:r-1)
    lab=get(gca,'YTickLabel');
    f=get(gca,'YTick');
    f(1)=[];
    [d,c]=size(lab);
    lab=strvcat('',lab(2:d,:));
    set(gca,'YTick',f,'YTickLabel',lab)
    lab='Inicial';
    i=1;
    while i<r
        lab=strvcat(lab,['Alocação' num2str(i)]);
        i=i+1;
    end
    set(gca,'XTickLabel',lab)
    clear c lab d
    c=[];
    for i=1:size(b,2)
    c=strvcat(c,['Barra ' num2str(b(i))]);
    end
    c=strvcat(c,'Limites');
    legend(c,'Location','SouthEast')
end
fprintf('\nAs tensões estão dentro dos valores estabelecidos\n')
flag=1;

%=====================================================
% printing output
%
fprintf('\n')
fprintf('\n Valores obtidos em p.u.: \n ')
fprintf('\n  Barra \tPg \t\t\tQg \t\tPd \t\t\tQd \t\t\tV \t \tÂngulo\n')
fprintf('_____________________________________________________________________________\n')
for a=1:nb
fprintf('\n %4d %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f',...
a,PG(a),QG(a),PD(a),QD(a),vm(a),va(a))
end


fprintf('\n\n \tfrom \tto \t\tPflowij \t\tPflowji \tPerdas Ativas  Reativas\n')
fprintf('_____________________________________________________________________________\n')
for a=1:nl
```

```
        fprintf('\n        \t%4d        \t%4d        \t%10.4f        \t%10.4f        \t%10.4f
\t%10.4f',ifr(a),ito(a),pflowij(a),pflowji(a),real(Skm(a)+Smk(a)),imag(Skm(a)+Smk(a)))
        end
        fprintf('\n')

        r = zeros(nb,1);
        r(iq)=deltaq;
        fprintf('\n\n Tensão sem alocação \t Tensão Corrigida \t\tQ alocado\n')
        fprintf('_____________________________________________________________\n')
        for a=1:nb
        fprintf('\n %10.4f \t\t\t   %10.4f \t\t %10.4f',vmantigo(1,a), vm(a), r(a) )
        end
        fprintf('\n')



    % Se a tensão estiver fora do limite desejado executa a rotina de
    % calculo de reativos a alocar
    else
        fprintf('\ntensões fora dos limites é necessário injetar reativos para ajustá-las\n')

        %Monta a submatriz L
        r=1;
        while(r<=nb)
            s=1;
            while(s<=nb)
                if r==s
    %               Lkk ou Lrr
                    num=r;
                    t=[adjacentes(num, ifr, ito, nl), num];
                    a=1;
                    u=size(t,2);
                    L(r,s)=-1*vm(num)*B(num,num);
                    while a<=u
                        L(r,s)=L(r,s)+vm(t(a))*(G(num,t(a))*sin(va(num)-va(t(a)))-
B(num,t(a))*cos(va(num)-va(t(a))));
                        a=a+1;
                    end
                else
    %               Lkm ou Lrs
                    L(r,s)=vm(r)*(G(r,s)*sin(va(r)-va(s))-B(r,s)*cos(va(r)-va(s)));
                end
                s=s+1;
            end
            r=r+1;
        end

        % Monta a matriz sensibilidade
        S=inv(L(iq,iq));

        % Agora resta criar a função que é a soma das injeções e as inequações
```

```
        % Como a função a ser minimizada é a simples soma dos reativos
        % alocados os coeficientes são todos 1
        f=ones(size(iq));

        % Os coeficiente das inequações são dados pela eq. 2.9

        A = [-1.*S; S; -1*eye(NPQ)];
        b = [vm(iq)-Vmin; Vmax-vm(iq); zeros(size(iq))];


        if(simplex==1)
            options = optimset('LargeScale', 'off', 'Simplex', 'on');
        else
            options = optimset('LargeScale', 'on', 'Simplex', 'off');
        end

        tic
            xmin = linprog(f,A,b,[],[],[],[],[],options);
        toc
        deltaq=deltaq+xmin;
        QN(iq) = QN(iq) + xmin;
    end
end
```

Funções/Rotinas auxiliares:

- Cálculo da matriz admitância (madmitancia.m):

```
%calcula a matriz admitancia a partir da eq. 1.37 do monticelli
% e a incidencia para compatibilidade do pflowij e ji


Id = eye(nl);
if nl==1
Afrom=[1 0]';
Ato=[0 1]';
A=Afrom-Ato;
elseif nl>1
A = Id(1:nb,ifr)-Id(1:nb,ito);
end
```

```
zl = rl+j*xl;

yl = 1./zl; %y=1/z termo a termo
g=real(yl);
b=imag(yl);

a=1;
r=1;
%elementos fora da diagonal principal
while(r<=nl)
    numl=ifr(r);
    numc=ito(r);
    Y(numl,numc)=-1*rel_traf(r)*yl(r);
    Y(numc,numl)=-1*rel_traf(r)*yl(r);
    r=r+1;
end

%elementos da diagonal
r=1;
while(r<=nb)
    t=adjacentes(r, ifr, ito, nl);
    % t tem agora os as posições das linhas que serão utilizadas no
    % somatorio do termo Ykk, linhas adjcentes a barra relativa a diagonal
    % a ser formada
    a=1;
    u=size(t,2);
    Y(r,r)=j*bshb(r);
    % começa colocando o termo que não está no somatório
    while a<=u
        [s,aij]=peganumlinha(r,t(a),ifr,ito,nl,rel_traf);
        %agora peganumlinha retorna tb o valor de aij a ser utilizado
        Y(r,r)=Y(r,r)+j*bsh(s)+aij^2*yl(s);
        a=a+1;
    end
    r=r+1;
end

G = real(Y);
B = imag(Y);
```

- Cálculo da matriz jacobiana (jacobiano.m):

```
% Calcula matriz jacobiana
% ip são as barras PV e PQ
% iq são as barras PQ
% vm vetor da magnitude das tensões
% va vetor dos ângulos das tensões
```

```
H=[];
M=[];
N=[];
L=[];
% MATRIZ H (PV + PQ)x(PV + PQ)
r=1;
    while(r<=size(ip,1))
        s=1;
        while(s<=size(ip,1))
            if ip(r)==ip(s)
                t=[adjacentes(ip(r), ifr, ito, nl), ip(r)];
                a=1;
                u=size(t,2);
                H(r,s)=-1*vm(ip(r))^2*B(ip(r),ip(r));
                while a<=u
                                        H(r,s)=H(r,s)-
vm(ip(r))*vm(t(a))*(G(ip(r),t(a))*sin(va(ip(r))-va(t(a)))-B(ip(r),t(a))*cos(va(ip(r))-
va(t(a))));
                    a=a+1;
                end
            else
                H(r,s)=vm(ip(r))*vm(ip(s))*(G(ip(r),ip(s))*sin(va(ip(r))-va(ip(s)))-
B(ip(r),ip(s))*cos(va(ip(r))-va(ip(s))));
            end
            s=s+1;
        end
        r=r+1;
    end

% MATRIZ N (PV + PQ)x(PQ)
t=[];
r=1;
    while(r<=size(ip,1))
        s=1;
        while(s<=size(iq,1))
            if ip(r)==iq(s)
                num=ip(r);
                t=[adjacentes(ip(r), ifr, ito, nl), ip(r)];
                a=1;
                u=size(t,2);
                N(r,s)=vm(num)*G(num,num);
                while a<=u
                    N(r,s)=N(r,s)+vm(t(a))*(G(num,t(a))*cos(va(num)-
va(t(a)))+B(num,t(a))*sin(va(num)-va(t(a))));
                    a=a+1;
                end
            else
                N(r,s)=vm(ip(r))*(G(ip(r),iq(s))*cos(va(ip(r))-
va(iq(s)))+B(ip(r),iq(s))*sin(va(ip(r))-va(iq(s))));
            end
            s=s+1;
```

```
        end
        r=r+1;
    end

% MATRIZ M (PQ)x(PV + PQ)
t=[];
r=1;
    while(r<=size(iq,1))
        s=1;
        while(s<=size(ip,1))
            if iq(r)==ip(s)
                num=iq(r);
                t=[adjacentes(num, ifr, ito, nl), num];
                a=1;
                u=size(t,2);
                M(r,s)=-1*vm(num)^2*G(num,num);
                while a<=u
                    M(r,s)=M(r,s)+vm(num)*vm(t(a))*(G(num,t(a))*cos(va(num)-
va(t(a)))+B(num,t(a))*sin(va(num)-va(t(a))));
                    a=a+1;
                end
            else
                M(r,s)=-1*vm(iq(r))*vm(ip(s))*(G(iq(r),ip(s))*cos(va(iq(r))-
va(ip(s)))+B(iq(r),ip(s))*sin(va(iq(r))-va(ip(s))));
            end
            s=s+1;
        end
        r=r+1;
    end

% MATRIZ L (PQ)x(PQ)
r=1;
    while(r<=size(iq,1))
        s=1;
        while(s<=size(iq,1))
            if iq(r)==iq(s)
                num=iq(r);
                t=[adjacentes(num, ifr, ito, nl), num];
                a=1;
                u=size(t,2);
                L(r,s)=-1*vm(num)*B(num,num);
                while a<=u
                    L(r,s)=L(r,s)+vm(t(a))*(G(num,t(a))*sin(va(num)-va(t(a)))-
B(num,t(a))*cos(va(num)-va(t(a))));
                    a=a+1;
                end
            else
                L(r,s)=vm(iq(r))*(G(iq(r),iq(s))*sin(va(iq(r))-va(iq(s)))-
B(iq(r),iq(s))*cos(va(iq(r))-va(iq(s))));
            end
            s=s+1;
```

```
        end
        r=r+1;
    end

J=[H N; M L];
```

- Função adjacentes:

```
function a=adjacentes(num, ifr, ito, nl)
% retorna os numeros das barras adjacentes(que são vizinhas, existe linha
% entre elas) a barra num utilizando os vetores ifr e ito e o
% numero de linhas (nl)
u=1;
t=1;
while t<=nl
    if(ifr(t)==num) % testa se num é um ito se for pega i ifr correspodente
        a(u)=ito(t);
        u=u+1;
    elseif(ito(t)==num) % testa se num é um ifr se for pega i ito correspodente
        a(u)=ifr(t);
        u=u+1;
    end
    t=t+1;
end
```

- Função pflowji:

```
function saida=pflowji(pos)
% Calcula pflow da barra j para barra i pela linha na posição pos nos
% vetores ifr ito r x e bsh primeiro tem de identificar os valores de i e j
% pela posição passada, a partir da matriz de admitancia A
global A nb vm va rl xl
coltemp=A(:,pos); %coltemp pega a coluna da matriz incidencia relativa a linha pos
for kk=1:nb
    %percorre a coluna procurando os valores 1 e -1, to e from
    switch coltemp(kk)
        case {1}
            j=kk;
        case {-1};
            i=kk;
    end
end

gkm=rl(pos)/(rl(pos)*rl(pos)+xl(pos)*xl(pos));
bkm=-xl(pos)/(rl(pos)*rl(pos)+xl(pos)*xl(pos));
saida=gkm*(vm(i)^2-vm(i)*vm(j)*cos(va(i)-va(j)))-bkm*vm(i)*vm(j)*sin(va(i)-va(j));
```

- Função pflowij:

```
function saida=pflowij(pos)
```

```
% Calcula pflow da barra i para barra j pela linha na posição pos nos
% vetores ifr ito r x e bsh primeiro tem de identificar os valores de i e j
% pela posição passada, a partir da matriz de admitancia A
global A nb vm va rl xl
coltemp=A(:,pos); %coltemp pega a coluna da matriz incidencia relativa a linha pos
for kk=1:nb
    %percorre a coluna procurando os valores 1 e -1, to e from
    switch coltemp(kk)
        case {1}
            i=kk;
        case {-1};
            j=kk;
    end
end

gkm=rl(pos)/(rl(pos)*rl(pos)+xl(pos)*xl(pos));
bkm=-xl(pos)/(rl(pos)*rl(pos)+xl(pos)*xl(pos));
saida=gkm*(vm(i)^2-vm(i)*vm(j)*cos(va(i)-va(j)))-bkm*vm(i)*vm(j)*sin(va(i)-va(j));
```

- Função peganumlinha:

```
function [saida,aij]=peganumlinha(k,m,ifr,ito,nl,rel_traf)
% essa função da como saida qual linha vai da barra k para barra m
% nao deve funcionar se tiver duas linhas

a=1;
while(a<=nl)
    %testa se ifr-ito e km
    if(ifr(a)==k)
        if(ito(a)==m)
            saida=a;
            aij=rel_traf(a);
%             aij tem a relação de transformação da barra t(a), que é a
%             relação de
            break;
        end
    end
    %testa se ifr-ito e mk
    if(ito(a)==k)
       if(ifr(a)==m)
           saida=a;
           aij=1;
           break;
       end
    end
    a=a+1;
end
```