UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Pedro Henrique Neves dos Santos

IT306W - Estimação de estado
Equação Normal e Tableau Esparso

Campinas
2020

Resumo

Neste quarto trabalho, será abordado os Estimadores de Estado, usando Equação Normal e Tableau Esparso.
Todos os códigos podem ser encontrados em
https://github.com/phneves/ElectricPowerSystemsAnalysisTools

Sumário

Capítulo 1

Introdução e teoria

A ferramenta de análise de sistemas de energia elétrica aqui discutida será a Estimação de Estado. Essa análise nos fornecerá um metodo de obter o estado mais provável do sistema (ou de parte dele) a partir de medidas realizadas e a partir do modelo (circuito equivalente) da rede. (MONTICELLI, 1999).

1.1 Modelagem

Para estimar o estado, será utilizado um conjunto de medidas que há disponível. Todas as redes de energia elétrica são constantemente monitoradas e, como toda medida, há uma incenteza atrelada, podendo ser erros aceitáveis e inaceitáveis (erros grosseiros).
É necessário que haja número de medidas em quantidade suficiente, para minimizar a influência dos erros grosseiros. Aliás, na hipótese de haver redundância no conjunto de medidas, é possível tratar erros nas medidas, já que pode-se lançar mão de ferramentas estatísticas.
Frequentemente, os valores calculados com o estado estimado, são mais confiáveis que o conjunto de medidas.
O modelo da rede e composto basicamente pelos circuitos equivalentes das linhas, transformadores, reguladores de tensão e por um gerador que proverá a referência angular da rede.
Dispositivos de chaveamento e geradores distribuídos também podem ser modelados.
A topologia da rede e determinada pelo Configurador Topológico, que deve ser executado a cada alteração na topologia da rede.(CASTRO, Consultado em 07/2020)

1.2 Formulação do problema básico

1.2.1 Mínimos Quadrados Ponderado

Será utilizado modelos que descrevem as medições e suas relações, como em 1.1

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} h_1(x_1, x_2, ..., x_n) \\ h_2(x_1, x_2, ..., x_n) \\ \vdots \\ h_m(x_1, x_2, ..., x_n) \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} = h(x) + e \tag{1.1}$$

Onde $h_i(x)$ é a função que correlaciona a i-ésima medição com o vetor de estado, $x^t = [x_1, x_2, \ldots, x_n]$ é o vetor de estado que contém as n variáveis de estado e $e^t = [e_1, e_2, \ldots, e_m]$ corresponde ao vetor de erros do processo de medição.
Os elementos de e possuem média nula, $E\{e_i e_i\} = 0$, e $Cov(e) = E[e.e^t] = R_z = diag\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2\}$.
A variância σ_i^2 é calculada de acordo com a precisão da medição.
No método dos mínimos quadrados ponderados a função 1.2 deve ser minimizada. O inverso do desvio padrão é usado na ponderação das medidas. (CASTRO, Consultado em 07/2020)

$$J(x) = \sum_{i=1}^{m} \frac{(z_i - h_i(x))^2}{\sigma_i^2} = [z - h(x)]' R_z^- 1[z - h(x)] \tag{1.2}$$

1.2.2 Equação normal

Ao minimizar a equação 1.2, é necessário derivá-la e igualar a zero. Como resultado, o estado estimado é obtido iterativamente com método dos minimos quadrados ponderados, conforme a equação 1.3.

$$G(x^\nu).\Delta x^\nu = H'(x^\nu).R_z^- 1[z - H(x^\nu)] \tag{1.3}$$

E,

$$x^{\nu+1} = x^\nu + \Delta x^\nu \tag{1.4}$$

A matriz Hessiana $G(x)$, chamada de matriz Ganho, é definida como em 1.5

$$G(x) = H'(x) R_z^{-1} H(x) \tag{1.5}$$

O uso da matriz G, de ganhos, é conhecido como solução via equação normal e esta solução apresenta boa convergênca. Em alguns casos, porém, a solução da equação 1.3 pode apresentar divergência (não será abordado neste trabalho).

A formação da matriz H, descrita em 1.5, é dada por 1.1.

Figura 1.1: Formulação da matriz H

$$
\begin{aligned}
\frac{\partial P_{km}}{\partial \theta_k} &= a_{km} V_k a_{mk} V_m g_{km} sin(\theta_{km} + \varphi_{km} - \varphi_{mk}) + \\
&\quad -a_{km} V_k a_{mk} V_m b_{km} cos(\theta_{km} + \varphi_{km} - \varphi_{mk}) \\
\frac{\partial P_{km}}{\partial \theta_m} &= -a_{km} V_k a_{mk} V_m g_{km} sin(\theta_{km} + \varphi_{km} - \varphi_{mk}) + \\
&\quad a_{km} V_k a_{mk} V_m b_{km} cos(\theta_{km} + \varphi_{km} - \varphi_{mk}) \\
\frac{\partial P_{km}}{\partial V_k} &= 2\ a_{km}^2 V_k g_{km} - a_{km} a_{mk} V_m g_{km} cos(\theta_{km} + \varphi_{km} - \varphi_{mk}) + \\
&\quad -a_{km} a_{mk} V_m b_{km} sin(\theta_{km} + \varphi_{km} - \varphi_{mk}) \\
\frac{\partial P_{km}}{\partial V_m} &= -a_{km} a_{mk} V_k g_{km} cos(\theta_{km} + \varphi_{km} - \varphi_{mk}) + \\
&\quad -a_{km} a_{mk} V_k b_{km} sin(\theta_{km} + \varphi_{km} - \varphi_{mk}) \\
\frac{\partial Q_{km}}{\partial \theta_k} &= -a_{km} V_k a_{mk} V_m b_{km} sin(\theta_{km} + \varphi_{km} - \varphi_{mk}) + \\
&\quad -a_{km} V_k a_{mk} V_m g_{km} cos(\theta_{km} + \varphi_{km} - \varphi_{mk}) \\
\frac{\partial Q_{km}}{\partial \theta_m} &= a_{km} V_k a_{mk} V_m b_{km} sin(\theta_{km} + \varphi_{km} - \varphi_{mk}) + \\
&\quad +a_{km} V_k a_{mk} V_m g_{km} cos(\theta_{km} + \varphi_{km} - \varphi_{mk}) \\
\frac{\partial Q_{km}}{\partial V_k} &= -2\ a_{km}^2 V_k (b_{km} + b_{km}^{sh}) + a_{km} a_{mk} V_m b_{km} cos(\theta_{km} + \varphi_{km} - \varphi_{mk}) + \\
&\quad -a_{km} a_{mk} V_m g_{km} sin(\theta_{km} + \varphi_{km} - \varphi_{mk}) \\
\frac{\partial Q_{km}}{\partial V_m} &= a_{km} a_{mk} V_k b_{km} cos(\theta_{km} + \varphi_{km} - \varphi_{mk}) + \\
&\quad -a_{km} a_{mk} V_k g_{km} sin(\theta_{km} + \varphi_{km} - \varphi_{mk})
\end{aligned}
$$

1.2.3 Tableau Esparso

Para evitar trabalhar com o produto de H (matriz G), como em 1.5, pode-se reescrever o problema como minimização com restrições, como em 1.6 e 1.7

$$minJ(x) = \frac{1}{2}[z - h(x)]R_z^{-1}[z - h(x)] \tag{1.6}$$

$$minJ(x) = \frac{1}{2}r' R_z^{-1} r \tag{1.7}$$

Sujeito a $r = z - h(x)$. Para resolver este problema, deve-se usar multiplicadores de Lagrange. A solução pode ser obtida iterativamente a partir de 1.8.

$$\begin{bmatrix} R_z & H(x^k) \\ H'(x^k) & 0 \end{bmatrix} \begin{pmatrix} \lambda^k \\ \Delta x^k \end{pmatrix} = \begin{bmatrix} z - h(x^k) \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta z(x^k) \\ 0 \end{bmatrix} \tag{1.8}$$

Neste processo, há a matriz H e não a matriz G, o que melhora o condicionamento da matriz a ser fatorada. (CASTRO, Consultado em 07/2020).

1.3 Algoritmo implementado

Basicamente, tem-se 4 etapas do processo de estimação de estado (NETO, Consultado em 07/2020).

1. Processamento da topologia. Todas as informações de representação da topologia são tratadas.

2. Analise de observabilidade. A partir do modelo barra-ramo obtido na primeira etapa, verifica-se se é possivel determinar as tensões e angulos em todas as barras, considerando as medidas disponíveis.

3. Estimação de estado. A partir da topologia, dos parâmetros e dos conjuntos de medidas disponíveis, o Estimador de Estado determina a estimação que melhor representa o sistema

4. Processamento de erros grosseiros. A presença de medidas analógicas com possibilidade de erros grosseiros afasta a solução verdadeira do estimador de estados. Se uma medida é identificada dessa forma, ela deve ser retirada da solução.

Capítulo 2

Estudos de caso

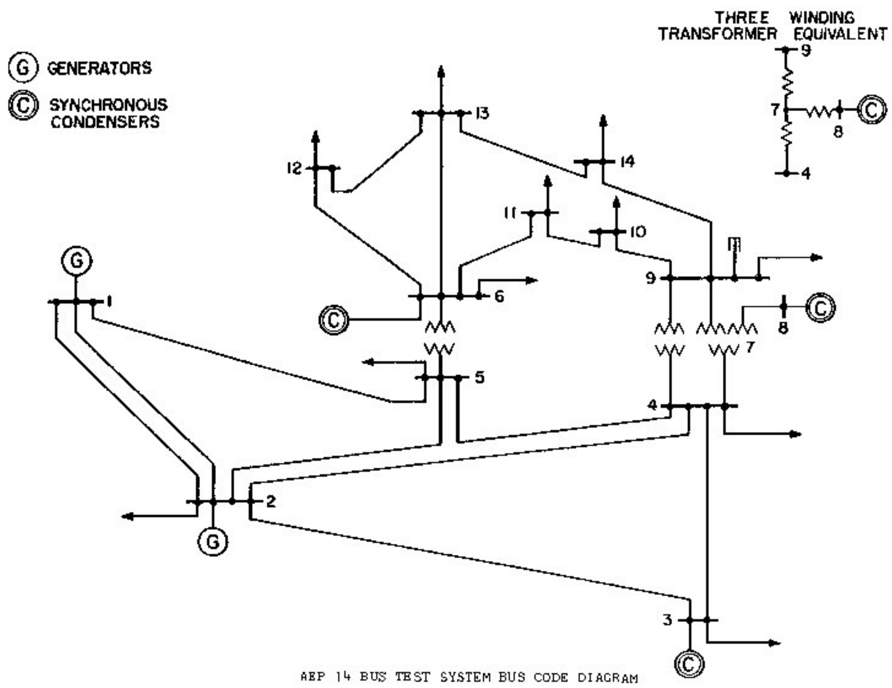
Neste capítulo, será estudado a rede de 14 barras do IEEE que pode ser encontrado em http://labs.ece.uw.edu/pstca/pf14/pg_tca14bus.htm

2.1 Rede de 14 barras IEEE

Considere a rede de 14 barras na figura 2.1.

Neste exercício, será resolvido as tensões e angulos de todas as barras desse sistema, utilizando

Figura 2.1: Rede de 14 barras IEEE



dados de medições disponíveis na rede.

2.1.1 Dados do Problema

Medições do problema 2.1. São dados obtidos da rede que serão usados para estimar o estado do sistema.

```
%           |Msnt |Type | Value | From | To | Rii |
zdata14   = [ %---- Voltage Magnitude ------------%
             1      1     1.06      1        0    9e-4;
             %-----------------------------------%
             %---- Real Power Injection ---------%
             2      2     0.1830    2        0    1e-4;
             3      2    -0.9420    3        0    1e-4;
             4      2     0.00      7        0    1e-4;
             5      2     0.00      8        0    1e-4;
             6      2    -0.0900   10        0    1e-4;
             7      2    -0.0350   11        0    1e-4;
             8      2    -0.0610   12        0    1e-4;
             9      2    -0.1490   14        0    1e-4;
            %------------------------------------%
            %---- Reative Power Injection -------%
            10      3     0.3523    2        0    1e-4;
            11      3     0.0876    3        0    1e-4;
            12      3     0.00      7        0    1e-4;
            13      3     0.2103    8        0    1e-4;
            14      3    -0.0580   10        0    1e-4;
            15      3    -0.0180   11        0    1e-4;
            16      3    -0.0160   12        0    1e-4;
            17      3    -0.0500   14        0    1e-4;
            %------------------------------------%
            %------ Real Power Flow ------------- %
            18      4     1.5708    1        2    64e-6;
            19      4     0.7340    2        3    64e-6;
            20      4    -0.5427    4        2    64e-6;
            21      4     0.2707    4        7    64e-6;
            22      4     0.1546    4        9    64e-6;
            23      4    -0.4081    5        2    64e-6;
            24      4     0.6006    5        4    64e-6;
            25      4     0.4589    5        6    64e-6;
            26      4     0.1834    6       13    64e-6;
            27      4     0.2707    7        9    64e-6;
```

```
         28     4   -0.0816  11       6   64e-6;
         29     4    0.0188  12      13   64e-6;
         %----------------------------------%
         %------ Real Power Flow ------------- %
         30     5   -0.1748   1       2   64e-6;
         31     5    0.0594   2       3   64e-6;
         32     5    0.0213   4       2   64e-6;
         33     5   -0.1540   4       7   64e-6;
         34     5   -0.0264   4       9   64e-6;
         35     5   -0.0193   5       2   64e-6;
         36     5   -0.1006   5       4   64e-6;
         37     5   -0.2084   5       6   64e-6;
         38     5    0.0998   6      13   64e-6;
         39     5    0.1480   7       9   64e-6;
         40     5   -0.0864  11       6   64e-6;
         41     5    0.0141  12      13   64e-6;];
         %------------------------------------%
```

Dados das barras do problema 2.1

```
% |Bus | Type | Vsp | theta | PGi | QGi | PLi | QLi | Qmin | Qmax |
busdata14 =
[ 1     1    1.060   0      0     0     0     0      0      0;
  2     2    1.045   0     40  42.4  21.7  12.7    -40     50;
 3     2    1.010   0      0  23.4  94.2  19.0     0     40;
 4     3    1.0     0      0     0  47.8  -3.9     0      0;
 5     3    1.0     0      0     0   7.6   1.6     0      0;
 6     2    1.070   0      0  12.2  11.2   7.5    -6     24;
 7     3    1.0     0      0     0   0.0   0.0     0      0;
 8     2    1.090   0      0  17.4   0.0   0.0    -6     24;
 9     3    1.0     0      0     0  29.5  16.6     0      0;
 10    3    1.0     0      0     0   9.0   5.8     0      0;
 11    3    1.0     0      0     0   3.5   1.8     0      0;
 12    3    1.0     0      0     0   6.1   1.6     0      0;
 13    3    1.0     0      0     0  13.5   5.8     0      0;
 14    3    1.0     0      0     0  14.9   5.0     0      0;];
```

Dados dos ramos do problema 2.1

```
%              |  From |  To   |   R     |   X     |     B/2  |  X'mer  |
%              |  Bus  | Bus   |  pu     |  pu     |     pu   | TAP (a) |
linedata14 =[1       2       0.01938   0.05917   0.0264         1
             1       5       0.05403   0.22304   0.0246         1
             2       3       0.04699   0.19797   0.0219         1
             2       4       0.05811   0.17632   0.0170         1
             2       5       0.05695   0.17388   0.0173         1
             3       4       0.06701   0.17103   0.0064         1
             4       5       0.01335   0.04211   0.0            1
             4       7       0.0       0.20912   0.0        0.978
             4       9       0.0       0.55618   0.0        0.969
             5       6       0.0       0.25202   0.0        0.932
             6      11       0.09498   0.19890   0.0            1
             6      12       0.12291   0.25581   0.0            1
             6      13       0.06615   0.13027   0.0            1
             7       8       0.0       0.17615   0.0            1
             7       9       0.0       0.11001   0.0            1
             9      10       0.03181   0.08450   0.0            1
             9      14       0.12711   0.27038   0.0            1
            10      11       0.08205   0.19207   0.0            1
            12      13       0.22092   0.19988   0.0            1
            13      14       0.17093   0.34802   0.0            1 ];
```

2.2 Resultado do Estimador de Estados

Os resultados das duas soluções convergiram para esse problema. Nota-se que o tempo computacional utilizando Tableau Esparso é menor, embora a comparação tenha apenas efeito didático já que o processador não é dedicado e pode variar o tempo de solução.

2.2.1 Equação Normal

Aqui, a matriz G foi calculada como em 1.5. O código está na seção 3.1

```
-------- State Estimation ------------------
---------------------------
| Bus |    V    |  Angle  |
| No  |   pu    |  Degree |
```

```
--------------------------
   1     1.0068      0.0000
   2     0.9899     -5.5265
   3     0.9518    -14.2039
   4     0.9579    -11.4146
   5     0.9615     -9.7583
   6     1.0185    -16.0798
   7     0.9919    -14.7510
   8     1.0287    -14.7500
   9     0.9763    -16.5125
  10     0.9758    -16.7476
  11     0.9932    -16.5397
  12     1.0009    -17.0203
  13     0.9940    -17.0583
  14     0.9647    -17.8967
---------------------------------------------
Tempo computacional =  0.0834 segundos.
```

2.2.2 Tableau Esparso

Aqui, a matriz G não foi calculada. O calculo foi feito como em 1.8. O código está na seção 3.2

```
-------- State Estimation ------------------
-------- Tableau Sparse   ------------------
--------------------------
| Bus |     V    |  Angle  |
| No  |    pu    |  Degree |
--------------------------
   1     1.0068      0.0000
   2     0.9899     -5.5265
   3     0.9518    -14.2039
   4     0.9579    -11.4146
   5     0.9615     -9.7583
   6     1.0185    -16.0798
   7     0.9919    -14.7510
   8     1.0287    -14.7500
   9     0.9763    -16.5125
  10     0.9758    -16.7476
  11     0.9932    -16.5397
```

```
  12     1.0009   -17.0203
  13     0.9940   -17.0583
  14     0.9647   -17.8967
---------------------------------------------
Tempo computacional =  0.0469 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.*

Entrada de dados do estimador de estados.

```
num = 14;
ybus = ybusppg(num);
zdata = zdatas(num); %pneves: Get Measurement data
bpq = bbusppg(num); % Get B data
nbus = max(max(zdata(:,4)),max(zdata(:,5))); % Get number of buses
type = zdata(:,2);
z = zdata(:,3); % Measuement values
fbus = zdata(:,4); % From bus
tbus = zdata(:,5); % To bus
Ri = diag(zdata(:,6)); % Measurement Error
V = ones(nbus,1); % Initialize the bus voltages
del = zeros(nbus,1); % Initialize the bus angles
E = [del(2:end); V];    % State Vector
G = real(ybus);
B = imag(ybus);
```

```
vi = find(type == 1); % Index of voltage magnitude measurements
ppi = find(type == 2); % Index of real power injection measurements
qi = find(type == 3); % Index of reactive power injection measurements
pf = find(type == 4); % Index of real powerflow measurements
qf = find(type == 5); % Index of reactive powerflow measurements
```

Calcula o número de medidas disponíveis.

```
%pneves
nvi = length(vi); % Number of Voltage measurements
npi = length(ppi); % Number of Real Power Injection measurements
nqi = length(qi); % Number of Reactive Power Injection measurements
npf = length(pf); % Number of Real Power Flow measurements
nqf = length(qf); % Number of Reactive Power Flow measurements
iter = 1;
tol = 5;
```

Itera-se comparando com a tolerância desejada.

```
while(tol > 1e-4)
    %Measurement Function, h
    h1 = V(fbus(vi),1);
    h2 = zeros(npi,1);
    h3 = zeros(nqi,1);
    h4 = zeros(npf,1);
    h5 = zeros(nqf,1);
```

```
    for i = 1:npi
        m = fbus(ppi(i));
        for k = 1:nbus
            h2(i) = h2(i) + V(m)*V(k)*(G(m,k)*cos(del(m)-del(k)) + ...
                B(m,k)*sin(del(m)-del(k)));
        end
    end

    for i = 1:nqi
        m = fbus(qi(i));
        for k = 1:nbus
            h3(i) = h3(i) + V(m)*V(k)*(G(m,k)*sin(del(m)-del(k)) - ...
                B(m,k)*cos(del(m)-del(k)));
        end
    end

    for i = 1:npf
        m = fbus(pf(i));
        n = tbus(pf(i));
        h4(i) = -V(m)^2*G(m,n) - V(m)*V(n)*(-G(m,n)*cos(del(m)-del(n))
```

```
            - B(m,n)*sin(del(m)-del(n)));
    end

    for i = 1:nqf
        m = fbus(qf(i));
        n = tbus(qf(i));
        h5(i) = -V(m)^2*(-B(m,n)+bpq(m,n))
            - V(m)*V(n)*(-G(m,n)*sin(del(m)-del(n))
            + B(m,n)*cos(del(m)-del(n)));
    end

    h = [h1; h2; h3; h4; h5];

    %pneves: Residue
    r = z - h;
```

Monta-se a matriz Jacobiana seguindo 1.1.

H_{11} Derivada de V com respeito a θ.

```
    H11 = zeros(nvi,nbus-1);
```

H_{12} Derivada de V com respeito a V.

```
    H12 = zeros(nvi,nbus);
    for k = 1:nvi
        for n = 1:nbus
            if n == k
                H12(k,n) = 1;
            end
        end
    end
```

H_{21} Derivada de P com respeito a θ.

```
    H21 = zeros(npi,nbus-1);
    for i = 1:npi
        m = fbus(ppi(i));
        for k = 1:(nbus-1)
            if k+1 == m
```

```
            for n = 1:nbus
                H21(i,k) = H21(i,k) + ...
                    V(m)* V(n)*(-G(m,n)*sin(del(m)-del(n)) + ...
                    B(m,n)*cos(del(m)-del(n)));
            end
            H21(i,k) = H21(i,k) - V(m)^2*B(m,m);
        else
            H21(i,k) = V(m)* V(k+1)*(G(m,k+1)*sin(del(m)-del(k+1))
            - B(m,k+1)*cos(del(m)-del(k+1)));
        end
    end
end
```

H_{22} Derivada de P com respeito a V.

```
H22 = zeros(npi,nbus);
for i = 1:npi
    m = fbus(ppi(i));
    for k = 1:(nbus)
        if k == m
            for n = 1:nbus
                H22(i,k) = H22(i,k) + ...
                V(n)*(G(m,n)*cos(del(m)-del(n)) + ...
                B(m,n)*sin(del(m)-del(n)));
            end
            H22(i,k) = H22(i,k) + V(m)*G(m,m);
        else
            H22(i,k) = V(m)*(G(m,k)*cos(del(m)-del(k)) + ...
            B(m,k)*sin(del(m)-del(k)));
        end
    end
end
```

H_{31} Derivada de Q com respeito a θ.

```
H31 = zeros(nqi,nbus-1);
for i = 1:nqi
    m = fbus(qi(i));
    for k = 1:(nbus-1)
        if k+1 == m
```

```
                for n = 1:nbus
                    H31(i,k) = H31(i,k) + ...
                    V(m)* V(n)*(G(m,n)*cos(del(m)-del(n)) + ...
                    B(m,n)*sin(del(m)-del(n)));
                end
                H31(i,k) = H31(i,k) - V(m)^2*G(m,m);
            else
               H31(i,k) = V(m)* V(k+1)*(-G(m,k+1)*cos(del(m)-del(k+1))
               - B(m,k+1)*sin(del(m)-del(k+1)));
            end
        end
    end
```

H_{32} Derivada de Q com respeito a V.

```
    H32 = zeros(nqi,nbus);
    for i = 1:nqi
        m = fbus(qi(i));
        for k = 1:(nbus)
            if k == m
                for n = 1:nbus
                    H32(i,k) = H32(i,k) + ...
                    V(n)*(G(m,n)*sin(del(m)-del(n)) - ...
                    B(m,n)*cos(del(m)-del(n)));
                end
                H32(i,k) = H32(i,k) - V(m)*B(m,m);
            else
               H32(i,k) = V(m)*(G(m,k)*sin(del(m)-del(k))
               - B(m,k)*cos(del(m)-del(k)));
            end
        end
    end
```

H_{41} Derivada de P com respeito a θ.

```
    H41 = zeros(npf,nbus-1);
    for i = 1:npf
        m = fbus(pf(i));
        n = tbus(pf(i));
        for k = 1:(nbus-1)
```

```
            if k+1 == m
                H41(i,k) = V(m)* V(n)*(-G(m,n)*sin(del(m)-del(n))
                + B(m,n)*cos(del(m)-del(n)));
            else if k+1 == n
                H41(i,k) = -V(m)* V(n)*(-G(m,n)*sin(del(m)-del(n))
                + B(m,n)*cos(del(m)-del(n)));
                else
                    H41(i,k) = 0;
                end
            end
        end
    end
```

H_{42} Derivada de P com respeito a V.

```
    H42 = zeros(npf,nbus);
    for i = 1:npf
        m = fbus(pf(i));
        n = tbus(pf(i));
        for k = 1:nbus
            if k == m
                H42(i,k) = -V(n)*(-G(m,n)*cos(del(m)-del(n))
                - B(m,n)*sin(del(m)-del(n))) - 2*G(m,n)*V(m);
            else if k == n
                H42(i,k) = -V(m)*(-G(m,n)*cos(del(m)-del(n))
                - B(m,n)*sin(del(m)-del(n)));
                else
                    H42(i,k) = 0;
                end
            end
        end
    end
```

H_{51} Derivada de Q com respeito a θ.

```
    H51 = zeros(nqf,nbus-1);
    for i = 1:nqf
        m = fbus(qf(i));
        n = tbus(qf(i));
        for k = 1:(nbus-1)
```

```
            if k+1 == m
                H51(i,k) = -V(m)* V(n)*(-G(m,n)*cos(del(m)-del(n))
                - B(m,n)*sin(del(m)-del(n)));
            else if k+1 == n
                H51(i,k) = V(m)* V(n)*(-G(m,n)*cos(del(m)-del(n))
                - B(m,n)*sin(del(m)-del(n)));
                else
                    H51(i,k) = 0;
                end
            end
        end
    end
```

H_{52} Derivada de Q com respeito a V.

```
    H52 = zeros(nqf,nbus);
    for i = 1:nqf
        m = fbus(qf(i));
        n = tbus(qf(i));
        for k = 1:nbus
            if k == m
                H52(i,k) = -V(n)*(-G(m,n)*sin(del(m)-del(n))
                + B(m,n)*cos(del(m)-del(n)))
                - 2*V(m)*(-B(m,n)+ bpq(m,n));
            else if k == n
                H52(i,k) = -V(m)*(-G(m,n)*sin(del(m)-del(n))
                + B(m,n)*cos(del(m)-del(n)));
                else
                    H52(i,k) = 0;
                end
            end
        end
    end
```

Matriz Jacobiana de medições é montada

```
    H = [H11 H12; H21 H22; H31 H32; H41 H42; H51 H52];
```

3.1 Equação normal

```
    % pneves: Gain Matrix, Gm
    Gm = H'*inv(Ri)*H;

    %pneves:  Objective Function
    J = sum(inv(Ri)*r.^2);

    %pneves:  State Vector
    dE = inv(Gm)*(H'*inv(Ri)*r);
    E = E + dE;
    del(2:end) = E(1:nbus-1);
    V = E(nbus:end);
    iter = iter + 1;
    tol = max(abs(dE));
end
```

3.2 Tableau esparso

```
%pneves: Tableau sparse
    if iter == 1
        [linhasH , colunasH] = size(H);
        HZeros = zeros(colunasH,colunasH);
    end
    MatrizH = [Ri H; H' HZeros];
    if iter == 1
        [linhasLambda , colunasLambda] = size(MatrizH);
        VetorLambda = zeros(linhasH,1);
        dE = zeros(colunasH,1);
    end
    MatrizLambda = [VetorLambda ; dE];
    Rzeros = zeros(colunasH,1);
    MatrizR = [r ; Rzeros];
    MatrizLambda = inv(MatrizH)*MatrizR;
    dE = MatrizLambda((linhasH+1):end);
    %dE = MatrizLambda(42:end);
    E = E + dE;
    del(2:end) = E(1:nbus-1);
```

```
    V = E(nbus:end);
    iter = iter + 1;
    tol = max(abs(dE));
end
```

3.3 Resultados

```
CvE = diag(inv(H'*inv(Ri)*H)); % Covariance matrix
Del = 180/pi*del;
E2 = [V Del]; % Bus Voltages and angles
disp('-------- State Estimation ------------------');
disp('--------------------------');
disp('| Bus |    V    |  Angle  | ');
disp('| No  |   pu    |  Degree | ');
disp('--------------------------');
for m = 1:nbus
    fprintf('%4g', m);
    fprintf('  %8.4f', V(m));
    fprintf('   %8.4f', Del(m));
    fprintf('\n');
end
disp('---------------------------------------------');
```

Capítulo 4

Discussões e análise de resultados

4.1 Composição do trabalho

Neste trabalho, foi abordado uma introdução à teoria de Estimadores de Estado com métodos da Equação Normal e Tableau Esparso.
A rede 14 barras IEEE foi escolhida para testar convergencia dos dois métodos, apresentados nas seções 3.1 e 3.2. A partir de medidas da rede, como na seção 2.1.1, foi possivel estimar o estado atual da rede 14 barras IEEE, como mostrado em 2.2.

4.2 Performance

Os dois métodos, explicados nas seções 3.1 e 3.2, convergiram para a mesma solução, que pode ser verificado em 2.2. O método Tableau Esparso, com leve vantagem computacional, por evitar o calculo da matriz G, como em 1.5, aliviando o método de calculo.
A economia se deve, também, ao fato da matriz gerada ser bastante esparsa.(NETO, Consultado em 07/2020).

Referências bibliográficas

CASTRO, C. A. **Cálculo de Fluxo de Carga Linearizado**. Campinas-SP: [s.n.], Consultado em 07/2020. Disponível em: <http://www.dsee.fee.unicamp.br/~ccastro>.

MONTICELLI, A. State Estimation in Electric Power Systems, 1999. DOI: 10.1007/978-1-4615-4999-4.

NETO, M. S. I. **Estimação de estado para Redes de Distribuição de Energia Elétrica Avançadas**. São Carlos-SP: [s.n.], Consultado em 07/2020. Disponível em: <https://www.teses.usp.br/teses/disponiveis/18/18154/tde-02082017-163837/publico/Mohamad.pdf>.