UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Pedro Henrique Neves dos Santos

IT306W - Fluxo de carga completo

Campinas
2020

Resumo

Há muitas décadas, a energia elétrica desempenha um papel vital na nossa sociedade. Desde necessidades mais simples, como iluminação pública, até necessidades mais complexas, como equipamentos médicos e industrias inteiras, só são factíveis pois temos energia elétrica. E ela é estável.
Para que isso seja possível, é necessário alguma metodologia que nos forneça o estado atual do sistema para que se possa, então, atuar apropriadamente.
Neste trabalho faremos um estudo sobre sobre fluxo de potência, uma metodologia que nos fornece tensões nodais e angulos de fase.

Os códigos fonte desse trabalho bem como histórico de modificação, podem ser encontrados no endereço: https://github.com/phneves/ElectricPowerSystemsAnalysisTools

Sumário

Capítulo 1

Introdução e teoria

A ferramenta de análise de sistemas de energia elétrica aqui discutida será a Análise de Fluxo de Potência. Essa análise nos fornecerá um metodo de cálculo de importantes grandezas de todo o sistema, a partir de algumas grandezas conhecidas, ou seja, que podem ser medidas. Esta é uma ferramenta aplicada à um sistema estático. (MONTICELLI, 1983)

1.1 Modelagem

Toda a rede será composta por ramos e barras, onde ramos são a representações das linhas de transmissão e transformadores. As barras são nós do sistema e nesses pontos de interesse, pode-se medir fisicamente algumas grandezas, a depender do tipo de barra. São definidas quatro variáveis à para cada k-ésima barra, correspondentes à tensão e à injeção de potência nesta barra, V_k, magnitude da tensão nodal, θ_k angulo da tensão nodal, P_k injeção líquida de potência ativa e Q_k injeção líquida de potência reativa. As barras são divididas em categorias como na tabela 1.1 de acordo com as grandezas conhecidas e suas incognitas.

Tabela 1.1: Categoria das barras de acordo com variáveis conhecidas

Tipo	Dados	Incógnitas	Caracteristicas
PQ	P_k,Q_k	V_k,θ_k	Barra de carga
PV	P_k,V_k	Q_k,θ_k	Barra de geração
Referência (Slack)	V_k,θ_k	P_k,Q_k	Barra de geração em grandes fornecedoras

1.2 Formulação do problema básico

1.2.1 Injeção de potências

Seja uma rede de energia genérica que contém um número de barras *(NB)* arbitrário. Para cada barra, é possível escrever duas equações de injeções de potência, como em 1.1 e 1.2. Elas são obtidas ao aplicar Lei de Kirchhoff das correntes em todas as NB barras do sistema.

$$P_k = V_k \sum_{m \in \kappa} V_m (G_{km} cos\theta_{km} + B_{km} sen\theta_{km}) \tag{1.1}$$

$$Q_k = V_k \sum_{m \in \kappa} V_m (G_{km} sen\theta_{km} - B_{km} cos\theta_{km}) \tag{1.2}$$

Portanto, em um sistema com *NB* barras, é possivel obter $2.NB$ equações.

Neste problema, utilizaremos como variáveis de estado, ou seja, nosso conjunto de incognitas que descreverão o sistema, as variáveis V e θ, representados em 1.3 e 1.4. Com esses valores resolvidos, é possível calcular todas as injeções de potência em todas as barras.

$$V^S = \begin{bmatrix} V_1^S & V_2^S & V_3^S & \dots & V_{NB}^S \end{bmatrix}^T \tag{1.3}$$

$$\theta^S = \begin{bmatrix} \theta_1^S & \theta_2^S & \theta_3^S & \dots & \theta_1^S \end{bmatrix}^T \tag{1.4}$$

Quando 1.3 e 1.4 forem resolvidas, será possivel aplicar em 1.1 e 1.2 para se obter 1.5 e 1.6, que será as injeções de potência para o estado atual da rede.

$$P_k = V_k^S \sum_{m \in \kappa} V_m^S (G_{km} cos\theta_{km}^S + B_{km} sen\theta_{km}^S) \tag{1.5}$$

$$Q_k = V_k^S \sum_{m \in \kappa} V_m^S (G_{km} sen\theta_{km}^S - B_{km} cos\theta_{km}^S) \tag{1.6}$$

Problema consiste em obter o estado (V^S, θ^S).

Com um simples algebrismo matemático, as equações 1.1 e 1.2 serão representas aqui como em 1.7 e 1.8.

$$P_k - V_k \sum_{m \in \kappa} V_m (G_{km} cos\theta_{km} + B_{km} sen\theta_{km}) = 0 \tag{1.7}$$

$$Q_k - V_k \sum_{m \in \kappa} V_m (G_{km} sen\theta_{km} - B_{km} cos\theta_{km}) = 0 \tag{1.8}$$

1.2.2 Matriz de Admitância

Seja a matriz de admitância nodal Y e ela se relacione com o vetor de tensões nodais V e vetor de injeções de corrente nodais I da forma como em 1.9.

$$I = Y.V \tag{1.9}$$

A regra de formação dessa matriz Y será dada por 1.10 e 1.11 (A dedução desse conjunto de formulas não será abordado neste trabalho).
A dimensão da matriz Y será de NBxNB.
Para os elementos fora da diagonal principal, será o valor negativo da admitância série entre as duas barras, como em 1.10.

$$Y_{km} = -y_{km} \tag{1.10}$$

Para os elementos na diagonal principal, será a soma das admitâncias conectadas à barra, como em 1.11, onde Ω_k é o conjunto de barras vizinhas da barra k.

$$Y_{kk} = \sum_{m \in \Omega_k} \left(y_{km} + j\frac{b_{km}^{sh}}{2} \right) \tag{1.11}$$

Ainda, a matriz Y será dividida em parte real(1.12) e parte imaginária (1.13). Dando origem às matrizes de condutância G e susceptância B, usadas nas equações de injeção de potência 1.1 e 1.2.

$$G = \mathbb{R}(Y) \tag{1.12}$$

$$B = \mathbb{I}(Y) \tag{1.13}$$

1.3 Formulação do método de Newton

O algoritmo de Newton é um método para calcular os zeros de funções reais de uma variável reais. Baseando em uma aproximação inicial arbitraria, $x^{(1)}$, tem-se 1.14 para $n > 1$. (UFRGS, Consultado em 05/2020)

$$x^{x+1} = x^n - \frac{f(x^{(n)}}{f'(x^{(N)}} = x^n - \Delta x \tag{1.14}$$

Fazendo $\Delta x = \frac{f(x^{(n)}}{f'(x^{(N)}}$

$$x^{x+1} = x^n - \Delta x \tag{1.15}$$

1.3.1 Aplicação do método de Newton em problemas de Fluxo de Potência

Para solucionar o problema do Fluxo de Potência utilizando o método de Newton, é necessário estabelecer equações que relacionem o problema na forma de $f(x) = 0$. A forma mais adequada, é utilizando as equações que modelam o balanceamento das injeções de potência 1.7 e 1.8, mas desta vez elas não vão a 0, pois ainda não estão resolvidas. Neste momento, igualaremos elas um Δ, que é tão próximo de 0 quanto se queira, como descrito nas equações 1.16 e 1.17.

$$\Delta P_k = P_k^{esp} - P_k^{calc}(V,\theta) = P_k^{esp} - V_k \sum_{m \in \kappa} V_m(G_{km}cos\theta_{km} + B_{km}sen\theta_{km}) \tag{1.16}$$

$$\Delta Q_k = Q_k^{esp} - Q_k^{calc}(V,\theta) = Q_k^{esp} - V_k \sum_{m \in \kappa} V_m(G_{km}sen\theta_{km} - B_{km}cos\theta_{km}) \tag{1.17}$$

Onde:

- $P_k^{esp} = P_k^G - P_k^C$ e $Q_k^{esp} = Q_k^G - Q_k^C$ que são os valores das injeções de potência ativa e reativa especificados para as barras.
- $P_k^{calc}(V,\theta)$ e $Q_k^{calc}(V,\theta)$ são calculados através das equações das potências nodais.
- ΔP_k e ΔQ_k são chamados de mismatches ou resíduos de potência ativa e reativa.

Em termos práticos, para a resolução do sistema de equações $g(x) = 0$ pelo método de Newton, é necessário a determinação do vetor de correção do estado Δx a cada iteração. Para cada iteração v, Δx é obtido através de 1.18 e verificado se a convergência ocorreu, isto é, se $|\Delta_x| < Tol$.

$$g(x^v) = -J(x^v).\Delta(x^v) \tag{1.18}$$

Onde:

$$g(x^v) = \begin{bmatrix} \Delta P^v \\ \Delta Q^v \end{bmatrix} \tag{1.19}$$

$$\Delta(x^v) = \begin{bmatrix} \Delta\theta^v \\ \Delta V^v \end{bmatrix} \tag{1.20}$$

$$J(x^{\nu}) = -\begin{bmatrix} \frac{\partial(P)}{\partial\theta} & \frac{\partial(P)}{\partial V} \\ \frac{\partial(Q)}{\partial\theta} & \frac{\partial(Q)}{\partial V} \end{bmatrix} \tag{1.21}$$

As submatrizes que compõem a matriz Jacobiana 1.21 são geralmente representadas por 1.22, 1.23, 1.24 e 1.25 e são obtidas equações das potências nodais 1.1 e 1.2

$$H = \frac{\partial(P)}{\partial\theta} \tag{1.22}$$

$$N = \frac{\partial(P)}{\partial V} \tag{1.23}$$

$$M = \frac{\partial(Q)}{\partial\theta} \tag{1.24}$$

$$L = \frac{\partial(Q)}{\partial V} \tag{1.25}$$

Resultando em 1.26.

$$\begin{bmatrix} \Delta P^{\nu} \\ \Delta Q^{\nu} \end{bmatrix} = \begin{bmatrix} H & N \\ M & L \end{bmatrix}^{\nu} . \begin{bmatrix} \Delta\theta^{\nu} \\ \Delta V^{\nu} \end{bmatrix} \tag{1.26}$$

As mastrizes H, N, M e L podem ter seus valores calculados utilizando as equações 1.27, 1.28, 1.29 e 1.30.(As deduções das matrizes H, N, M e L podem ser encontradas em livros didáticos e, portanto, serão omitidas neste trabalho).

$$\begin{cases} H_{kk} = \frac{\partial(P_k)}{\partial\theta_k} = -B_{kk}V_k^2 - V_k \sum_{m\in\kappa} V_m(G_{km}sen\theta_{km} - B_{km}cos\theta_{km}) \\ H_{km} = \frac{\partial(P_k)}{\partial\theta_m} = V_k V_m(G_{km}sen\theta_{km} - B_{km}cos\theta_{km}) \\ H_{mk} = \frac{\partial(P_k)}{\partial\theta_k} = -V_k V_m(G_{km}sen\theta_{km} + B_{km}cos\theta_{km}) \end{cases} \tag{1.27}$$

$$\begin{cases} N_{kk} = \frac{\partial(P_k)}{\partial V_k} = G_{kk}V_k \sum_{m\in\kappa} V_m(G_{km}cos\theta_{km} + B_{km}sen\theta_{km}) \\ N_{km} = \frac{\partial(P_k)}{\partial V_m} = V_k(G_{km}cos\theta_{km} + B_{km}sen\theta_{km}) \\ N_{mk} = \frac{\partial(P_m)}{\partial V_k} = V_m(G_{km}cos\theta_{km} + B_{km}sen\theta_{km}) \end{cases} \tag{1.28}$$

$$\begin{cases} M_{kk} = \frac{\partial(Q_k)}{\partial\theta_k} = -G_{kk}V_k^2 + V_k \sum_{m\in\kappa} V_m(G_{km}cos\theta_{km} + B_{km}sen\theta_{km}) \\ M_{km} = \frac{\partial(Q_k)}{\partial\theta_m} = -V_k V_m(G_{km}cos\theta_{km} + B_{km}sen\theta_{km}) \\ M_{mk} = \frac{\partial(Q_m)}{\partial\theta_k} = -V_k V_m(G_{km}cos\theta_{km} - B_{km}sen\theta_{km}) \end{cases} \tag{1.29}$$

$$\begin{cases} L_{kk} = \frac{\partial(Q_k)}{\partial V_k} = -B_{kk}V_k + \sum_{m\in\kappa} V_m(G_{km}sen\theta_{km} - B_{km}cos\theta_{km}) \\ L_{km} = \frac{\partial(Q_k)}{\partial V_m} = V_k(G_{km}sen\theta_{km} - B_{km}cos\theta_{km}) \\ L_{mk} = \frac{\partial(Q_m)}{\partial V_k} = -V_m(G_{km}sen\theta_{km} + B_{km}cos\theta_{km}) \end{cases} \tag{1.30}$$

1.4 Algoritmo implementado

Aqui será utilizado um algoritmo discutido nos slides do Professor Castro. (CASTRO, Consultado em 05/2020).

1. O contador de iterações é inicializado $\nu = 0$. Escolhe-se condições de contorno iniciais para as magnitudes das tensões nodais nas barras PQ e ângulos de fase das tensões nodais nas barras PQ e PV, onde não houver esse valor previamente estabelecido. O vetor x pode ser montado como em 1.31.

$$x = \begin{bmatrix} \theta^0 & V^0 \end{bmatrix}^T \tag{1.31}$$

2. Calcula-se $P_k(V_\nu, \theta)$ para as barras PQ e PV.
   Calcula-se $Q_k(V_\nu, \theta_\nu)$ para as barras PQ. Calcula-se os respectivos mismatches de potência 1.20.

3. Testa-se a convergência: Se $max\{|\Delta P_k^\nu|\}_k = PQ\ , PV \leq \varepsilon P$ e $max\{|\Delta Q_k^\nu|\}_k = PQ \leq \varepsilon P$
   Então processo iterativo convergiu para a solução $[\theta^\nu V^\nu]^T$. Pular para o passo 7.
   Senão, prosseguir para passo 4.

4. Calcula-se a matriz Jacobiana como em 1.21

5. Calcula-se as correções resolvendo o sistema 1.26 e determina-se nova solução ($V^{\nu+1}$, $\theta^{\nu+1}$)

6. Incrementa-se contador de iterações $\nu = \nu + 1$ e volta-se ao passo 2.

7. Calcula-se P_k para a barra de referência e Q_k para as barras de referência e PV.

Capítulo 2

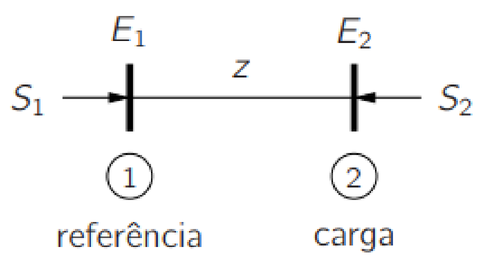
Estudos de caso

Neste capítulo, será estudado uma rede elementar de 2 barras e 1 linha de transmissão (na seção 2.1) e uma rede maior, de 14 barras e 20 ramos (na seção 2.2). Neste trabalho, a rede elementar tem um papel fundamental para compreensão de cada iteração do método, além de servir como calibração e teste para todas as modificações subsequentes. Para cada modificação no código de referência, ainda que pequena, deve ser retestado tendo a saída desta rede como parâmetro.

2.1 Rede de 2 barras e 1 linha de transmissão

Considere a rede de duas barras e uma linha de transmissão 2.1. Esse é o exemplo mais simples possível e foi apresentado e resolvido nos slides de aula.

Figura 2.1: Rede de duas barras e uma linha de transmissão



$$Dados \begin{cases} E_1 = 1,0112\angle 0^o pu \\ z = 0,01 + j0,05pu \\ S_2 = -(1,0 + j0)pu \end{cases} \tag{2.1}$$

2.1.1 Equacionamento

A matriz admitância será dada por 2.2, que foi calculado com as regras 1.10 e 1.11.

$$Y = \begin{bmatrix} 3,8462 - j19,2308 & -3,8462 + j19,2308 \\ -3,8462 + j19,2308 & 3,8462 - j19,2308 \end{bmatrix} \tag{2.2}$$

Portanto, tem-se as equações nodais P_2 e Q_2 em 2.3. As incognitas são θ_2 e V_2

$$\begin{cases} P_2 = V_2V_1(G_{21}cos\theta_{21} + B_{21}sen\theta_{21}) + V_2^2G_{22} \\ Q_2 = V_2V_1(G_{21}sen\theta_{21} - B_{21}cos\theta_{21}) - V_2^2B_{22} \end{cases} \tag{2.3}$$

Portanto, as equações de fluxo serão 2.4

$$\begin{cases} P_2^{esp} - P_2 = -1 - P_2 = 0 \\ Q_2^{esp} - Q_2 = 0 - Q_2 = 0 \end{cases} \tag{2.4}$$

Por fim, monta-se as equações linearizadas de fluxo de carga em 2.5.

$$\begin{bmatrix} \Delta P_2 \\ \Delta Q_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial(P_2)}{\partial\theta_2} & \frac{\partial(P_2)}{\partial V_2} \\ \frac{\partial(Q_2)}{\partial\theta_2} & \frac{\partial(Q_2)}{\partial V_2} \end{bmatrix} . \begin{bmatrix} \Delta\theta_2 \\ \Delta V_2 \end{bmatrix} \tag{2.5}$$

Onde, a partir de 1.27, 1.28, 1.29 e 1.30, tem-se 2.6.

$$\begin{cases} \frac{\partial(P_2)}{\partial\theta_2} = -V_2V_1(G_{21}sen\theta_{21} - B_{21}cos\theta_{21}) \\ \frac{\partial(P_2)}{\partial V_2} = V_1(G_{21}cos\theta_{21} + B_{21}sen\theta_{21}) + 2V_2G_{22} \\ \frac{\partial(Q_2)}{\partial\theta_2} = V_2V_1(G_{21}cos\theta_{21} + B_{21}sen\theta_{21}) \\ \frac{\partial(Q_2)}{\partial V_2} = -V_1(G_{21}sen\theta_{21} - B_{21}cos\theta_{21}) \end{cases} \tag{2.6}$$

Processo iterativo $\nu = 0$

Para a primeira iteração ($\nu = 0$), tem-se:

$$\begin{cases} V_2 = 1pu \\ \theta_2 = 0 \end{cases} \tag{2.7}$$

$$\begin{cases} P_2 = -0,0431 \\ \Delta P_2 = -0,9569 \\ Q_2 = -0,2154 \\ \Delta Q_2 = 0,2154 \end{cases} \tag{2.8}$$

Como os mismatches ainda são maiores que a tolerância, atualiza-se o estado, incrementa-se $\nu = \nu + 1$. A nova matriz Jacobiana ficará com 2.9.

$$J = \begin{bmatrix} 19,4462 & 3,8031 \\ -3,8031 & 19,0154 \end{bmatrix} \tag{2.9}$$

Processo iterativo $\nu = 1$

Para a segunda iteração ($\nu = 1$), tem-se:

$$\begin{cases} P_2 = -0,9960 \\ \Delta P_2 = -0,0040 \\ Q_2 = 0,0240 \\ \Delta Q_2 = -0,0240 \end{cases} \tag{2.10}$$

Como os mismatches ainda são maiores que a tolerância, atualiza-se o estado, incrementa-se $\nu = \nu + 1 = 2$. A nova matriz Jacobiana ficará com 2.11.

$$J = \begin{bmatrix} 19,2535 & 2,8560 \\ -4,8515 & 19,2781 \end{bmatrix} \tag{2.11}$$

Processo iterativo $\nu = 2$

Para a terceira iteração ($\nu = 2$), tem-se:

$$\begin{cases} P_2 = -1 \\ \Delta P_2 = 0 \\ Q_2 = 0 \\ \Delta Q_2 = 0 \end{cases} \tag{2.12}$$

Como os mismatches são menores que a tolerância, o método convergiu.

Calculo de E_2

Pela equação de potência na barra 2, tem-se 2.13.

$$S_1 = E_1.I_1^*2 = E1.[\frac{1}{Z}.(E_1 - E_2]^* = 1.01 + j0,05pu \tag{2.13}$$

Resumo das iterações está aprensentado na tabela 2.1.

Tabela 2.1: Resumo das iterações da rede

Iteração	E_2	E_2
0	1+j0	$1\angle 0^o$
1	1-j0,0494	$1\angle - 2.84^o$
2	0,09988-0,0495	$1\angle - 2,8^o$

2.1.2 Solução por software

Com os dados 2.1 é possivel montar os *arrays* no código abaixo. Eles serão usados como *inputs* do código utilizado neste trabalho.

```
nome_da_rede = 'Rede de 2 barras e 1 ramos - Demostração do Slide';
baseMVA      = 1  ; % MVA - Os dados do problema já estavam em pu
%Núm-Tipo-V-Âng-Pg-Qg-Pc-Qc-bshk % (PU/Graus) - 3ref; 2PV; 0PQ
barras = [1 3 1.0112 0 0 0 00.0 0 0
          2 0 0.0000 0 0 0 -1.0 0 0 ];
ramos  = [1 2 0.01 0.05 0.0 1.0 0.0 ];%De-Para-r-x-bshl-tap-fi
```

A saída dessa rede, calculado pelo algoritmo descrito no capítulo 3, é dado nos blocos de código a seguir. Nesse bloco temos a formação da matriz de admitância. É a mesma matriz calculada em 2.2, como esperado.

```
Y =
3.8462 -19.2308i  -3.8462 +19.2308i
-3.8462 +19.2308i   3.8462 -19.2308i
G =
3.8462   -3.8462
-3.8462    3.8462
B =
-19.2308    19.2308
19.2308   -19.2308
```

As iterações e os mismatches estão no próximo bloco

```
> Iteração - 0
  * Máximo mismatch P = 1.0431 (barra 0002)
  * Máximo mismatch Q = 0.2154 (barra 0002)
> Iteração - 1
  * Máximo mismatch P = -0.0268 (barra 0002)
  * Máximo mismatch Q = -0.0291 (barra 0002)
> Iteração - 2
  * Máximo mismatch P = -0.0000 (barra 0002)
  * Máximo mismatch Q = -0.0001 (barra 0002)
```

Após 3 iterações, o método convergiu para solução com tolerância de $0,001pu$.

```
> Estado da rede
  Barra Tipo        Mag   Fase          P       Q        Qsh
      1    3     1.0112   0.00    -0.9904  0.0480    0.0000
      2    0     1.0198   2.78     1.0000  0.0001    0.0000
> Fluxos de potência
     De Para        Pkm     Qkm         Pmk     Qmk      Ploss    Qloss
      1    2    -0.9904  0.0480      1.0000  0.0001     0.0096   0.0481
> Tempo computacional =  0.0798 segundos.
```

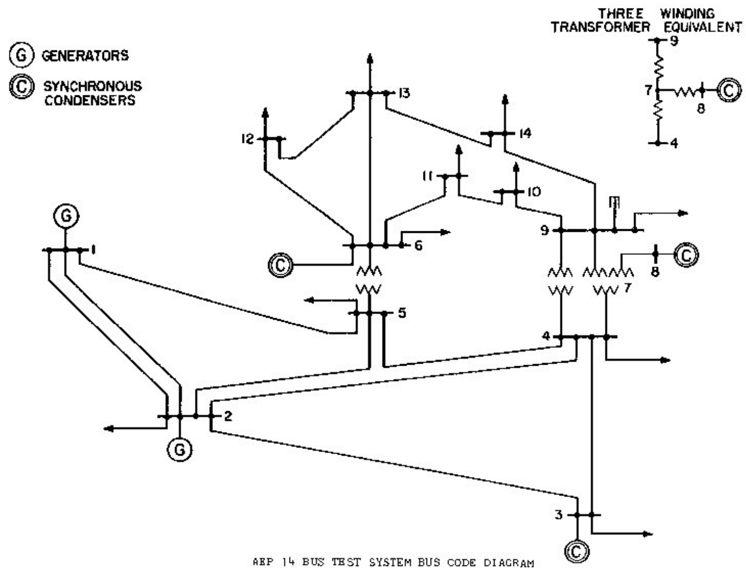
Note que na barra 2, a tensão está como calculado na tabela 2.1. $E_2 = 1\angle - 2,8^o$.

2.2 Rede 14 bus IEEE

Considere uma rede de 14 barras e 20 ramos. Este exemplo consta no código de referência e também pode ser encontrado no endereço: https://labs.ece.uw.edu/pstca/pf14/pg_tca14bus.htm. Neste trabalho não haverá equacionamento manual desta rede pois seria muito extenso. Portanto, apresentaremos dados da simulação.

Na figura 2.2, há a figura do estudo de caso de rede 14 barras e 20 ramos do IEEE.

Figura 2.2: Rede de 14 barras IEEE



2.2.1 Entradas do software

Esta é a tabela de entradas com os parâmetros da rede, que foi disponibilizada pelo IEEE.

```
nome_da_rede = '14 barras IEEE';
baseMVA = 1.0; %pneves já está dividido por 100
%         Número - Tipo - V - Ângulo - Pg - Qg - Pc - Qc - bshk
barras =[ 1    3    1.00    0.00    0.00    0.00   0.00    0.000    0.00
          2    2    1.00    0.00    0.00    0.00   0.217   0.127    0.00
          3    2    1.00    0.00    0.00    0.00   0.942   0.190    0.00
          4    0    1.00    0.00    0.00    0.00   0.478  -0.039    0.00
          5    0    1.00    0.00    0.00    0.00   0.076   0.016    0.00
          6    2    1.00    0.00    0.00    0.00   0.112   0.075    0.00
          7    0    1.00    0.00    0.00    0.00   0.000   0.000    0.00
          8    2    1.00    0.00    0.00    0.00   0.000   0.000    0.00
          9    0    1.00    0.00    0.00    0.00   0.295   0.166    0.19
```

```
          10    0    1.00    0.00    0.00    0.00   0.900   0.058    0.00
          11    0    1.00    0.00    0.00    0.00   0.350   0.018    0.00
          12    0    1.00    0.00    0.00    0.00   0.610   0.016    0.00
          13    0    1.00    0.00    0.00    0.00   0.135   0.058    0.00
          14    0    1.00    0.00    0.00    0.00   0.149   0.050    0.00];
%         De - Para - r - x - bshl - tap - fi
ramos=[1     5  0.05403   0.22304     0.0492  1.000   0.00
       1     2  0.01938   0.05917     0.0528  1.000   0.00
       2     5  0.05695   0.17388     0.0346  1.000   0.00
       2     4  0.05811   0.17632     0.0340  1.000   0.00
       2     3  0.04699   0.19797     0.0438  1.000   0.00
       3     4  0.06701   0.17103     0.0000  1.000   0.00
       4     5  0.01335   0.04211     0.0000  1.000   0.00
       4     7  0.00000   0.20912     0.0000  1.000   0.00
       4     9  0.00000   0.55618     0.0000  1.000   0.00
       5     6  0.00000   0.25202     0.0000  1.000   0.00
       6    12  0.12291   0.25581     0.0000  1.000   0.00
       6    13  0.06615   0.13027     0.0000  1.000   0.00
       6    11  0.09498   0.19890     0.0000  1.000   0.00
       7     9  0.00000   0.11001     0.0000  1.000   0.00
       7     8  0.00000   0.17615     0.0000  1.000   0.00
       9    10  0.03181   0.08450     0.0000  1.000   0.00
       9    14  0.12711   0.27038     0.0000  1.000   0.00
      10    11  0.08205   0.19207     0.0000  1.000   0.00
      12    13  0.22092   0.19988     0.0000  1.000   0.00
      13    14  0.17093   0.34802     0.0000  1.000   0.00];
```

2.2.2 Solução por software

É possivel notar que com tolerância = 0.000001pu, a rede convergiu para solução em 4 iterações, em 0.39s

```
> Relatório final
  Convergiu em 4 iterações
> Estado da rede
Barra Tipo       Mag   Fase         P       Q       Qsh
  1    3    1.0000   -0.00    4.9041 -0.5519    0.0000
  2    2    1.0000 -12.32   -0.2170  1.5340    0.0000
  3    2    1.0000 -24.46   -0.9420  0.7185    0.0000
```

4	0	0.9301	-23.23	-0.4780	0.0390	0.0000
5	0	0.9300	-20.79	-0.0760	-0.0160	0.0000
6	2	1.0000	-41.97	-0.1120	0.9492	0.0000
7	0	0.9434	-34.85	-0.0000	0.0000	0.0000
8	2	1.0000	-34.85	0.0000	0.3212	0.0000
9	0	0.9304	-40.93	-0.2950	-0.1660	0.1645
10	0	0.9014	-46.00	-0.9000	-0.0580	0.0000
11	0	0.9289	-46.05	-0.3500	-0.0180	0.0000
12	0	0.9206	-47.88	-0.6100	-0.0160	0.0000
13	0	0.9593	-44.41	-0.1350	-0.0580	0.0000
14	0	0.9225	-43.72	-0.1490	-0.0500	0.0000

> Fluxos de potência

De	Para	Pkm	Qkm	Pmk	Qmk	Ploss	Qloss
1	5	1.5319	0.1897	-1.4026	0.2981	0.1293	0.4878
1	2	3.3722	-0.7416	-3.1419	1.3920	0.2303	0.6503
2	5	0.8477	0.1660	-0.8049	-0.0675	0.0428	0.0985
2	4	1.0464	0.1297	-0.9816	0.0355	0.0649	0.1652
2	3	1.0307	-0.1537	-0.9800	0.3236	0.0507	0.1700
3	4	0.0380	0.3949	-0.0274	-0.3680	0.0105	0.0269
4	5	-0.7876	0.2713	0.7983	-0.2375	0.0107	0.0338
4	7	0.8454	0.0269	-0.8454	0.1460	0.0000	0.1729
4	9	0.4732	0.0733	-0.4732	0.0741	0.0000	0.1474
5	6	1.3333	-0.0091	-1.3333	0.5271	0.0000	0.5180
6	12	0.4295	0.1233	-0.4049	-0.0722	0.0245	0.0511
6	13	0.3778	0.1271	-0.3673	-0.1064	0.0105	0.0207
6	11	0.4139	0.1717	-0.3949	-0.1317	0.0191	0.0399
7	9	0.8454	0.1570	-0.8454	-0.0656	0.0000	0.0914
7	8	0.0000	-0.3030	0.0000	0.3212	0.0000	0.0182
9	10	0.8854	0.0242	-0.8566	0.0524	0.0288	0.0766
9	14	0.1382	-0.0342	-0.1352	0.0406	0.0030	0.0063
10	11	-0.0434	-0.1104	0.0449	0.1137	0.0014	0.0033
12	13	-0.2051	0.0562	0.2168	-0.0456	0.0118	0.0107
13	14	0.0155	0.0940	-0.0138	-0.0906	0.0017	0.0034

> Tempo computacional = 0.3942 segundos.

Capítulo 3

Código comentado

Os códigos fonte desse trabalho bem como histórico de modificação, podem ser encontrados no endereço: *https://github.com/phneves/ElectricPowerSystemsAnalysisTools.*

3.1 Entradas e configurações

Iniciamos com limpeza das variáveis no *workspace* e da tela. Também há configurações do script.

```
clear all;
clc;
tol = 0.00001;
% Arquivo de dados da rede
%P2GTD
RedePequena
graf = 'nao'; % Mostrar graficos ao final do cálculo
itmax = 20;% Número máximo de iterações permitido
% Tensões mínima e máxima permitidas
vmin = 0.0;
vmax = 2.0;
% Considerar cargas dependentes da tensão?
% 'nao'->modelo de pot cte. 'sim'->ModeloCarga deve ser especificado
cargasV = 'nao';
graus_to_rad = pi/180;% Conversão graus <-> radianos
rad_to_graus = 180/pi;
Debug = true; %pneves: Enable debug prints.
```

Carrega-se então, os valores da rede nas variáveis apropriadas. Vetores receberão dados das barras.

```
%Número - Tipo - V - Ângulo - Pg - Qg - Pc - Qc - bshk
for k = 1:nb            %pneves: extracting bar values into arrays
    numext(k)    = barras(k,1);
    tipo(k)      = barras(k,2);
    v(k)         = barras(k,3);
    ang(k)       = barras(k,4) * graus_to_rad;
    pg(k)        = barras(k,5)/baseMVA; %pneves: PU
    qg(k)        = barras(k,6)/baseMVA;
    pc(k)        = barras(k,7)/baseMVA;
    qc(k)        = barras(k,8)/baseMVA;
    bshk(k)      = barras(k,9)/baseMVA;
        pnom(k)      = pg(k) - pc(k); %pneves: P liquido
        qnom(k)      = qg(k) - qc(k); %pneves: Q liquido
    numint(barras(k,1)) = k;
end
```

Vetores receberão dados dos ramos.

```
for l = 1:nr                    %        Carregamento dos vetores de ramo
        de(l)    = ramos(l,1);
        para(l)  = ramos(l,2);
        r(l)     = ramos(l,3);
        x(l)     = ramos(l,4);
        bshl(l)  = ramos(l,5)/2.; %shunt aqui é dividido por 2
        tap(l)   = ramos(l,6);
        fi(l)    = 0; %- ramos(l,7) * graus_to_rad;
end
fprintf('\n> Modelo de carga: potência constante (default).\n')
```

Neste ponto do código, será criada a matriz de admitância. A regra de formação dessa matriz está descrita em 1.10 e 1.11.

```
%pneves: Matriz Y dimensionada para NBxNB
Y = zeros(nb,nb);
for k = 1:nb
    Y(k,k) = i*bshk(k); %pneves: Cada Y(k,k) recebe 0+i*shunt
end
```

Preenchimento da matriz admitância.

```
for l = 1:nr %pneves: As formulas de formação da matriz
    k = de(l)%estão no Tópico Matriz de Admitância
    m = para(l)
    y(l) = 1/(r(l) + i*x(l))
    akk(l) = 1/(tap(l)*tap(l))
    amm(l) = 1.0
    akm(l) = 1/tap(l)
    Y(k,k) = Y(k,k) + akk(l)*y(l) + i*bshl(l)
    Y(m,m) = Y(m,m) + amm(l)*y(l) + i*bshl(l)
    Y(k,m) = Y(k,m) - akm(l)*y(l)
    Y(m,k) = Y(m,k) - akm(l)*y(l)
end
G = real(Y); %pneves: matriz de condutância
B = imag(Y); %pneves: matriz de susceptância
if (Debug == true) %pneves: Debug
    fprintf('DEBUG> Y, G and B complete\n');
    Y
    G
    B
end
```

Aqui será definido o estado inicial da rede, com $V = 1$ PU para barras PQ, com $\theta = 0$ para barras PQ e PV. O chamado *Flat guess*.

```
for k = 1:nb
    if tipo(k) ~= 3
        ang(k) = 0.0;
        if tipo(k) < 2
            v(k)= 1.0;
        end
    end
end
```

3.2 Subsistema 1 - Iterativo

Calcula-se as Potências nodais.

$pcalc(k)$ e $qcalc(k)$ são calculados para cada barra. Eles serão usados nos calculos de H_{kk}, N_{kk},

M_{kk} e L_{kk}, submatrizes da matriz Jacobiana 1.21.

```
iter = 0;         %         Inicializar contador de iterações
maxDP = 10^5;     %         Inicializar maiores mismatches
maxDQ = 10^5;
while abs(maxDP) > tol | abs(maxDQ) > tol %        Processo iterativo
    %Cálculo das potências nodais
    for k = 1:nb
        pcalc(k) =  G(k,k)*v(k)*v(k);
        qcalc(k) = -B(k,k)*v(k)*v(k);
    end
    for l = 1:nr
        k = de(l);
        m = para(l);
        ab = ang(k) - ang(m) + fi(l);
        gkm = akm(l)*real(y(l));
        bkm = akm(l)*imag(y(l));
        pcalc(k) = pcalc(k) + v(k)*v(m)*(-gkm*cos(ab)-bkm*sin(ab));
        pcalc(m) = pcalc(m) + v(k)*v(m)*(-gkm*cos(ab)+bkm*sin(ab));
        qcalc(k) = qcalc(k) + v(k)*v(m)*(-gkm*sin(ab)+bkm*cos(ab));
        qcalc(m) = qcalc(m) - v(k)*v(m)*(-gkm*sin(ab)-bkm*cos(ab));
    end
```

As equações de $pcalc(k)$ (bloco de código acima) são termos somatórios descritas nas equações 1.27, 1.28, 1.29 e 1.30.

No bloco de código abaixo, tem-se o cáculo dos *mismatches* de potência (P e Q).

$$\begin{cases} DP(k) = pesp(k) - pcalc(k) \\ DQ(k) = qesp(k) - qcalc(k) \end{cases} \tag{3.1}$$

As equações descritas em 3.1 são as diferenças entre as potências esperadas e as calculadas, respectivamente para P e Q.

Esses valores serão os novos ΔP e ΔQ, como descrito na equação 1.20.

No bloco subsequente, a variável de controle do número de iteração é incrementada. Neste algorítmo, v está descrito como *iter*.

O vetor *DS* é montado.

```
%Cálculo dos mismatches de potência
DP = zeros(nb,1);
DQ = zeros(nb,1);
maxDP = 0;
maxDQ = 0;
busDP = 0;
busDQ = 0;
for k = 1:nb
    pesp(k) = pnom(k);
    qesp(k) = qnom(k);
    if tipo(k) ~= 3
        DP(k) = pesp(k) - pcalc(k);
        if abs(DP(k)) > abs(maxDP)
            maxDP = DP(k);
            busDP = numext(k);
        end
    end
    if tipo(k) <= 1
        DQ(k) = qesp(k) - qcalc(k);
        if abs(DQ(k)) > abs(maxDQ)
            maxDQ = DQ(k);
            busDQ = numext(k);
        end
    end
end
```

```
iteracao(iter+1) = iter;
mismP(iter+1) = abs(maxDP);
mismQ(iter+1) = abs(maxDQ);

DS = [ DP ; DQ ];

fprintf('\n> Iteração - %d\n',iter)
fprintf('* Máximo mismatch P = %06.4f (barra %04d)\n',maxDP,busDP)
fprintf('* Máximo mismatch Q = %06.4f (barra %04d)\n',maxDQ,busDQ)
```

Caso o *mismatch* seja maior que a tolerância, a matriz Jacobiana será montada e o estado será atualizado.

```
    if abs(maxDP) > tol | abs(maxDQ) > tol
        %        Montar e inverter a matriz Jacobiana
        H = zeros(nb,nb); M=H; N=H; L=H;
        for k = 1:nb
            H(k,k) = -qcalc(k) - v(k)*v(k)*B(k,k);
            if tipo(k) == 3
                H(k,k) = 10^10;
            end
            N(k,k) = ( pcalc(k) + v(k)*v(k)*G(k,k) )/v(k) ;
            M(k,k) = pcalc(k) - v(k)*v(k)*G(k,k);
            L(k,k) = ( qcalc(k) - v(k)*v(k)*B(k,k) )/v(k) ;
            if tipo(k) >= 2
                L(k,k) = 10^10;
            end
        end
        for l = 1 : nr,
              k = de(l) ;
              m = para(l) ;
              ab = ang(k) - ang(m) + fi(l) ;

              H(k,m) =   v(k)*v(m)*( G(k,m)*sin(ab)-B(k,m)*cos(ab)) ;
              H(m,k) =   v(k)*v(m)*(-G(k,m)*sin(ab)-B(k,m)*cos(ab)) ;
              N(k,m) =   v(k)*(G(k,m)*cos(ab)+B(k,m)*sin(ab)) ;
              N(m,k) =   v(m)*(G(k,m)*cos(ab)-B(k,m)*sin(ab)) ;
              M(k,m) = - v(k)*v(m)*(G(k,m)*cos(ab)+B(k,m)*sin(ab)) ;
              M(m,k) = - v(k)*v(m)*(G(k,m)*cos(ab)-B(k,m)*sin(ab)) ;
              L(k,m) =   v(k)*(G(k,m)*sin(ab)-B(k,m)*cos(ab)) ;
              L(m,k) = - v(m)*(G(k,m)*sin(ab)+B(k,m)*cos(ab)) ;
        end
        J = [ H N ; M L ];
        %        Calcular o vetor de correção de estado
        DV = inv(J)*DS;
        %        Atualizar estado
        v = v + DV(nb+1:2*nb)';
        ang = ang + DV(1:nb)';
```

Verifica-se se há divergência, ou seja, se o método está caminhando para uma resposta.

```
        %        Verificar se houve divergência
        for k = 1:nb
            if v(k) < vmin | v(k) > vmax
                fprintf('\n> Tensões fora dos limites ->
                divergência.\n')
                fprintf('> Execução interrompida.\n\n')
                if graf == 'sim'
                            graficos
                end
                msgbox('Tensões fora dos limites -> divergência.
                Execução interrompida.','Aviso','warn');
                return
            end
        end
```

Verifica-se se o método excedeu o número de iterações máximas.

```
    %        Incrementar contador de iterações

    iter = iter + 1;

        if iter > itmax
            fprintf('\n> Número máximo de iterações permitido foi
            excedido.\n')
            fprintf('> Execução interrompida.\n\n')
            if graf == 'sim'
                graficos
            end
            msgbox('Número máximo de iterações permitido foi
            excedido. Execução interrompida.','Aviso','warn');
            return
        end
    end
end          % final do while
```

3.3 Subsistema 2 - Cálculo de P_k e Q_k

Neste ponto, o software já fez todos os calculos e se saiu do laço de repetição *while*, então ele convergiu para resposta.

As potências serão recalculadas com os valores finais para gerar o relatório.

```
%%
%'Fim do cálculo de fluxo de carga. Preparando relatório de saída');
fprintf('\n> Fim do cálculo de fluxo de carga.
Preparando relatório de saída ...\n')
```

Será calculada as potências nodais.

```
%         Cálculo das potências nodais
for k = 1:nb
        pcalc(k) = G(k,k)*v(k)*v(k);
    qcalc(k) = -B(k,k)*v(k)*v(k);
end
for l = 1:nr
        k = de(l);
    m = para(l);
    ab = ang(k) - ang(m) + fi(l);
    gkm = akm(l)*real(y(l));
    bkm = akm(l)*imag(y(l));
    pcalc(k) = pcalc(k) + v(k)*v(m)*(-gkm*cos(ab)-bkm*sin(ab));
    pcalc(m) = pcalc(m) + v(k)*v(m)*(-gkm*cos(ab)+bkm*sin(ab));
    qcalc(k) = qcalc(k) + v(k)*v(m)*(-gkm*sin(ab)+bkm*cos(ab));
    qcalc(m) = qcalc(m) - v(k)*v(m)*(-gkm*sin(ab)-bkm*cos(ab));
end
```

Será calculado o fluxo de potência nos ramos

```
%         Cálculo dos fluxos de potência nos ramos
for l = 1:nr
    k = de(l);
    m = para(l);
    gkm = real(y(l));
    bkm = imag(y(l));
    ab = ang(k) - ang(m) + fi(l);
    vkm = v(k)*v(m);
```

```
    pkm(l) = akk(l)*v(k)*v(k)*gkm -
        akm(l)*vkm*(gkm*cos(ab)+bkm*sin(ab));
    pmk(l) = amm(l)*v(m)*v(m)*gkm -
        akm(l)*vkm*(gkm*cos(ab)-bkm*sin(ab));
    qkm(l) = -akk(l)*v(k)*v(k)*(bkm+bshl(l)) +
        akm(l)*vkm*(bkm*cos(ab)-gkm*sin(ab));
    qmk(l) = -amm(l)*v(m)*v(m)*(bkm+bshl(l)) +
        akm(l)*vkm*(bkm*cos(ab)+gkm*sin(ab));
    pperdas(l) = pkm(l) + pmk(l);
    qperdas(l) = qkm(l) + qmk(l);
end
```

Apresentação do relatório final.

```
fprintf('\n\n> Relatório final\n\n') %        Relatório final
fprintf('  Convergiu em %d iterações\n\n',iter)
fprintf('> Estado da rede\n\n')
fprintf('   Barra Tipo   Mag Fase    P   Q   Qsh\n')
for k = 1:nb
    fprintf('%7d %4d %9.4f %6.2f %9.4f %7.4f %9.4f \n',
    numext(k),tipo(k),v(k),(ang(k)*rad_to_graus),pcalc(k),
    qcalc(k), bshk(k)*v(k)^2)
end
fprintf('\n> Fluxos de potência\n\n')
fprintf('   De  Para    Pkm   Qkm   Pmk   Qmk Ploss   Qloss\n')
for l = 1:nr
    fprintf('%7d %4d %9.4f %7.4f %9.4f %7.4f %9.4f %7.4f\n',
    de(l),para(l),pkm(l),qkm(l),pmk(l),qmk(l),pperdas(l),qperdas(l))
end
if graf == 'sim'
    graficos
end
tempo = toc; % terminando contagem de tempo computacional
fprintf('\n\n> Tempo computacional = %7.4f segundos.',tempo)
fprintf('\n\n> Fim da simulação.\n\n')
```

Capítulo 4

Discussões e análise de resultados

4.1 Composição do trabalho

Foi abordado neste trabalho uma introdução à teoria de Fluxo de Potência e um dos métodos de resolução bastante importante, o método de Newton. Ainda que sem muitas demonstrações matemáticas, todas as equações que geram o modelo matemático e os passos necessários para alimentar o método de resolução estão descritas podem ser vistas nos tópicos 1.1, 1.2 e 1.3.

Já no capítulo 2, pôde-se colocar as equações supracitadas em prática. Utilizando redes de referência, foi possivel comparar e validar seus resultados. Foi utilizado um modelo de rede bastante pequeno no início, apenas duas barras e um ramo (2.1) e todo o passo a passo foi verificado. A rede convergiu em 3 iterações e levou $0,0798s$

Outra rede verificada, foi a de 14 barras e 20 ramos 2.2. Neste estudo não houve equacionamento por ser, praticamente, inviável calcular manualmente. A simulação convergiu após 4 iterações e levou $0,3942s$.

4.2 Performance

Comparando-se os dois casos, é notório que, com os aumentos de ramos e barras, a solução demorará mais para convergir. Ainda que nesse exemplo haja diferenças na tolerância da convergência, pode-se comprovar que com redes maiores, esse método apenas não trará resultados em prazos apropriados, ainda que a convergência aconteça.

Lembrando que a comparação de performance aconteceu apenas temporalmente, pois a tolerância foi fixada previamente.

4.3 Trabalhos futuros

Nos trabalhos futuros, ainda nesta disciplina, será importante comparar dois fatores com os próximos métodos, o primeiro e bastante abordado aqui: o tempo de convergência e o número de interações.

Ainda pode-se abordar a acuracia e robustês de cada solução. Basicamente, é necessário responder à seguinte pergunta: caso o palpite de solução inicial seja diferente do *flat initial guess*, o método ainda apresenta convergência?

Referências bibliográficas

CASTRO, P. D. C. A. **Cálculo de Fluxo de Carga**. Campinas-SP: [s.n.], Consultado em 05/2020. Disponível em: <http://www.dsee.fee.unicamp.br/~ccastro/>.

MONTICELLI, A. J. **Fluxo de carga em redes de energia elétrica**. 1. ed. São Paulo: Editora Edgard Blücher Ltda, 1983. ISBN 978-3-8417-2506-6.

UFRGS. **Método de Newton-Raphson**. Porto Alegre RS: [s.n.], Consultado em 05/2020. Disponível em: <https://www.ufrgs.br/reamat/CalculoNumerico/livro-sci/sdeduv-metodo_de_newton-raphson.html>.