第一次大作业：第一题 电阻网络

姓名：赵瑞博
学号：1800011355

2020 年 3 月 30 日

0.1 方法框架

对于无源电阻网络，用指标 $i = 0, 1, 2, 3 \ldots, n$ 标记各个节点，记节点 i, j 之间的直连导纳为 $g_{ij} = \frac{1}{r_{ij}}$，断路导纳记为 0。同时记节点 i 的电势为 U_i，由外部流入该节点电流为 I_i，由基尔霍夫第二定律可列出：

$$\sum_{j \neq i} g_{ij}(U_i - U_j) = I_i,\ i = 1, 2, \ldots, n \tag{1}$$

可化为矩阵方程：

$$GU = I \tag{2}$$

其中导纳矩阵各元为

$$g_{ij} = \begin{cases} \sum_{k \neq i} g_{ik}, & i = j, \\ -g_{ij}, & i \neq j. \end{cases} \tag{3}$$

导纳矩阵 G 为奇异矩阵，存在零特征值特征向量，对应电势零点可任取。如果要求 i, j 之间的等效电阻，可取节点 i 为电势零点，此时可将导纳矩阵第 i 行、第 i 列删去，在维持对称性的同时消去奇异性，同时删去 I_i，化为 $n - 1$ 元的线性方程组：

$$G'U' = I' \tag{4}$$

其中 I_j 取单位电流 $I_j = 1$，则有等效电阻数值上：

$$R_{ij} = U_j \tag{5}$$

经过上述操作后，那么问题就在于导纳矩阵的建立与矩阵方程的求解。

0.2 语言与简单类、函数搭建

编程语言选择为 $Python$

为方便接下来的算法编写与计算进行，在 $wheels.py$ 中定义简单的矩阵类 $Matrix$，并为其重载与定义了

- 索引与整行整列的获取与删除
- 正负号与矩阵的加法、减法、乘法、数乘、内积、求幂等基本运算
- 相等判定

- 矩阵求转置、迹、行列式、共轭

另外定义了几种特殊矩阵生成函数与上三角矩阵前代算法等几种基本求解算法的函数。

另外由于 *Python* 语言内置复数且重新定义复数没有什么学习意义，故直接使用内置的复数类型。

1 正方形电阻网络求解

1.1 导纳矩阵的生成

正方形电阻网络节点指标顺序如下图所示（n 为边上节点数）

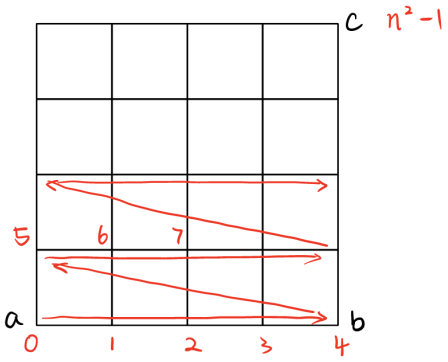

图 1: 正方形电阻网络节点标号

这样选取指标后，可以看出对于节点 i，$i-1, i+1, i-n, i+n$ 四点可能与其相接，导纳矩阵可由如下步骤生成

Algorithm 1 生成正方形电阻网络导纳矩阵

Input: 边上节点数 n

Output: 导纳矩阵 g

function $get\_G(n)$
  $g \leftarrow n^2 \times n^2$ 零矩阵
  for $i = 0 \rightarrow n^2 - 1$ **do**
    for $j = 0 \rightarrow n^2 - 1$ **do**
      if $|j-1| == 1$ **or** $|j-1| == n$ **then**
        $g_{ij} = -1, g_{ii} += 1$
      end if
    end for
  end for
  for $i = 1 \rightarrow n - 1$ **do**
    $g_{i*n,i*n-1} = 0, g_{i*n,i*n} -= 1, g_{i*n-1,i*n} = 0, g_{i*n-1,i*n-1} -= 1$
  end for
  return g
end function

1.2 直接法求解

设其需要求 a, b 节点间等效电阻（$a < b$）

1.2.1 导纳矩阵去奇异性

如基本操作中所述，不妨取 $U_a = 0$，导纳矩阵去奇异性只需删去导纳矩阵的第 a 行与第 a 列即可。这在 $wheels.py$ 中定义的 $Matrix$ 类中包含删去整行整列的方法。

1.2.2 导纳矩阵 Cholesky 分解

由于导纳矩阵 G 是一个对称正定矩阵，可将其进行 $Cholesky$ 分解

$$G = LDL^T \tag{6}$$

其中 L 为下三角矩阵，D 为对角矩阵。如此，方程求解可变为

$$L^T U = (LD)^{-1} I \tag{7}$$

这样一个上三角方程组的求解问题。在进行 Cholesky 分解时，由于导纳矩阵 G 是一个对称正定带状矩阵，最终可使 L 为下三角带状矩阵，半带宽与 G 相同为 n，令 D 的元素为 $d_{ii} = \frac{1}{l_{ii}}$，通过矩阵乘法可知：

$$l_{ij} = a_{ij} - \sum_{k=r}^{j-1} l_{ik} l_{jk} / l_{kk} \tag{8}$$

其中

$$r = \begin{cases} 1, & i <= n, \\ i - n, & i > n. \end{cases} \tag{9}$$

如此对带状矩阵优化后速度远快于直接未优化算法。

Algorithm 2 导纳矩阵 Cholesky 分解带状优化

Input: 去奇异的导纳矩阵 g, 半带宽 n

Output: 分解后的 L, D 覆盖的 g

$m = g$ 的阶数
for $i = 0 \to m - 1$ **do**
  if $i < n$ **then**
    for $j = 0 \to i$ **do**
      for $k = 0 \to j - 1$ **do**
        $g_{ij} = g_{ij} - g_{ik} * g_{jk}/g_{kk}$
      end for
    end for
  else
    for $j = i - n \to i$ **do**
      for $k = 0 \to j - 1$ **do**
        $g_{ij} = g_{ij} - g_{ik} * g_{jk}/g_{kk}$
      end for
    end for
  end if
end for

1.2.3 电流 I 的处理

对于方程 (4) 来说，I 除了 $b-1$ 项（因为 a 移去，故后面索引全部前移）为 1 外，其余均为 0

而由上个步骤所知，若化为上三角方程组，方程右侧的 I 应改为 $(LD)^{-1}I$，由于 LD 是一个下三角矩阵，我们可以求解

$$LDx = I \tag{10}$$

所得 x 即为方程 (7) 的右端列向量。下三角矩阵求解采用前代算法。

Algorithm 3 电流列向量 I 的构建与处理

Input: g, 与其阶数 m, 半带宽 n, 第二个代求节点指标 b

Output: 处理后的 I

$I = m \times 1$ 的零矩阵
$I_{b-1} = 1$
for $i = 0 \to m - 1$ **do**
  for $j = 0 \to i - 1$ **do**
    $I_i = I_i - g_{ij}/g_{jj} * I_j$
  end for
  $I_i = I_i$
end for

1.2.4 求解

求解直接带入到 $wheels.py$ 中定义的上三角矩阵回代求解函数 $up_tri_msolve(M,b)$，两参数分别输入 g^T, I，函数返回解向量 x, 取 x_{b-1} 即为代求等效电阻 R_{ab}

1.3 迭代法求解

1.3.1

与直接法相同，需要将导纳矩阵 g 去奇异化，方法也相同。而后由于 g 并没有进行分解操作，方程右侧 I 也只需为

$$I_i = \begin{cases} 1, & i = b-1, \\ 0, & else. \end{cases} \tag{11}$$

1.3.2 迭代法求解方程

刚开始采用的时 $Jacobi$ 迭代法，然而 $Jacobi$ 迭代法收敛速度太慢，经常 1000 次迭代也无法达到想要的精度，故又采取 $Gauss-Seidel$ 迭代法。此时在边长为 1，4，16 的情况下相较于 $Jacobi$ 迭代法有很大改善，然而在边长 64 的情况下，计算时间超出可承受范围。为解决计算时间长的问题，从计算规模入手，由于导纳矩阵是一个带状稀疏矩阵，直接进行矩阵乘法很多计算量浪费在了 0 元素的计算上，故在循环缩进限制，对带状矩阵进行优化，使得计算速度大幅提升。为了更好的提升收敛速度，采用超松弛迭代法，在几次实验下，松弛系数取 1.9 获得了更好的结果，故松弛系数定为 1.9。自此，迭代算法如下 取解 x 中 x_{b-1} 即为代求等效电阻。

Algorithm 4 迭代法求解

Input: g, I, 阶数 m, 半宽 n, 最大迭代次数 M, 判停标准 e

Output: x

两组解向量 x, x_0 初始化
for $k = 0 \to M-1$ **do**
    $x_0 = x$
    for $i = 0 \to m-1$ **do**
        for $j = \max(0, i-n) \to \min(m-1, i+n), j \neq i$ **do**
            $x_i = x_i - g_{ij} * x_j$
        end for
        $x_i = x_i / g_{ii}$
        $x_i = -0.9 * (x_0)_i + 1.9 * x_i$
    end for
    if $|x - x_0| < e$ **then**
        STOP
    else if $k == M-1$ **then**
        Not converged at given M and e **STOP**
    end if
end for

1.4 结果与时间对比

表 1: 正方形电阻网络结果与时间对比

节点	边长	直接法结果	直接法用时 (s)	迭代法 M, e	迭代法结果	迭代法用时 (s)
ac	1	1.0	0.0	1000,1e-10	1.00000321	0.00696
ab	1	0.75	0.00097	1000,1e-10	0.75000163	0.00599
ac	4	2.13636363	0.00304	1000,1e-10	2.13637192	0.04785
ab	4	1.90151515	0.00197	1000,1e-10	1.90152229	0.02991
ac	16	3.68559246	0.62556	1000,1e-10	3.68556938	3.31785
ab	16	3.46358794	0.82887	1000,1e-10	3.46356175	3.61336
ac	64	5.39237679	440.44	2000,1e-4	5.08148424	752.98
ab	64	5.17164678	440.81	2000,1e-4	4.89899806	747.84

结果上，对于计算规模较小的情况，经过优化后的直接法无论在准确度与速度上都超过了迭代法。但是可以发现，迭代法随计算规模增耗时增长速率要小于直接发，这意味着在大规模计算下，迭代的速度优势将会体现的更加明显。

2 三角形电阻网络

2.1 导纳矩阵的生成

三角形电阻网络节点指标顺序如下图所示（n 为边上节点数）

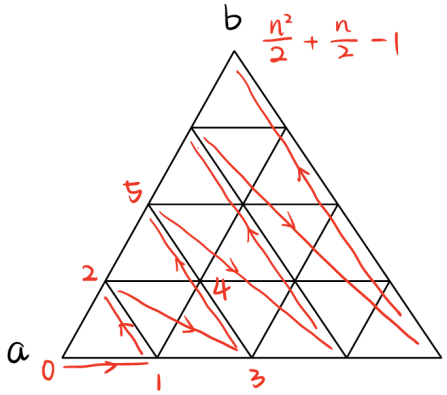

图 2: 三角形电阻网络节点标号

这样选取指标后，可以看出对于节点 i，$i-k-1, i-k, i-1, i+1, i+k+1, i+k+2$ 六点可能与其相接，其中 k 为层数，最大为 $n-1$，导纳矩阵可由如下步骤生成

Algorithm 5 生成三角形电阻网络导纳矩阵

Input: 边上节点数 n

Output: 导纳矩阵 g

function $get\_G(n)$
    $g \leftarrow \frac{n^2+n}{2} \times \frac{n^2+n}{2}$ 零矩阵
    $g_{00} = 2, g_{01} = -1, g_{02} = -1$
    $i = 1$
    for $k = 1 \rightarrow n-2$ **do**
        $g_{i,i-k} = -1, g_{i,i+1} = -1, g_{i,i+k+1} = -1, g_{i,i+k+2} = -1, g_{ii} = 4$
        for $j = 1 \rightarrow k-1$ **do**
            $s = i + j$
            $g_{s,s-k-1} = -1, g_{s,s-k} = -1, g_{s,s-1} = -1, g_{s,s+1} = -1, g_{s,s+k+1} = -1, g_{s,s+k+2} = -1, g_{ii} = 6$
        end for
        $s = i + k$
        $g_{s,s-k-1} = -1, g_{s,s-1} = -1, g_{s,s+k+1} = -1, g_{s,s+k+2} = -1, g_{ii} = 4$
        $i = s + 1$
    end for
    $g_{i,i-n+1} = -1, g_{i,i+1} = -1, g_{ii} = 2$
    for $j = 1 \rightarrow n-1$ **do**
        $s = i + j$
        $g_{s,s-n} = -1, g_{s,s-n+1} = -1, g_{s,s-1} = -1, g_{s,+1} = -1, g_{ii} = 4$
    end for
    $s = i + n - 1$
    $g_{s,s-n} = -1, g_{s,s-1} = -1, g_{ii} = 2$
    return g
end function

2.2 结果与时间对比

由于解方程的算法与 1 相同，故不在赘述。直接看其结果 对数据进行分析基本可以获得与

表 2: 三角形电阻网络结果与时间对比

节点	边长	直接法结果	直接法用时 (s)	迭代法 M, e	迭代法结果	迭代法用时 (s)
ab	1	0.66666666	0.0	1000,1e-10	0.66666849	0.00299
ab	4	1.67479675	0.00100	1000,1e-10	1.68480024	0.01698
ab	16	3.02437252	0.13666	1000,1e-10	3.02435500	0.94249
ab	64	4.50323002	106.58	2000,1e-4	4.34808117	286.11

1.4 相同的结论，即小规模计算直接法占优，但是迭代法耗时随计算规模增大要低于直接法。

3 六边形电阻网络

3.1 导纳矩阵的生成

六边形电阻网络，在如下图所示经过三端网等效变化后，可化为三角形网络如图。不同的是红色线代表组织为 3，绿色阻值为 2。节点指标顺序如下图所示（n 为边上节点数）

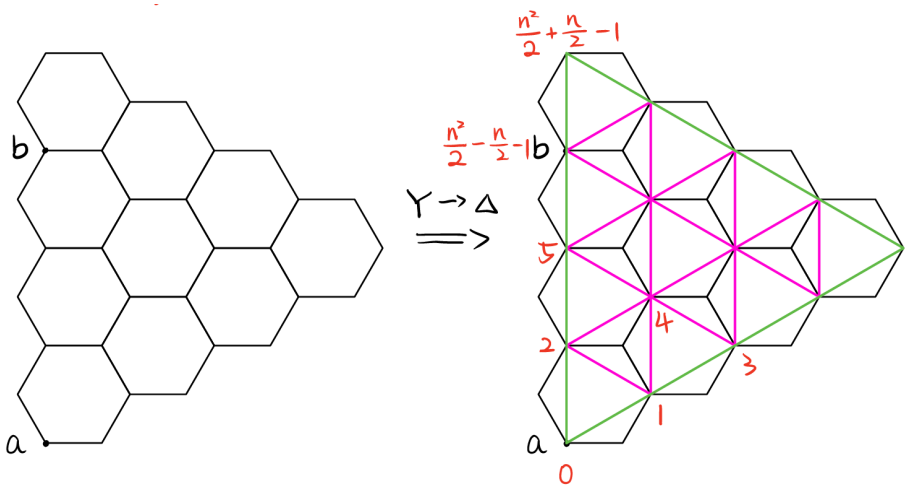

图 3: 六边形电阻网络节点标号

类似于 2.1 的算法，导纳矩阵计算如下所示

Algorithm 6 生成六边形电阻网络导纳矩阵

Input: 边上节点数 n

Output: 导纳矩阵 g

function $get\_G(n)$

  $g \leftarrow \frac{n^2+n}{2} \times \frac{n^2+n}{2}$ 零矩阵

  $g_{00} = 1, g_{01} = -1/2, g_{02} = -1/2$

  $i = 1$

  for $k = 1 \to n-2$ **do**

    $g_{i,i-k} = -1/2, g_{i,i+1} = -1/3, g_{i,i+k+1} = -1/2, g_{i,i+k+2} = -1/3, g_{ii} = 5/3$

    for $j = 1 \to k-1$ **do**

      $s = i + j$

      $g_{s,s-k-1} = -1/3, g_{s,s-k} = -1/3, g_{s,s-1} = -1/3, g_{s,s+1} = -1/3, g_{s,s+k+1} = -1/3, g_{s,s+k+2} = -1/3, g_{ii} = 2$

    end for

    $s = i + k$

    $g_{s,s-k-1} = -1/2, g_{s,s-1} = -1/3, g_{s,s+k+1} = -1/3, g_{s,s+k+2} = -1/2, g_{ii} = 5/3$

    $i = s + 1$

  end for

  $g_{i,i-n+1} = -1/2, g_{i,i+1} = -1/2, g_{ii} = 1$

  for $j = 1 \to n-1$ **do**

    $s = i + j$

    $g_{s,s-n} = -1/3, g_{s,s-n+1} = -1/3, g_{s,s-1} = -1/2, g_{s,+1} = -1/2, g_{ii} = 5/3$

  end for

  $s = i + n - 1$

  $g_{s,s-n} = -1/2, g_{s,s-1} = -1/2, g_{ii} = 1$

  return g

end function

3.2 结果与时间对比

由于解方程的算法亦与 1 相同，故不在赘述。直接看其结果 对数据进行分析基本可以获得

表 3: 六边形电阻网络结果与时间对比

节点	边长	直接法结果	直接法用时 (s)	迭代法 M, e	迭代法结果	迭代法用时 (s)
ab	4	2.81937394	0.00097	1000,1e-10	2.81937444	0.01896
ab	16	6.53437653	0.14065	1000,1e-10	6.53436294	0.94642
ab	64	10.83543674	123.38	2000,1e-3	10.41005341	265.60

与 1.4 相同的结论，即小规模计算直接法占优，但是迭代法耗时随计算规模增大要低于直接法。经过上面多组数据的分析，可以看到截断标准定为 1e-10 时，结果可以达到 5 位以上的有效位数；但是更高的截断标准往往意味着更多的迭代次数，为了控制时间，不得不在一些情况下牺牲精确度降低阶段标准。

4 交流网络

4.1 导纳矩阵的生成

三角形交流网络，引入复阻抗的概念后，相当于将三角形电阻网络扩充到复数域，其他没有什么区别。记圆频率为 ω，红色线代表感抗 $X_L = \omega j$，蓝色代表容抗 $X_C = -j/\omega$。节点指标顺序如下图所示（n 为边上节点数）

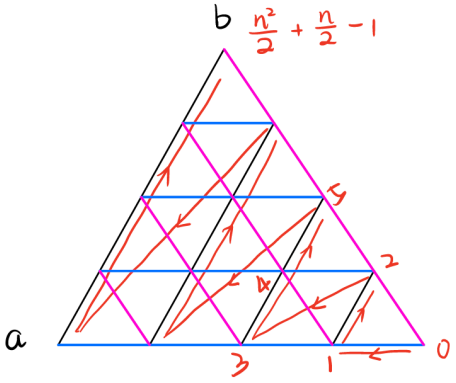

图 4: 交流网络节点标号

类似于 2.1 的算法，导纳矩阵计算如下所示

Algorithm 7 生成交流网络导纳矩阵

Input: 边上节点数 n, 圆频率 w

Output: 导纳矩阵 g

function $get\_G(n,w)$

  $g \leftarrow \frac{n^2+n}{2} \times \frac{n^2+n}{2}$ 零矩阵

  $g_{00} = 1j * (w - 1/w), g_{01} = -1j * w, g_{02} = 1j/w$

  $i = 1$

  for $k = 1 \rightarrow n - 2$ **do**

    $g_{i,i-k} = -1j * w, g_{i,i+1} = -1, g_{i,i+k+1} = -1j * w, g_{i,i+k+2} = 1j/w, g_{ii} = 1 + 1j * (2 * w - 1/w)$

    for $j = 1 \rightarrow k - 1$ **do**

      $s = i + j$

      $g_{s,s-k-1} = 1j/w, g_{s,s-k} = -1j * w, g_{s,s-1} = -1, g_{s,s+1} = -1, g_{s,s+k+1} = -1j * w, g_{s,s+k+2} = 1j/w, g_{ii} = 2 + 2j * (w - 1/w)$

    end for

    $s = i + k$

    $g_{s,s-k-1} = 1j/w, g_{s,s-1} = -1, g_{s,s+k+1} = -1j * w, g_{s,s+k+2} = 1j/w, g_{ii} = 1 + 1j * (w - 2/w)$

    $i = s + 1$

  end for

  $g_{i,i-n+1} = -1j * w, g_{i,i+1} = -1, g_{ii} = 1 + 1j * w$

  for $j = 1 \rightarrow n - 1$ **do**

    $s = i + j$

    $g_{s,s-n} = 1j/w, g_{s,s-n+1} = -1j * w, g_{s,s-1} = -1, g_{s,+1} = -1, g_{ii} = 2 + 1j * (w - 1/w)$

  end for

  $s = i + n - 1$

  $g_{s,s-n} = 1j/w, g_{s,s-1} = -1, g_{ii} = 1 - 1j/w$

  return g

end function

4.2 直接法求解与结果

直接发仍采用 1.2 的方法，此方法完全兼容复数运算，故算法上没有改动。对于圆频率 w 在 $0,2$ 上均匀取 2000 个点。由于当 $w=1$ 时，矩阵对角线上出现 0 项，无法直接用 1.2 中算法求解，故扔掉此点，依靠附近点逼近。所获得的阻抗与相角响应曲线如图：

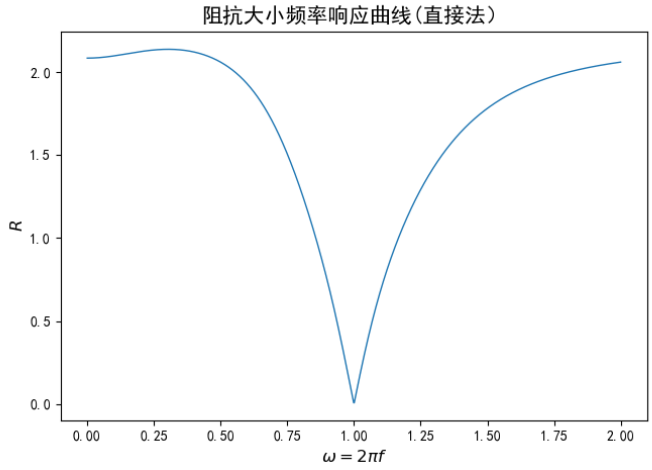

图 5: 阻抗频率响应曲线

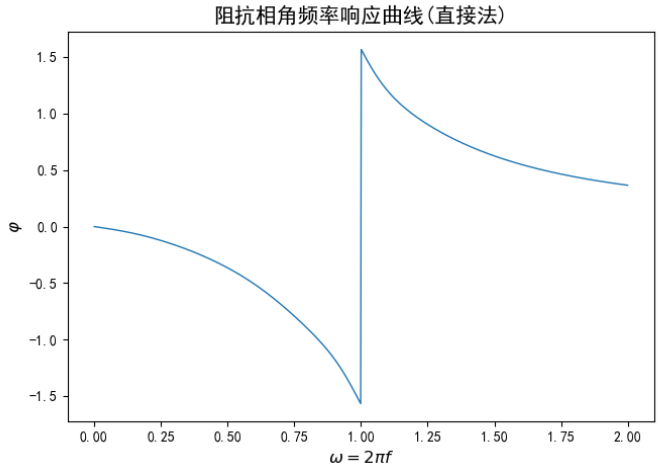

图 6: 相角频率响应曲线

可以看出来共振峰只有一处 $w=1$，对应阻抗为 0，而相角为突变处。

程序耗时 $1.12798262s$。

4.3 迭代法求解与结果

由于迭代法需要有截断判据，故重新定义模长，使其包含复向量的取模。

考虑 1.3 迭代过程同样兼容复数运算，故直接采取超松弛法迭代，然而结果是不收敛。而后重新尝试了最基本的 $gauss_seidel$ 迭代法，并尝试先将导纳矩阵乘复共轭实数化后仍然无法得到收敛结果。也就是说，直接发更加的稳定，迭代法则要以来数据类型等条件。