

## Sistema MATLAB para Resolução de Problemas de Trânsito de Fluxo de Potência

---



Licenciatura em Engenharia Eletrotécnica – Sistemas Elétricos de Energia | ANSEE | Ano letivo 2018 / 2019

Professor: Manuel João Gonçalves

Alexandre Amorim (11161497)

## Índice Geral

<b>1. Introdução .....</b>	<b>3</b>
<b>2. Manual de Utilizador .....</b>	<b>8</b>
<b>4. Conclusão .....</b>	<b>23</b>

## 1. Introdução

---

O objetivo essencial de um Sistema Elétrico de Energia (SEE) é a satisfação das necessidades de energia elétrica dos respectivos consumidores.

Os estudos de trânsito de fluxos de potência (também designados por fluxos de cargas) são de grande importância no planeamento e na exploração dos SEE. A informação mais importante, obtida destes estudos, são as tensões nos barramentos e o trânsito de potências ativa e reativa nas linhas.

Para os cálculos de trânsito de fluxos de potências considera-se a potência injetada ( $P_{inj}$ ) num barramento, que é calculada como a diferença entre a potência produzida ( $P_G$ ) e a potência consumida no barramento ( $P_C$ ).

$$P_{inj} = P_G - P_C \Leftrightarrow = P_i + jQ_i = (P_i^G - P_i^C) + j(Q_i^G - Q_i^C) \quad (1)$$

Os diferentes tipos de barramentos que podem existir numa rede elétrica são:

- › Barramentos PQ – São barramentos em que são conhecidos os valores de potência ativa e reativa injetada. Correspondem a barramentos em que as cargas são conhecidas. Como resultado dos cálculos dos trânsitos de potências, calcula-se a tensão em módulo e fase;
- › Barramentos PV – São barramentos para os quais o módulo da tensão é conhecido e se conhece também a potência ativa injetada. O valor da potência reativa injetada é obtido assim como o ângulo da tensão, como resultado dos estudos de fluxo de cargas
- › Barramento de referência dos argumentos - É um tipo de barramento fictício que é considerado para resolver o problema do fluxo de cargas. Qualquer barramento PV pode ser escolhido como referência para o argumento das tensões, normalmente considera-se um argumento da tensão nulo, passando a conhecer-se a tensão em módulo e fase. Atribui-se a este barramento a função de compensação ou balanço das potências ativas (com a exceção do Modelo DC, porque  $R \ll X \Rightarrow R \sim 0$ , logo não existem perdas ativas, não havendo por isso necessidade de as compensar).

Existem vários métodos para realizar estudos de fluxo de cargas:

- Método DC (posteriormente designado apenas DC) - caracterizado pela sua simples aplicação, devido às suas simplificações, isto é, os módulos das tensões são constantes e iguais a 1 pu em todos os barramentos, as resistências dos componentes são desprezadas

e o trânsito da energia reativa não é considerado. É o método mais rápido de todos e é utilizado como solução inicial para outros métodos.

- ›  $R \ll X \Rightarrow R \sim 0$ , logo não existem perdas ativas
- ›  $Y_{sh} \sim 0, Q_{ic} = 0$
- ›  $V_i = 1 \text{ p.u.}$
- ›  $\theta_{ik}$  pequenos,  $\sin \theta_{ik} = \theta_{ik} = \theta_i - \theta_k$  e  $\cos \theta_{ik} = 1$

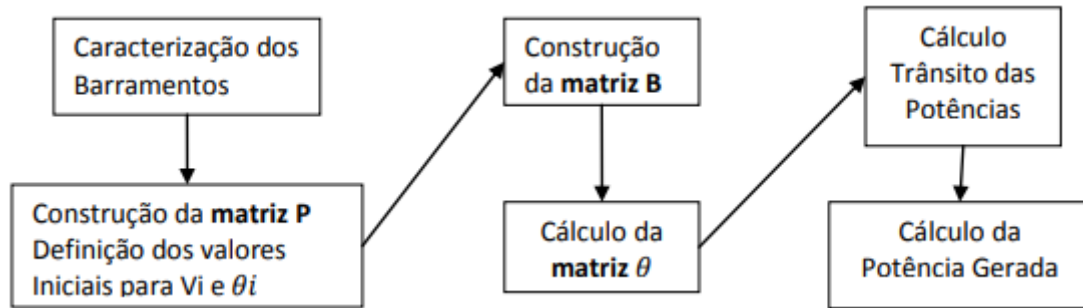


Figura 1 – Fluxograma para o cálculo do trânsito de energia pelo método DC

- Gauss Seidel (posteriormente designado GS) - é um método iterativo de fácil programação, bem comportado e permite a utilização de fatores de aceleração. As iterações aumentam bastante com o aumento do número de barramentos (especialmente PV), tornando-se desadequado para redes com elevado número de barramentos de produção.

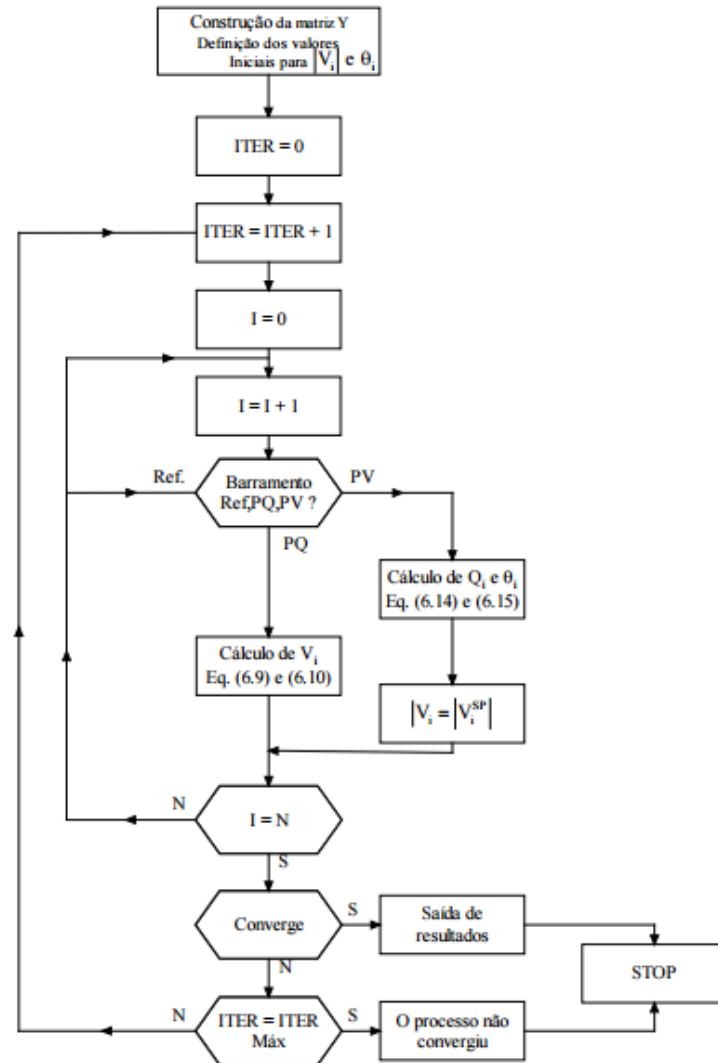


Figura 2 - Fluxograma para o cálculo do trânsito de energia pelo método de Gauss-Seidel

- Método Newton Raphson (posteriormente designado NR) - assim como o anterior, trata-se de um método iterativo, sendo o com menor número de iterações de todos, pesado computacionalmente, mais completo aplica-se a outros tipos de estudos e cujo tempo de iteração é elevado. Neste caso o aumento do número de barramentos PV implica um menor aumento de iterações. O Método de Newton-Raphson é o mais completo e abrangente método, permitindo fazer outro tipo de estudos.

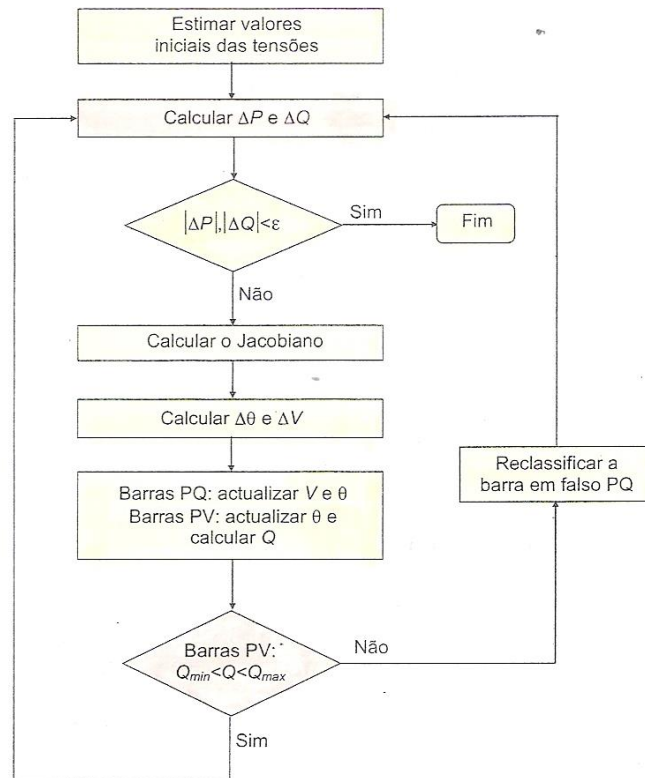


Figura 3 - Fluxograma para o cálculo do trânsito de energia pelo método de Newton-Raphson

- Método Rápido Desacoplado (posteriormente designado FDL) - é um método simplificado do método NR, sendo mais rápido e menos pesado. Precisa de um maior número de iterações (menor que o GS), mas o tempo necessário para cada uma é menor, comparativamente com o método anterior. Também é caracterizado por ser necessário calcular a matriz Jacobiana, apenas na primeira iteração.

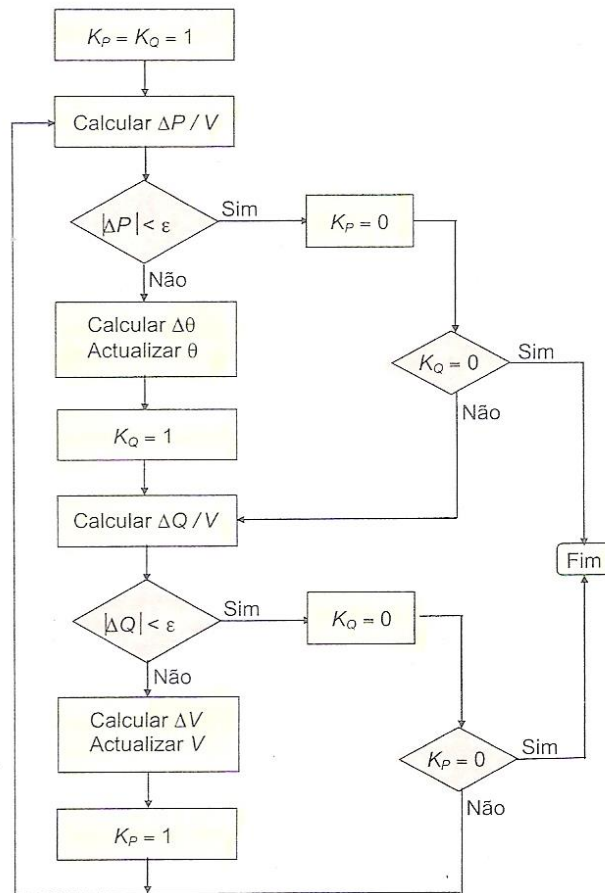


Figura 4 - Fluxograma para o cálculo do trânsito de energia pelo método do desacoplamento

## 2. Manual de Utilizador

---

Esta secção do documento descritivo, que acompanha a aplicação desenvolvida em Matlab, visa demonstrar de forma sucinta o funcionamento da aplicação.

O programa criado permite analisar qualquer rede, desde que obedeça aos seguintes critérios:

- ✓ A rede deve estar organizada pela seguinte ordem: dados das linhas, dados das cargas, dados da produção e tipo de barramentos;
- ✓ O tipo de barramento deve ser introduzido por ordem crescente e apresentado da seguinte forma: tipo['Q', 'V', 'R'], isto é, "Q" corresponde ao barramento PQ, "V" corresponde barramento PV e "R" corresponde ao barramento de referência dos argumentos (sendo de compensação em todos os métodos numéricos, exceto no método DC).
- ✓ Os dados da produção deverão conter o número do barramento a que se refere, o valor da potência gerada e o valor da tensão do respetivo;
- ✓ Os dados das cargas deverão conter o número do barramento, o valor da potência ativa consumida e o valor da potência reativa consumida;
- ✓ Os dados das linhas deverão conter o número do barramento de origem, o número do barramento de destino, o valor da resistência, o valor reatância e o valor da impedância *shunt*;
- ✓ Todos os valores da rede deverão ser inseridos em unidades pu;

Definida a rede, basta clicar em *Run* e analisar os resultados.

De seguida apresentamos o código do programa onde consta uma breve explicação do que o código executa.

Posteriormente a clicarmos no *Run*, é nos apresentado o menu onde podemos introduzir o método que pretendemos.



The screenshot shows the MATLAB Editor with a script file named 'TrabalhoANSEE\_1161497\_1111720.m'. The script contains a menu for selecting a method. The Command Window shows the output of the script, which lists four options: 1 -> Para Modelo DC, 2 -> Para Método de Gauss-Seidel, 3 -> Para Método de Newton-Raphson, and 4 -> Para Método Rápido Desacoplado. The prompt 'Introduza o método pretendido:' is also visible.

```

39 % Menu para escolha do método pretendido
40 % Mostra os diferentes métodos que podem ser seleccionados
41
42 disp('Escolha o método pretendido')
43 disp(' ')
44 disp('1 -> Para Modelo DC')
45 disp('2 -> Para Método de Gauss-Seidel')
46 disp('3 -> Para Método de Newton-Raphson')
47 disp('4 -> Para Método Rápido Desacoplado')
48 disp(' ')
49 metodo=input('Introduza o método pretendido: '); % Escolha do método a executar
    
```

Como prevenção implementamos um pequeno teste, a fim de verificar se o valor introduzido pelo utilizador se encontra dentro das opções que são apresentadas. No caso do teste invalidar esse valor, é apresentada a seguinte mensagem no *Command Window* ao utilizador:

The screenshot shows the MATLAB Editor with a script file named 'TrabalhoANSEE\_1161497\_1111720.m'. The script contains a conditional statement that checks if the input method is less than 1 or greater than 4. If the input is invalid, it displays an error message: 'O número que introduziu é inválido. Por favor, tente novamente.' The Command Window shows the output of the script, which lists the same four options as before, and the prompt 'Introduza o método pretendido:'.

```

50
51 %Eliminar possíveis erros de escolha do utilizador
52
53 if metodo<1 || metodo>4
54     clc
55     disp('O número que introduziu é inválido.') % Erro em caso de uma entrada inválida
56     disp('Por favor, tente novamente.')
57     disp(' ')
58     disp('1 -> Para Modelo DC')
59     disp('2 -> Para Método de Gauss-Seidel')
60     disp('3 -> Para Método de Newton-Raphson')
61     disp('4 -> Para Método de Rápido Desacoplado')
62     disp(' ')
63     metodo=input('Introduza o método pretendido: ');
64 else
65     clc
66 end
67
68
    
```

Outro aspeto comum a todos os métodos, é que é necessário efetuar operações matemáticas iniciais, sendo estes descritos na seguinte figura:

The screenshot shows the MATLAB Editor with a script file named 'TrabalhoANSEE\_1161497\_1111720.m'. The script contains a series of initial calculations for the power flow problem. The calculations involve defining variables for the number of lines, the number of buses, the resistances, the reactances, the shunt admittances, the active power consumed by the loads, the active power generated by the generators, the reactive power consumed by the loads, and the reactive power generated by the generators. The calculations are performed using matrix operations and the 'j' symbol for the imaginary unit.

```

69 %% Cálculos iniciais
70
71 Origem=linhas(:,1); %Nó de Origem
72 Destino=linhas(:,2); %Nó de Destino
73 R=linhas(:,3); %Resistências das linhas
74 X=linhas(:,4); %Reatâncias das linhas
75 Ysh=(linhas(:,5))*1j; %Admitâncias shunt das linhas
76 Barramento=1:length(tipo); %Id dos barramentos
77 Pc=cargas(:,2); %Potência ativa consumida pelas cargas
78 Pg=zeros(length(tipo),1); %Potência ativa gerada no barramento
79 Pl=prod(:,1); %Potência ativa gerada no barramento
80 Z=R+j*X; %Impedâncias das linhas
81 Qc=cargas(:,3); %Potência Reativa consumidas pelas cargas
82 Qg=zeros(length(tipo),1); %Potência Reativa gerada no barramento
83 tQc=Qc'; %Potência Reativa consumida no barramento
84
    
```

Nota: O símbolo “j” desempenha a função de transpor um vetor ou uma matriz.

De forma a integrar todos os métodos no mesmo ficheiro optamos por recorrer ao comando *switch*, sendo que os seus respetivos cases se distribuem da seguinte forma:

Case 1 -> Método DC

Case 2 -> Método GS

Case 3 -> Método NR

Case 4 -> Método RD

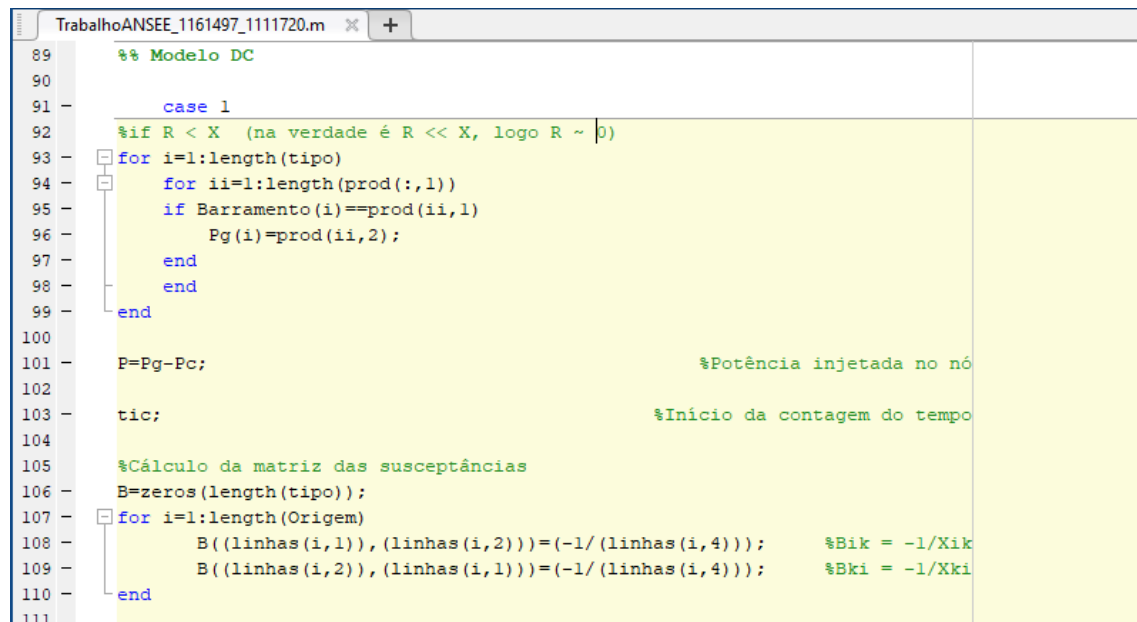
Logicamente, o primeiro método a ser explicado é o **método DC**.

Iniciamos por calcular a potência injetada nos barramentos. Sendo que a equação que nos permite determinar esses valores é a equação (1).

De seguida calculamos a matriz das susceptâncias (B), que é caracterizada pelas seguintes equações:

$$B_{ik} = \frac{-1}{X_{ik}}; B_{ii} = \sum_{k=1 \neq i}^N \frac{-1}{X_{ik}} \quad (2)$$

Sendo que para efeito de cálculos iremos eliminar os valores inerentes ao barramento de referência, tanto os valores nas colunas como nas linhas.



```

89 %% Modelo DC
90
91 case 1
92     %if R < X (na verdade é R << X, logo R ~ 0)
93     for i=1:length(tipo)
94         for ii=1:length(prod(:,1))
95             if Barramento(i)==prod(ii,1)
96                 Pg(i)=prod(ii,2);
97             end
98         end
99     end
100
101     P=Pg-Pc; %Potência injetada no nó
102
103     tic; %Início da contagem do tempo
104
105     %Cálculo da matriz das susceptâncias
106     B=zeros(length(tipo));
107     for i=1:length(Origem)
108         B((linhas(i,1)),(linhas(i,2)))=(-1/(linhas(i,4))); %Bik = -1/Xik
109         B((linhas(i,2)),(linhas(i,1)))=(-1/(linhas(i,4))); %Bki = -1/Xki
110     end
111

```

```

104
105 %Cálculo da matriz das susceptâncias
106 B=zeros(length(tipo));
107 for i=1:length(Origem)
108     B((linhas(i,1)),(linhas(i,2)))=(-1/(linhas(i,4))); %Bik = -1/Xik
109     B((linhas(i,2)),(linhas(i,1)))=(-1/(linhas(i,4))); %Bki = -1/Xki
110 end
111
112 for i=1:length(tipo)
113     h=0;
114     for ii=1:length(tipo)
115         h=h+abs(B((i),(ii))); %Bii= S(-1/Xik).-1
116     end
117     B((i),(i))=h;
118 end
119 for i=1:length(tipo)
120     if tipo(i)=='R' %Se for o BarRef eliminamos a sua linha e coluna
121         B(i,:)=[];
122         B(:,i)=[];
123         P(i,:)=[];
124         BarramentoRef=i;
125     end
126 end
127
128 t=(B^-1)*P; %Vetor coluna dos ângulos das tensões
129

```

O vetor coluna dos ângulos das tensões é calculado através da seguinte equação matricial:

$$[\theta] = [B]^{-1} \times [P] \quad (3)$$

```

129
130 for i=1:length(tipo)
131     if i<BarramentoRef %Se o Id do barramento for inferior ao do BarRef
132         teta(i,1)=t(i);
133     elseif i==BarramentoRef %Se o Id do barramento for igual ao do BarRef
134         teta(i,1)=0;
135     else
136         teta(i,1)=t(i-1); %Se o Id do barramento for superior ao do BarRef
137     end
138 end
139
140 for i=1:length(Origem)
141     FluxoPot(i,1)=(teta(Origem(i))-teta(Destino(i)))/(X(i));
142 end
143
144 Pij(1,:)=Origem; %Nó do Origem
145 Pij(2,:)=Destino; %Nó de Destino
146 Pij(3,:)=FluxoPot;
147
148 Time=toc; %Fim do tempo de contagem
149

```

Por último calculámos o fluxo das potências nas linhas através da seguinte equação:

$$P_{ij} = \frac{\theta_i - \theta_j}{x_{ij}} \quad (4)$$

O resultado final deste algoritmo é apresentado da seguinte forma:

```

150 - disp('-----Modelo DC-----')
151 - disp(' ')
152 - display(['Tempo ', num2str(Time), ' segundos']);
153 - disp(' ')
154 - for i=1:length(tipo)
155 -     disp(['teta ', num2str(i), ' = ', num2str(teta(i), '%.4f')]);
156 - end
157 - disp(' ')
158 - fprintf('Pij =\n\n')
159 - fprintf(' %d %d %.4f\n', Pij)
160
161 %% Método Gauss-Seidel
162
163 case 2

```

Command Window Output:

```

-----Modelo DC-----
Tempo 0.0049622 segundos

teta 1 = -0.0575
teta 2 = -0.0450
teta 3 = 0.0000

Pij =

     1     2    -0.1250
     1     3    -0.5750
     2     3    -0.2250

```

O método numérico seguinte é o **método de Gauss-Seidel**. Primeiramente calculamos a matriz das admitâncias nodais (Y), através das seguintes equações:

$$Y_{ik} = \frac{-1}{Z_{ik}}; Y_{ii} = \sum_{k=1 \neq i}^N \frac{-1}{Z_{ik}} + \sum Y_{sh} \quad (5)$$

Além disso, é necessário definir as soluções iniciais, como o número de iterações máximo e o erro máximo relativo aos desvios das potências.

```

161 %% Método Gauss-Seidel
162
163 case 2
164
165 for i=1:length(tipo)
166     for ii=1:length(prod(:,1))
167         if Barramento(i)==prod(ii,1)
168             Pg(ii)=prod(ii,2);
169         end
170     end
171 end
172
173 % Cálculo da matriz y
174 Y=zeros(length(tipo));
175 for i=1:length(Origem)
176     Y(Origem(i),Destino(i))=-(1/Z(i)); %Yik = -1/Zik
177     Y(Destino(i),Origem(i))=-(1/Z(i)); %Yki = -1/Zki
178     Y(Origem(i),Origem(i))=Y(Origem(i),Origem(i))+ 1/Z(i)+Ysh(i)/2;%Yii = S(-1/Zik)+Ysh(ik)
179     Y(Destino(i),Destino(i))=Y(Destino(i),Destino(i))+ 1/Z(i)+Ysh(i)/2;%Ykk = S(-1/Zki)+Ysh
180 end
181
182 %Solução Inicial
183 itermax=200;
184 erromax=0.01; %Erro máximo definido para os desvios de potência
185 U=zeros(length(tipo),1);
186 U(P1(:))=prod(:,3);

```

O método seguinte é o **método de Newton-Raphson**. Similar ao método de Gauss-Seidel é necessário calcular a matriz Y e parametrizar a solução inicial. Posteriormente, calculamos as potências injetadas através da equação (1).

```

329
330 %% Método Newton-Raphson
331
332 case 3
333
334 for i=1:length(tipo)
335     for ii=1:length(prod(:,1))
336         if bar(i)==prod(ii,1)
337             Pg(i)=prod(ii,2);
338         end
339     end
340 end
341 %-----
342
343 %Matriz y
344 Y=zeros(length(tipo));
345 for i=1:length(Origem)
346     Y(Origem(i),Destino(i))=-(1/Z(i));
347     Y(Destino(i),Origem(i))=-(1/Z(i));
348     Y(Origem(i),Origem(i))=Y(Origem(i),Origem(i))+ 1/Z(i)+Ysh(i)/2;
349     Y(Destino(i),Destino(i))=Y(Destino(i),Destino(i))+ 1/Z(i)+Ysh(i)/2;
350 end
351

```

```

351
352 %Solução inicial
353 itermax=100;
354 erromax=0.01;
355 U=zeros(length(tipo),1);
356 U(P1(:))=prod(:,3);
357 Q=zeros(length(tipo),1);
358 P=zeros(length(tipo),1);
359
360 %Calculo das potências injetadas nos barramentos PV e PQ
361 for i=1:length(tipo)
362     if tipo(i)=='Q'
363         U(i)=1;
364         P(i,1)=Pg(i)-Pc(i);
365         Q(i,1)=Qg(i)-Qc(i);
366     elseif tipo(i)=='V'
367         P(i,1)=Pg(i)-Pc(i);
368     end
369 end
370 Uini=U;
371 thetaini=zeros(length(Uini),1);
372
373 disp('Método Newton-Raphson');
374 tic;
375 flag=1;
376 niter=0;

```

Na figura abaixo verificamos algumas diferenças em relação ao método anterior como os vetores A, B e C. O elemento do vetor A é igual ao número de barramentos PV existente na rede, o elemento do vetor B é igual ao número de barramentos PQ existente na rede e por fim o vetor C é a agregação dos vetores A e B ordenados ascendentemente. Esta ordenação será fulcral para operações matemáticas *a posteriori*.

De seguida, calculamos  $I_{cal}$ , para determinar  $S_{cal}$  e por fim obter  $P_{cal}$  e  $Q_{cal}$ .

Com estes, valores podemos determinar os desvios  $\Delta P$  e  $\Delta Q$ .

```

TrabalhoANSEE_1161497_1111720.m
372 -
373 - disp('-----Método Newton-Raphson-----');
374 - tic;
375 - flag=1;
376 - niter=0;
377 - A=find(tipo=='V');
378 - B=find(tipo=='Q');
379 - C(:,1)=[A,B];
380 - C=sort(C); %ordena ascendentemente
381 - %Cálculo das potencias para barramentos FV
382 - Ical=zeros(length(tipo),1);
383 - for i=1:length(tipo)
384 -     if tipo(i)=='V'
385 -         for ii=1:length(tipo)
386 -             Ical(i)=Ical(i)+(Y(i,ii)*Uini(ii));
387 -         end
388 -         Scal(i)=(Uini(i)*conj(Ical(i)));
389 -         Pcal(i,1)=real(Scal(i));
390 -         Qcal(i,1)=imag(Scal(i));
391 -     end
392 - end
393 - %Cálculo das potencias para barramentos PQ
394 - for i=1:length(tipo)
395 -     if tipo(i)=='Q'
396 -         for ii=1:length(tipo)
397 -             Ical(i)=Ical(i)+(Y(i,ii)*Uini(ii));
398 -         end

```

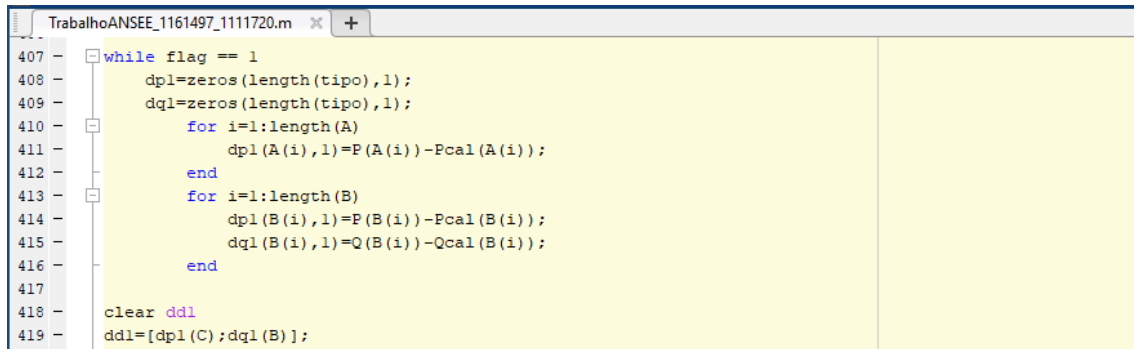
```

TrabalhoANSEE_1161497_1111720.m
392 - end
393 - %Cálculo das potencias para barramentos PQ
394 - for i=1:length(tipo)
395 -     if tipo(i)=='Q'
396 -         for ii=1:length(tipo)
397 -             Ical(i)=Ical(i)+(Y(i,ii)*Uini(ii));
398 -         end
399 -         Scal(i)=(Uini(i)*conj(Ical(i)));
400 -         Pcal(i,1)=real(Scal(i));
401 -         Qcal(i,1)=imag(Scal(i));
402 -     end
403 - end
404 -
405 - v=abs(Uini);
406 -
407 - while flag == 1
408 -     dpl=zeros(length(tipo),1);
409 -     dql=zeros(length(tipo),1);
410 -     for i=1:length(A)
411 -         dpl(A(i),1)=P(A(i))-Pcal(A(i));
412 -     end
413 -     for i=1:length(B)
414 -         dpl(B(i),1)=P(B(i))-Pcal(B(i));
415 -         dql(B(i),1)=Q(B(i))-Qcal(B(i));
416 -     end
417 -
418 -     clear dd1

```

Após obtermos os desvios  $\Delta P$  e  $\Delta Q$ , prosseguimos para o preenchimento do nosso vetor  $dd1$ , que pode ser em termos genéricos é caracterizado da seguinte forma:

$$dd1 = \begin{bmatrix} \Delta P_i \\ \Delta Q_i \end{bmatrix} \quad (6)$$



```

TrabalhoANSEE_1161497_1111720.m
407 - while flag == 1
408 -     dpl=zeros(length(tipo),1);
409 -     dq1=zeros(length(tipo),1);
410 -     for i=1:length(A)
411 -         dpl(A(i),1)=P(A(i))-Pcal(A(i));
412 -     end
413 -     for i=1:length(B)
414 -         dpl(B(i),1)=P(B(i))-Pcal(B(i));
415 -         dq1(B(i),1)=Q(B(i))-Qcal(B(i));
416 -     end
417 -
418 -     clear ddl
419 -     ddl=[dpl(C);dq1(B)];

```

O passo seguinte é determinar a matriz Jacobiana (Y), porém no método Newton-Raphson, esta matriz é subdividida em quatro submatrizes (H, M, N, L). Assim sendo, é necessário determinar cada uma destas submatrizes, sendo que é necessário salvasguardar que as submatrizes H e L dependem do número de barramento PQ existentes na rede e as submatrizes M e N dependem do número de barramento PV existentes na rede.

De uma forma geral, podemos saber de antemão que as dimensões das submatrizes respeitarão a seguinte caracterização:

$$H \begin{cases} \text{número de linhas} = \text{número de barramentos} - 1 \\ \text{número de colunas} = \text{número de barramentos} - 1 \end{cases}$$

$$N \begin{cases} \text{número de linhas} = \text{número de barramentos} - 1 \\ \text{número de colunas} = \text{número de barramentos PQ} \end{cases}$$

$$M \begin{cases} \text{número de linhas} = \text{número de barramentos PQ} \\ \text{número de colunas} = \text{número de barramentos} - 1 \end{cases}$$

$$L \begin{cases} \text{número de linhas} = \text{número de barramentos PQ} \\ \text{número de colunas} = \text{número de barramentos PQ} \end{cases}$$

Ao longo do algoritmo em que se determina as submatrizes irá se encontrar a seguinte linha de código:

```
if C(i)~=A
```

Isto pois, se o id do elemento do vetor C não corresponder a um barramento do tipo PV (A), não necessitamos de realizar os cálculos com o Qcal, basta trabalhar com o Q inicial.

Inicialmente iremos começar por determinar a submatriz H (a ordem é indiferente).

As operações matemáticas necessárias para calcular os elementos da submatriz H são:

$$H_{ii} = -Q_i - B_{ii} \times V_i^2; H_{ik} = L_{ik} = V_i V_k [G_{ik} \sin \theta_{ik} - B_{ik} \cos \theta_{ik}] \quad (7)$$

```

TrabalhoANSEE_1161497_1111720.m
420 %Preenchimento da matriz Jacobiana
421 H=zeros(length(tipo)-1);
422 M=zeros(length(tipo)-length(prod(:,1)),length(tipo)-1);
423 N=zeros(length(tipo)-1,length(tipo)-length(prod(:,1)));
424 L=zeros(length(tipo)-length(prod(:,1)),length(tipo)-length(prod(:,1)));
425
426 for i=1:length(H(:,1))
427     for ii=1:length(H(1,:))
428         if i==ii
429             if C(i)~=A
430                 H(i,ii)=-Q(C(i),1)-imag(Y(C(i),C(ii))) * v(C(i))^2;
431             else
432                 H(i,ii)=-Qcal(C(i),1)-imag(Y(C(i),C(ii))) * v(C(i))^2;
433             end
434         else
435             H(i,ii)=v(C(i)) * v(C(ii)) * (real(Y(C(i),C(ii))) * sin(tetaini(C(i))-tetaini(C(ii)))-imag(Y(C(i),C(ii))) * cos(tetaini(C(i))-tetaini(C(ii))));
436         end
437     end
438 end
439

```

Seguida da submatriz L:

$$L_{ii} = Q_i - B_{ii} \times V_i^2; L_{ik} = V_i V_k [G_{ik} \sin \theta_{ik} - B_{ik} \cos \theta_{ik}] \quad (8)$$

```

TrabalhoANSEE_1161497_1111720.m
440 for i=1:length(L(:,1))
441     for ii=1:length(L(1,:))
442         if i==ii
443             if C(i)~=A
444                 L(i,ii)=Q(C(i),1)-imag(Y(C(i),C(ii))) * v(C(i))^2;
445             else
446                 L(i,ii)=Qcal(C(i),1)-imag(Y(C(i),C(ii))) * v(C(i))^2;
447             end
448         else
449             L(i,ii)=v(C(i)) * v(C(ii)) * (real(Y(C(i),C(ii))) * sin(tetaini(C(i))-tetaini(C(ii)))-imag(Y(C(i),C(ii))) * cos(tetaini(C(i))-tetaini(C(ii))));
450         end
451     end
452 end
453

```

Posteriormente da submatriz N:

$$N_{ii} = P_i + G_{ii} \times V_i^2; N_{ik} = -M_{ik} = V_i V_k [G_{ik} \cos \theta_{ik} + B_{ik} \sin \theta_{ik}] \quad (9)$$

```

TrabalhoANSEE_1161497_1111720.m
454 for i=1:length(N(:,1))
455     for ii=1:length(N(1,:))
456         if i==ii
457             N(i,ii)=P(C(i),1)+real(Y(C(i),C(ii))) * v(C(i))^2;
458         else
459             N(i,ii)=v(C(i)) * v(C(ii)) * (real(Y(C(i),C(ii))) * cos(tetaini(C(i))-tetaini(C(ii)))+imag(Y(C(i),C(ii))) * sin(tetaini(C(i))-tetaini(C(ii))));
460         end
461     end
462 end
463

```

Por fim a submatriz M:

$$M_{ii} = P_i - G_{ii} \times V_i^2; M_{ik} = -V_i V_k [G_{ik} \cos \theta_{ik} + B_{ik} \sin \theta_{ik}] \quad (10)$$

```

464 for i=1:length(M(:,1))
465     for ii=1:length(M(1,:))
466         if i==ii
467             M(i,ii)=P(C(i),1)-real(Y(C(i),C(ii))) * v(C(i))^2;
468         else
469             M(i,ii)=-v(C(i)) * v(C(ii)) * (real(Y(C(i),C(ii))) * cos(tetaini(C(i))-tetaini(C(ii)))+imag(Y(C(i),C(ii))) * sin(tetaini(C(i))-tetaini(C(ii))));
470         end
471     end
472 end
473

```



Obtendo as quatro submatrizes, basta agregá-las e obtemos a matriz Jacobiana (Y):

$$Y = \begin{bmatrix} H & N \\ M & L \end{bmatrix} \quad (19)$$

Tendo a matriz Jacobiana e os desvios das potências, podemos finalmente obter o vetor coluna com as variações do módulo das tensões e dos seus respectivos argumentos.

$$\begin{bmatrix} \Delta\theta_i \\ \frac{\Delta V_i}{V_i} \end{bmatrix} = \begin{bmatrix} H & N \\ M & L \end{bmatrix}^{-1} \begin{bmatrix} \Delta P_i \\ \Delta Q_i \end{bmatrix} \quad (20)$$

Obtendo  $\Delta\theta_i$  e  $\frac{\Delta V_i}{V_i}$ , adicionamos aos valores anteriores para obter os novos valores calculados, ou seja, realizamos as seguintes operações:

$$V(i)_{novo} = \left| V(i)_{antigo} + \frac{\Delta V_i}{V_i} \right| \quad \angle \theta(i)_{antigo} + \Delta\theta_i \quad (22)$$

```

TrabalhoANSEE_1161497_1111720.m
474 - clear J1 J2 J t
475 - J1=[H;M];
476 - J2=[N;L];
477 - J=[H,N;M,L];
478 - t=inv(J)*ddl;
479 - v=abs(Uini);
480 -
481 - for i=1:length(B)
482 -     v(B(i))=v(B(i))+t(length(C)+1);
483 - end
484 -
485 - for i=1:length(C)
486 -     tetaini(C(i))=tetaini(C(i))+t(i);
487 - end
488 - [X,Z] = pol2cart(tetaini,v);
489 - Uini=X+ii*Z;

```

Para verificar se a solução convergiu, necessitamos alguns passos. Sendo o primeiro o cálculo de Ical, com os novos valores das tensões. De seguida, calculamos Scal, que nos permite obter Pcal e Qcal.

```

TrabalhoANSEE_1161497_1111720.m
488 - [X,Z] = pol2cart(tetaini,v);
489 - Uini=X+ii*Z;
490 - Ical=zeros(length(tipo),1);
491 - clear Scal Pcal Qcal
492 - for i=1:length(tipo)
493 -     if tipo(i)=='V'
494 -         for ii=1:length(tipo)
495 -             Ical(i)=Ical(i)+(Y(i,ii)*Uini(ii));
496 -         end
497 -         Scal(i)=(Uini(i)*conj(Ical(i)));
498 -         Pcal(i,1)=real(Scal(i));
499 -         Qcal(i,1)=imag(Scal(i));
500 -     end
501 - end
502 - for i=1:length(tipo)
503 -     if tipo(i)=='Q'
504 -         for ii=1:length(tipo)
505 -             Ical(i)=Ical(i)+(Y(i,ii)*Uini(ii));
506 -         end
507 -         Scal(i)=(Uini(i)*conj(Ical(i)));
508 -         Pcal(i,1)=real(Scal(i));
509 -         Qcal(i,1)=imag(Scal(i));
510 -     end
511 - end

```

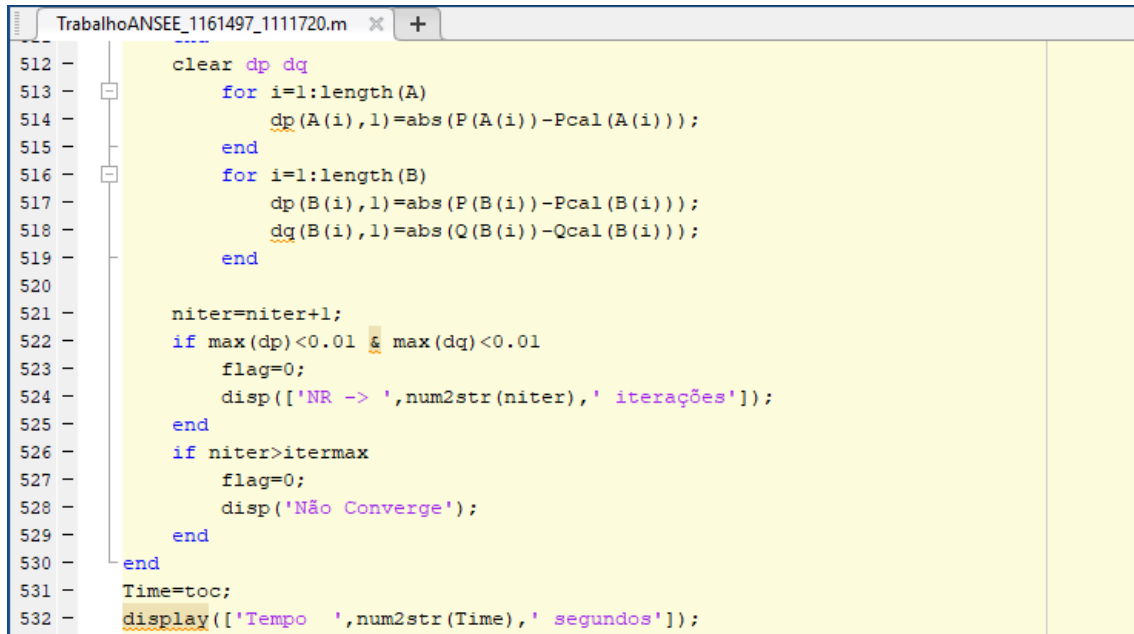
Para verificar se convergiu basta realizar o seguinte teste:

$$\Delta P = |P_{cal} - P_{ini}| \leq 0,01 \text{ (1\%)} \quad (22)$$

$$\Delta Q = |Q_{cal} - Q_{ini}| \leq 0,01 \text{ (1\%)} \quad (22)$$

Se estas condições se verificarem podemos afirmar que a solução convergiu.

No caso das condições anteriores não se verificarem e/ou o número de iterações for superior ao número máximo de iterações, podemos afirmar que a solução não convergiu.



```

512 - clear dp dq
513 - for i=1:length(A)
514 -     dp(A(i),1)=abs(P(A(i))-Pcal(A(i)));
515 - end
516 - for i=1:length(B)
517 -     dp(B(i),1)=abs(P(B(i))-Pcal(B(i)));
518 -     dq(B(i),1)=abs(Q(B(i))-Qcal(B(i)));
519 - end
520 -
521 - niter=niter+1;
522 - if max(dp)<0.01 & max(dq)<0.01
523 -     flag=0;
524 -     disp(['NR -> ',num2str(niter),' iterações']);
525 - end
526 - if niter>itermax
527 -     flag=0;
528 -     disp('Não Converge');
529 - end
530 - end
531 - Time=toc;
532 - display(['Tempo ',num2str(Time),' segundos']);

```

Se convergir, é apresentado os módulos e os argumentos das tensões. Calcula-se os fluxos de potência nas linhas (32):

$$S_{ik} = U_i \times \left( \frac{U_i - U_k}{Z_{ik}} \right)^* \quad (32)$$

Assim como as perdas nas linhas:

$$perdas_{ik} = S_{ik} + S_{ki} \quad (33)$$

```

533 - final=cell(length(v),3);
534 - for i=1:length(v)
535 -     final{i,1}=strcat('u',int2str(i));
536 -     final{i,2}=num2str(v(i));
537 -     final{i,3}=num2str(tetaini(i));
538 - end
539 - disp('Tensões')
540 - disp(v)
541 - disp('Argumentos')
542 - disp(tetaini)
543 -
544 - for i=1:length(Origem)
545 -     Z(Origem(i),Destino(i))=linhas(i,3)+li*linhas(i,4);
546 - end
547 -
548 - perdas=zeros(length(Origem));
549 - for i=1:length(Origem)
550 -     S(Origem(i),Destino(i))=Uini(Origem(i))*conj((Uini(Origem(i))-Uini(Destino(i)))/Z(Origem(i),Destino(i)));
551 -     S(Destino(i),Origem(i))=Uini(Destino(i))*conj((Uini(Destino(i))-Uini(Origem(i)))/Z(Origem(i),Destino(i)));
552 -     perdas(Origem(i),Destino(i))=S(Origem(i),Destino(i))+S(Destino(i),Origem(i));
553 - end
554 - disp('Sik(Fluxo de Potencia de I para k)');
555 - disp(S);
556 - disp('Perdas (Linha I k)');
557 - disp(perdas)
558 -

```

Por fim, calculamos a potência reativa Q gerada no caso de ser uma barramento PV:

$$Q_G = Q_{cal} + Q_{cargas}$$

```

559 - ical=zeros(length(prod(:,1)));
560 - for i=1:length(tipo)
561 -     if tipo(i)~='Q'
562 -         for ii=1:length(tipo)
563 -             ical(i)=ical(i)+(Y(i,ii)*Uini(ii));
564 -             Qcal(i)=imag(Uini(i)*conj(ical(i)));
565 -             Qg(i)=Qcal(i)+tQc(i);
566 -         end
567 -     end
568 - end
569 - tQg=Qg';
570 - tQg=tQg(prod(:,1));
571 - Qgerado(1,:)=P1;
572 - Qgerado(2,:)=tQg;
573 -
574 - disp('Qgerado');
575 - fprintf(' %d %f\n',Qgerado)
576 -

```

Sendo que o resultado final no *Command Window* é o seguinte:

```

Command Window
New to MATLAB? See resources for Getting Started.
-----Método Newton-Raphson-----

NR -> 2 iterações

Tempo 0.051062 segundos

Tensões
    0.9987
    1.0100
    1.0200

Argumentos
    -0.1000
    -0.1334
         0

Sik(Fluxo de Potência de i para k)
    0.0000 + 0.0000i    0.3227 - 0.1394i    -1.0227 - 0.0595i
   -0.3214 + 0.1518i    0.0000 + 0.0000i    -0.6786 + 0.0631i
    1.0332 + 0.1647i    0.6877 + 0.0280i    0.0000 + 0.0000i

Perdas (Linha I k)
    0.0000 + 0.0000i    0.0012 + 0.0124i    0.0105 + 0.1052i
    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0091 + 0.0911i
    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i

fx Qgerado
    2    0.51339
    3    0.29115
fx >>

```

O último método é o **método do Rápido Desacoplado**.



### 3. Apresentação de resultados

Este capítulo apresenta os principais dados obtidos, através da aplicação desenvolvida, para as três redes estudadas.

#### REDE 1

Método	DC	GS	NR	FDL
U1	1L-0.0575	1.00L-0.055	1.00L-0.055	1.00L-0.055
U2	1L-0.045	1.01L-0.043	1.01L-0.043	1.01L-0.043
U3	1L0	1.02L0	1.02L0	1.02L0
Erro		0.003539	0.000023	0.005373
Iterações		5	2	2
Tempo	0.010151 s	0.018128 s	0.177553 s	0.103233 s

#### Rede 2

Método	DC	GS	NR	FDL
U1	1L0	1L0	1L0	1L0
U2	1L-0.024	0.982L-0.024	0.982L-0.023	0.982L-0.023
U3	1L-0.053	0.964L-0.052	0.964L-0.051	0.964L-0.051
U4	1L0.021	1.02L0.015	1.02L0.015	1.02L0.015
Iterações		6	2	4
Erro		0.007179	0.000025	0.002817
Tempo	0.01296 seg	0.020390 seg	0.173666 seg	0.082482 seg

#### Rede 3

Método	DC	GS	NR	FDL
U1	1L0	1.060L0	1.060L0	1.060L0

U2	1L-0.087	1.045L-0.084	1.045L-0.086	1.045L-0.086
U3	1L-0.226	1.010L-0.217	1.010L-0.220	1.010L-0.220
U4	1L-0.185	1.027L-0.177	1.026L-0.181	1.026L-0.181
U5	1L-0.159	1.033L-0.153	1.033L-0.156	1.033L-0.156
U6	1L-0.265	1.070L-0.253	1.070L-0.260	1.070L-0.260
U7	1L-0.245	1.045L-0.228	1.045L-0.235	1.045L-0.235
U8	1L-0.245	1.090L-0.228	1.090L-0.235	1.090L-0.235
U9	1L-0.277	1.028L-0.256	1.028L-0.263	1.028L-0.263
U10	1L-0.283	1.028L-0.260	1.028L-0.267	1.028L-0.267
U11	1L-0.277	1.045L-0.259	1.045L-0.266	1.045L-0.266
U12	1L-0.284	1.053L-0.267	1.053L-0.274	1.053L-0.275
U13	1L-0.287	1.046L-0.268	1.046L-0.275	1.046L-0.275
U14	1L-0.304	1.017L-0.279	1.017L-0.286	1.017L-0.286
Iterações		50	3	7
Erro		0.009948	0.000027	0.008997
Tempo	0.016835 seg	0.050368 seg	0.208321 seg	0.095150 seg

## 4. Conclusão

Este último capítulo apresenta as principais conclusões obtidas através do estudo realizado no âmbito da disciplina de Análise de Sistemas Elétricos de Energia.

O principal objetivo deste trabalho foi aplicar os conhecimentos lecionados nesta unidade curricular, de forma a criar uma ferramenta computacional capaz de analisar o trânsito de potência de um SEE.

A implementação desta aplicação mostrou-se muito útil porque permite, em poucos segundos, obter os resultados de qualquer rede, tendo em conta os vários métodos aqui abordados, tornando-se uma ferramenta importante para o estudo de redes elétricas.

Fazendo uma pequena comparação dos métodos, no que diz respeito ao tempo de execução, o modelo DC é o mais rápido, seguido do modelo GS, depois o modelo FDL e por último o modelo NR. Esta relação verificou-se para todas as redes estudadas.

Comparando os métodos iterativos, verifica-se que, apesar do erro máximo definido pelo utilizador ter sido fixado em 0.01, o método NR apresenta sempre um erro inferior em relação aos restantes modelos.

Relativamente ao número de iterações, o método NR apresenta sempre um número inferior em relação aos restantes, com exceção da primeira rede (com 3 barramentos - PQ, PV e REF) em que o número de iterações é igual nos métodos NR e FDL.

Tendo em conta que o número de iterações do modelo NR é sempre igual ou inferior aos restantes modelos iterativos e que, por outro lado, o tempo consumido é sempre superior, conclui-se que o tempo por iteração é mais elevado no método NR do que nos restantes métodos.

Importa ainda salientar o elevado número de iterações (50) obtidas através do método GS para o estudo da rede 3 (com 14 barramentos), o que demonstra que o número de iterações aumenta bastante com o aumento do número de barramentos. Por outro lado, o número de iterações, utilizando o método NR, sobe apenas de 2 para 3, quando se passa da rede de 3 barramentos para a rede de 14 barramentos, ou seja, o aumento de barramentos implica um aumento muito ligeiro no número de iterações.

De uma forma geral, apesar das diferenças aqui apresentadas, os valores encontrados para os módulos e argumentos das tensões para os diferentes métodos iterativos, são muito próximos, o que nos leva a acreditar que todos os métodos são fiáveis.

No caso do modelo DC, os valores dos módulos e argumentos das tensões são menos precisos, fruto das suas simplificações, contudo, tendo em conta alguns aspetos já referidos, este método pode também ser útil, nomeadamente para servir de solução inicial para os restantes métodos.

### **Referência Bibliográficas**

1. Documentos disponibilizados no moodle
2. Paiva, José Pedro Sucena, “*Redes de energia elétrica - Uma análise sistémica*”, IST Press, Lisboa, 2005