# 高等电力网络分析第一次作业

## 1、节点导纳矩阵、节点阻抗矩阵的计算与分析

（1）基于 IEEE 14 节点系统生成节点不定导纳矩阵 $Y_0$，思考 case14 程序中何处体现了地节点？以地为参考节点生成节点导纳矩阵 $Y$。

①在返回的 mpc 结构体中，可以在 bus（母线矩阵）的第二/三列找到 bus 的并联电导 Gs 和并联电纳 Bs，这体现了节点和地之间的联系。此外，在 branch(支路)的第五列给出了线路的电纳，可见是采用 π 型等值电路，也可以体现地节点。

```
%% bus data
%   bus_i  type  Pd  Qd  Gs  Bs  area  Vm  Va  baseKV  zone
mpc.bus = [
  1  3  0  0  0  0  1  1.06  0  0  1  1.06  0.94;
  2  2  21.7  12.7  0  0  1  1.045  -4.98  0  1  1.06  0.94;
  3  2  94.2  19  0  0  1  1.01  -12.72  0  1  1.06  0.94;
  4  1  47.8  -3.9  0  0  1  1.019  -10.33  0  1  1.06  0.94;
```

```
%   Branch Data Format
%       1   f, from bus number
%       2   t, to bus number
%   (-)     (circuit identifier)
%       3   r, resistance (p.u.)
%       4   x, reactance (p.u.)
%       5   b, total line charging susceptance (p.u.)
%       6   rateA, MVA rating A (long term rating), set to 0 for unlimited
%       7   rateB, MVA rating B (short term rating), set to 0 for unlimited
```

②生成节点不定导纳矩阵 $Y_0$ ：（main_1_1.m）

分为以下几步：初始化为 $Y_0$ = zeros(n+1, m) (n,m 分别为母线、支路数)

a. 对于所有支路：将支路的电纳 b/2 添加到 branch 的 from 和 to 母线和地节点(n + 1)之间；将支路的导纳 1/(r + jx) 添加到利用 branch 的 from 和 to 母线之间；对于有变比的支路，添加支路导纳修正关联矩阵；对于自带接地支路的节点，还要添加接地支路。利用公式$\mathrm{Y0} = \mathrm{Y0} + \ \mathrm{M}y_b M^T$ 将上述支路逐一添加，形成 $Y_0$。（具体见源码）

得到节点不定导纳矩阵 $Y_0$ （前 3*3 元素）如下：

```
Y0 =

  列 1 至 3

   6.0250 -19.3961i  -4.9991 +15.2631i   0.0000 + 0.0000i
  -4.9991 +15.2631i   9.5213 -30.2423i  -1.1350 + 4.7819i
   0.0000 + 0.0000i  -1.1350 + 4.7819i   3.1210 - 9.8379i
```

③以地为参考节点生成节点导纳矩阵 $Y$：在上面得到的不定导纳矩阵 $Y_0$ 中划去第 n+1 行第 n+1 列即可得到节点导纳矩阵 $Y$。

（2）用 matpower 中 makeYbus 程序生成节点导纳矩阵，验证（1）中节点导纳矩阵 $Y$ 的正确性，思考并总结 makeYbus 程序的编程技巧。

使用命令 Y – makeYbus(mpc)发现差在 1.0e-14 量级，因此验证了此前的计算结果。

```
>> norm(Y - Y_makeYbus, 1)

ans =

   7.3241e-15
```

**makeYbus()编程技巧总结：**

1、用字母代替具体编号，这样方便索引和修改

```
%% define named indices into bus, branch matrices
[PQ, PV, REF, NONE, BUS_I, BUS_TYPE, PD, QD, GS, BS, BUS_AREA, VM, ...
    VA, BASE_KV, ZONE, VMAX, VMIN, LAM_P, LAM_Q, MU_VMAX, MU_VMIN] = idx_bus;
```

2、赋值采用矩阵整行整列索引，避免使用 for 语句

```
stat = branch(:, BR_STATUS);                    %% ones at in-service branches
Ys = stat ./ (branch(:, BR_R) + 1j * branch(:, BR_X));  %% series admittance
Bc = stat .* branch(:, BR_B);                   %% line charging susceptance
```

3、使用中间变量，增加代码可读性

```
%% bus indices
f = branch(:, F_BUS);                  %% list of "from" buses
t = branch(:, T_BUS);                  %% list of "to" buses
```

（3）采用支路追加法生成节点阻抗矩阵 $Z$。（main_1_2.m）

先手动添加第一条线路，然后分为树支和连支添加剩下的支路。最后添加母线的接地导纳。所求出 Z 矩阵与（2）中求得的 inv(Y)的差别（e-13 量级）验证了其正确性。

```
Z =

  列 1 至 3

   0.0162 - 2.2442i   0.0049 - 2.2811i  -0.0010 - 2.3052i
   0.0049 - 2.2811i   0.0094 - 2.2684i   0.0018 - 2.2978i
  -0.0010 - 2.3052i   0.0018 - 2.2978i   0.0270 - 2.2158i
```

```
>> norm(inv(Y) - Z, 1)

ans =

   3.9280e-13
```

## 2、网络矩阵的稀疏技术

（1）对 IEEE 14 节点系统的节点导纳矩阵 *Y* 进行 LDU 分解，生成对应的因子表。(main_2.m line 5; factorize.m)

首先，用 MATLAB 自带的 lu()函数直接分解得到 L1, D1, U1；自己写程序 factorize(), 采用常规方法函数进行因子分解，得到 L2, D2, U2.乘起来与 Y 比较验证：

```
norm(L1 * D1 * U1 - Y, 1)

ans =

  1.3692e-14
```

```
norm(L2 * D2 * U2 - Y, 1)

ans =

  9.7279e-15
```

（2）采用 Tinney-2 编号方法、Tinney-3 编号方法对 IEEE 14 节点系统重新编号并重新进行 LDU 分解，与原始编号下的 LDU 分解进行对比，探讨 3 种编号方法的优劣。(main_2.m line 19; TinneyTwo.m; TinneyThree.m)

编写函数 TinneyTwo()和 TinneyThree()两个函数分别用于重新编号。采用 MATLAB 的 tic/toc 函数对重新编号过程（包括编号后排序）、编号后 LDU 分解过程进行计时。每个过程运行 1000 次，得到时间花费如下：

| 编号方式 | 编号过程时间(s) | LDU 分解过程(s) | Y 矩阵非零元个数 |
|---|---|---|---|
| 原始编号 | - | 5.12 | 56 |
| Tinney-2 | 0.34 | 2.46 | 38 |
| Tinney-3 | 0.45 | 2.32 | 38 |

分析：①可以看到对 14 节点而言，采用 Tinney-3 和 Tinney-2 编号方法后 LDU 分解过程花费的时间大约只有原始编号的 50%左右， Y 矩阵的非零元个数都从 56 个减少到了 38 个。

②Tinney-3 方法比 Tinney-2 方法更能减少 LDU 分解的时间花费，但减少的比例（6%）并不大，而同时 Tinney-3 方法所需要的编号时间大约比 Tinney-2 编号方法多 30%。本例中两者用于编号和 LDU 分解的总时间花费差别不大。

③ 两种优化编号过程花费的时间都远小于随后 LDU 分解中节约的时间。可以合理归纳，当节点数目较大时，花费较小的时间优化编号可以明显降低花费的总时间。而且 Tinney-3 方法和 Tinney-2 方法的效果相差不大。因此，仅仅在 LDU 分解本身需要花费很长时间，或者编号一次后可以保持很久的场景下需要 Tinney-3 方法。

（3）基于 LDU 分解的结果，采用连续回代法重新生成节点阻抗矩阵 Z，并与之前的结果对比以验证正确性。(main_2.m line 62; lud2Z.m)

编写 lud2Z()函数用于从 LDU 矩阵生成 Z 矩阵；将程序得到的 Z 矩阵与直接对 Y 矩阵求逆的结果对比（如下），两者相差在 1e-13 量级，验证了连续回代法生成节点阻抗矩阵 Z 的正确性。

```
>> norm(inv(Y) - Z, 1)

ans =

   5.7827e-13
```

## 3、网络矩阵的修正解法

（1）在节点 5 与节点 8 之间添加一条 *r*、*x*、*b* 参数（标幺值）分别为 0.016、0.058、0.162 的支路，分别采用面向支路、面向节点修正的补偿法计算修正后的节点导纳矩阵 *Y'*，探讨 2 种修正算法的区别。(main_3.m)

$$\begin{bmatrix} 1 & 1 & \\ -1 & & 1 \end{bmatrix}\begin{bmatrix} \Delta y & & \\ & \Delta y_c & \\ & & \Delta y_c \end{bmatrix}\begin{bmatrix} 1 & -1 \\ 1 & \\ & 1 \end{bmatrix} \quad \text{v.s.} \quad \begin{array}{c} i \\ j \end{array}\begin{bmatrix} 1 & \\ & 1 \end{bmatrix}\begin{bmatrix} \Delta y + y_c & -\Delta y \\ -\Delta y & \Delta y + y_c \end{bmatrix}\begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$$

区别：①修正复杂度不同：采用面向支路时，需要添加三条支路：(5, n+1), (8, n+1), (5, 8)，需要对 Y 修正三次；采用面向节点的方法，只需要考虑两个节点，而对 Y 只进行一次修正，在程序中更为清晰。

②特殊情况下可能网络发生的变化不是对称的，关联矢量也不是互为转置的，因此使用面向节点的方法更具一般性。

③在因子表的秩 1 修正中，每次加入等效为一个支路，因此使用面向支路的方法与因子表的修正的对应性更强。

（2）在（1）的基础上，再在节点 11 与节点 12 之间添加一条 *r*、*x*、*b* 参数（标幺值）分别为 0.012、0.085、0.158 的支路，分别采用因子表的秩 1 修正算法、因子表的局部再分解算法计算新因子表，探讨 2 种修正算法的计算效率和适用场景。(main_3.m line 42; rankOne.m; rankOneAdd.m)

①编写 rankOne()函数，使用迭代的方法进行因子分解表的秩 1 修正；迭代的终止条件为 D 矩阵维度为 1×1，此时直接对 D 进行修正即可。

②对于局部再分解（main_3.m line 69）方法，直接用代码实现。将上述两种方法得到的新的 LDU 矩阵相乘结果分别与修正后的 Y 矩阵进行比较，差别在 e-14 量级，验证了两种方法的正确性。

```
>> main_3
秩 1 修正误差:  1.1255e-14
局部再分解误差:  1.1137e-14
```

（3）根据（2）中的新因子表，计算添加两条支路后的节点导纳矩阵 *Y”*和节点阻抗矩阵 *Z”*。使用 *Y’*、*Y”*、*Z”*验证矩阵求逆辅助定理的正确性，并尝试解释 *c* 的物理意义。(main_3 line 85)

①利用（2）中的新因子表，直接将 LDU 相乘即可得到节点导纳矩阵 *Y”*；用此前写的 ldu2Z 可以利用 LDU 矩阵生成 *Z”*。*可以与追加法生成的*节点导纳矩阵 Y2 比较，验证 LDU 分解的正确性。

```
>> norm(L * D * U - Y2, 1)

ans =

  1.1137e-14
```

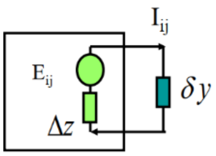
②利用 Y’的逆矩阵 Z’，采用支路追加法添加(11, 12), (11, n+1), (12, n+1)三条支路，即可生成 Z”。将 Z”与（2）中生成的 Z 矩阵相比，验证矩阵求逆辅助定理的正确性。

```
>> norm(Z - Z_new, 1)

ans =

  5.0019e-14
```

③c 的物理意义：




矩阵求逆辅助定理（$\tilde{\mathbf{A}}=\mathbf{A}+\mathbf{M}a\mathbf{N}^{\mathrm{T}}, c=(a^{-1}+\mathbf{N}^{T}\mathbf{A}^{-1}\mathbf{M})^{-1} \Rightarrow \tilde{\mathbf{A}}^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{M}c\mathbf{N}^{\mathrm{T}}\mathbf{A}^{-1}$）

用导纳发生编号的两个节点组成的端口的戴维南等效电路进行分析，可以发现 c 刚好是这个回路中的等效阻抗。

$$c^{-1}=\delta y^{-1}+\mathbf{M}^{\mathrm{T}}\mathbf{Y}^{-1}\mathbf{M}=\delta y^{-1}+\Delta z$$

在矩阵求逆定理中，c 和等效电动势（假设注入电流均为 1）相乘即得到回路电路，这部分电流添加到端口上就可以对原有的各节点的电压进行修正，所得到的新的节点电压的物理意义就是新的阻抗矩阵。这样，就对原有的阻抗矩阵进行了修正。

**回路电流** $\dot{I}_{ij}=c\dot{E}_{ij}=c\mathbf{M}^{\mathrm{T}}\mathbf{Y}^{-1}\dot{\mathbf{I}}$

**补偿电流** $\Delta\dot{\mathbf{I}}=-\mathbf{M}\dot{I}_{ij}=-\mathbf{M}c\mathbf{M}^{\mathrm{T}}\mathbf{Y}^{-1}\dot{\mathbf{I}}$

**补偿后总注入电流** $\dot{\tilde{\mathbf{I}}}=\dot{\mathbf{I}}+\Delta\dot{\mathbf{I}} \Rightarrow \dot{\tilde{\mathbf{V}}}=\mathbf{Y}^{-1}\dot{\tilde{\mathbf{I}}}$