# 第三章作业

## 1 问题叙述

一个 $n$ 级电阻网络组成的电路如图所示。

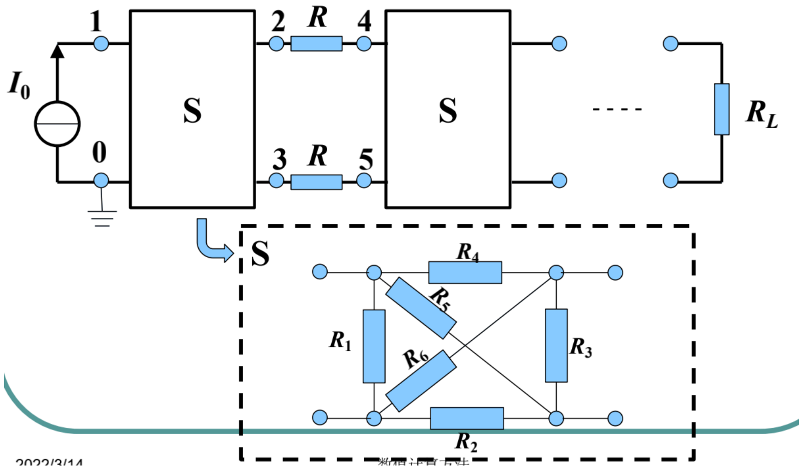


图 1: 问题叙述

图中，$I_0$ 为恒流源，所有电阻阻值为 $R$，当 $n \geq 1$ 时，节点数为 $4n$。

- 由欧姆定律和基尔霍夫定律建立求解各节点电势 $V_i$ 的线性代数方程组。
- 若 $I_0 = 1A$，$R = 1\Omega$. 请确定 $n = 1 - 5$ 时，每个节点的电势 $V_i$，请用不同方法求解并进行对比。

## 2 问题分析

利用节点电压法，以 $4n$ 个节点为自变量，列写 $4n$ 个节点电压方程。

$$G_{n1}U_{n1} + G_{n2}U_{n2} + \cdots + G_{nn}U_{nn} = I_{snn} \tag{1}$$

其中:

- $G_{ii}$：自电导，恒为正，表示连接于节点 i 的支路的电导之和
- $G_{ij}$：互电导，恒为负，表示节点 i、j 的公共支路的电导之和
- $U_{ni}$：节点 i 的节点电压
- $I_{sii}$：电源注入节点 i 的电流

然后将其写为线性方程组标准形式:

$$Ax = b \tag{2}$$

利用多种方法进行求解并进行对比，分析其优劣性。

使用理论分析得到真值：首先只看 0，1，2，3 四个节点，将 2、3 节点之间的电阻 $R_3$ 断路，则由对称性，2 和 3 是自然等位点。随后将 $R_3$ 以及后面所有电阻接回去，则由自然等位点的性质，节点 2、3 的电压相等，因此后面所有电阻中的电流均为 0。因此，除了 0、1 节点，其他所有节点的电压值均一样，且仅有 $R1$、$R2$、$R4$、$R5$、$R6$ 中存在电流，可以很快算出解的理论值为：

$$V_0 = 0V, V_1 = 0.5V, V_i = 0.25V(i \geq 2) \tag{3}$$

## 目录

## 3 第一问解答

首先利用节点电压法列出 $4n$ 个节点处的方程：

$$V_0 = 0 \tag{4}$$

$$3\frac{V_1}{R} = \frac{V_0}{R} + \frac{V_2}{R} + \frac{V_3}{R} + I_0 \tag{5}$$

$$\vdots$$

$$4\frac{V_{4i}}{R} = \frac{V_{4i-2}}{R} + \frac{V_{4i+1}}{R} + \frac{V_{4i+2}}{R} + \frac{V_{4i+3}}{R}, i = 1, 2, \cdots, n-1 \tag{6}$$

$$4\frac{V_{4i+1}}{R} = \frac{V_{4i-1}}{R} + \frac{V_{4i}}{R} + \frac{V_{4i+2}}{R} + \frac{V_{4i+3}}{R}, i = 1, 2, \cdots, n-1 \tag{7}$$

$$4\frac{V_{4i+2}}{R} = \frac{V_{4i}}{R} + \frac{V_{4i+1}}{R} + \frac{V_{4i+3}}{R} + \frac{V_{4i+4}}{R}, i = 0, 1, \cdots, n-2 \tag{8}$$

$$4\frac{V_{4i+3}}{R} = \frac{V_{4i}}{R} + \frac{V_{4i+1}}{R} + \frac{V_{4i+2}}{R} + \frac{V_{4i+5}}{R}, i = 0, 1, \cdots, n-2 \tag{9}$$

$$\vdots$$

$$4\frac{V_{4n-2}}{R} = \frac{V_{4n-4}}{R} + \frac{V_{4n-3}}{R} + 2\frac{V_{4n-1}}{R} \tag{10}$$

$$4\frac{V_{4n-1}}{R} = \frac{V_{4n-4}}{R} + \frac{V_{4n-3}}{R} + 2\frac{V_{4n-2}}{R} \tag{11}$$

然后将其写为矩阵形式：

$$Ax = b \tag{12}$$

$$A = \begin{bmatrix} 1 & & & & & & & & & & \\ -1 & 3 & -1 & -1 & & & & & & & \\ -1 & -1 & 4 & -1 & -1 & & & & & & \\ -1 & -1 & -1 & 4 & & -1 & & & & & \\ & & -1 & & 4 & -1 & -1 & -1 & \cdots & & \\ & & & -1 & -1 & 4 & -1 & -1 & \cdots & & \\ & & & \vdots & \vdots & \vdots & \vdots & \vdots & & & \\ & & & & & & -1 & -1 & 4 & -2 \\ & & & & & & -1 & -1 & -2 & 4 \end{bmatrix} \tag{13}$$

$$x = \begin{bmatrix} V_0 & V_1 & V_2 & \cdots & V_{n-1} \end{bmatrix}^T \tag{14}$$

$$b = \begin{bmatrix} 0 & I_0R & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}^T \tag{15}$$

其中，

$$A(1,1) = 1, A(2,1) = -1, A(2,2) = 3, A(2,3) = -1, A(2,4) = -1$$

$$A(4i-1,4i-3)=-1, A(4i-1,4i-2)=-1, A(4i-1,4i-1)=4$$

$$A(4i-1,4i)=-1, A(4i-1,4i+1)=-1$$

$$A(4i,4i-3)=-1, A(4i,4i-2)=-1, A(4i,4i-1)=-1$$

$$A(4i,4i)=4, A(4i,4i+2)=-1$$

$$A(4i+1,4i-1)=-1, A(4i+1,4i+1)=4, A(4i+1,4i+2)=-1$$

$$A(4i+1,4i+3)=4, A(4i+1,4i+4)=-1$$

$$A(4i+2,4i)=-1, A(4i+2,4i+1)=-1, A(4i+2,4i+2)=4$$

$$A(4i+2,4i+3)=-1, A(4i+2,4i+4)=-1 (i=1,2,\cdots,n-1)$$

$$A(4n-1,4n-3)=-1, A(4n-1,4n-2)=-1, A(4n-1,4n-1)=4, A(4n-1,4n)=-2$$

$$A(4n,4n-3)=-1, A(4n,4n-2)=-1, A(4n,4n-1)=-2, A(4n,4n)=4$$

矩阵 $A$ 的其余项等于 0。

矩阵构造的 Matlab 代码如下：

```matlab
n0=3;% 原题中的n
n=4*n0;
a=zeros(n,n);
b=zeros(n,1);
x=zeros(n,1);
a(1,1)=1;
a(2,1)=-1;a(2,2)=3;a(2,3)=-1;a(2,4)=-1;
for i=1:n0-1
    a(4*i-1,4*i-3)=-1;a(4*i-1,4*i-2)=-1;a(4*i-1,4*i-1)=4;
    a(4*i-1,4*i)=-1;a(4*i-1,4*i+1)=-1;
    a(4*i,4*i-3)=-1;a(4*i,4*i-2)=-1;a(4*i,4*i-1)=-1;
    a(4*i,4*i)=4;a(4*i,4*i+2)=-1;
    a(4*i+1,4*i-1)=-1;a(4*i+1,4*i+1)=4;a(4*i+1,4*i+2)=-1;
    a(4*i+1,4*i+3)=-1;a(4*i+1,4*i+4)=-1;
    a(4*i+2,4*i)=-1;a(4*i+2,4*i+1)=-1;a(4*i+2,4*i+2)=4;
    a(4*i+2,4*i+3)=-1;a(4*i+2,4*i+4)=-1;
end
a(4*n0-1,4*n0-3)=-1;a(4*n0-1,4*n0-2)=-1;a(4*n0-1,4*n0-1)=4;a(4*n0-1,4*n0)=-2;
```

```
a(4*n0,4*n0-3)=-1;a(4*n0,4*n0-2)=-1;a(4*n0,4*n0-1)=-2;a
    (4*n0,4*n0)=4;
a0=a;
b(2)=1;
```

# 4 第二问解答

## 4.1 直接法

- 经过有限步算术运算，可求得方程组的精确解的方法。（若在计算过程中没有舍入误差）
- 可预先估算使用机器时间，计算量小，但要占用较多内存，程序复杂。一般说来，适用于方程组的系数矩阵阶数不太高的问题。

### 4.1.1 Guass 列主元消去法

原始 Guass 消去法有以下几个问题：

- 被 0 除：消去和回代中都存在这个问题。
- 舍入误差：大规模问题中，每一个计算结果都依赖于前面的结果。
- 病态方程组
- 奇异方程组：两个方程组完全相等时，n 个未知数而只有 n-1 个方程，对大规模方程组，不容易发现这种奇异性，利用奇异方程组的行列式为 0 进行判断。

因此，可以引入 Guass 列主元消去法：

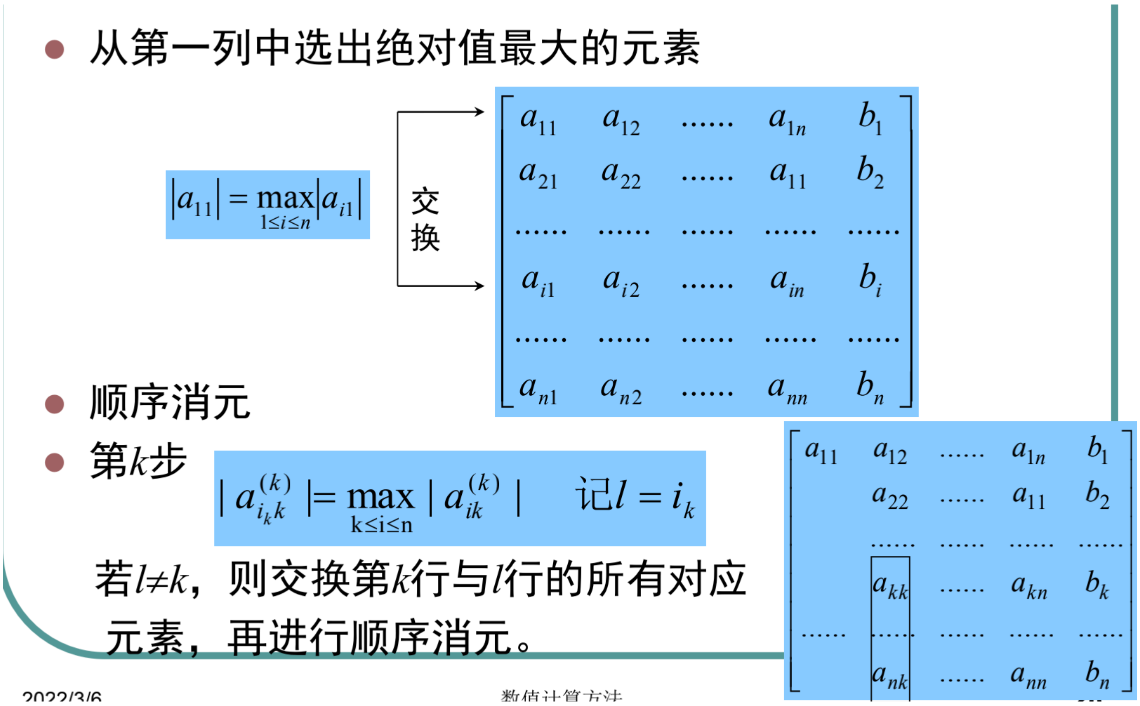


图 2: Guass 列主元消去法

Matlab 代码:

```
% 消元
jiaohuan=0;
for  i=1:n-1
    max=abs(a(i,i));
    m=i;
    for  j=i+1:n
        if  max<abs(a(j,i))
            max=abs(a(j,i));
            m=j; % 寻找列最大值
        end
    end
    if(m~=i)
        jiaohuan=jiaohuan+1;
        for  k=i:n
            c(k)=a(i,k);
            a(i,k)=a(m,k);
            a(m,k)=c(k);
        end
        d=b(i);
```

```
            b(i)=b(m);
            b(m)=d;
        end
        for k=i+1:n
            factor=a(k,i)/a(i,i);
            for j=i+1:n
                a(k,j)=a(k,j)-a(i,j)*factor;
            end
            b(k)=b(k)-b(i)*factor;
        end
end
% 回带
x(n)=b(n)/a(n,n);
for i=n-1:-1:1
    sum=b(i);
    for j=i+1:n
        sum=sum-a(i,j)*x(j);
    end
    x(i)=sum/a(i,i);
end
```

计算结果：以 $n=3$ 为例，$V_1=0, V_2=0.500000000000000, V_3=V_4=0.250000000000000, V_i(i\geq 5)=0.250000000000001$

经求解，n 取 1 到 5 中任何一个值时，其误差均非常小，最大的与真值相对误差也只有 4.884981308350689e-15。

由于本问题的特点，在对角线上的元素为正且在行、列最大，因此在原始 Guass 消元时对角线上元素已经必定是列上最大的元素，不需要进行交换。在实际程序运行是，jiaohuan 变量的值为 0，也从实际上验证了不需要交换。因此，在之后的求解中，都不再进行列主元交换。

### 4.1.2 LU 分解法

将 $A$ 分解成 $A=LU$ 的形式：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}}_{\text{单位下三角矩阵}} \underbrace{\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}}_{\text{上三角矩阵}}$$

图 3: LU 分解

则方程转换为求解：

$$L(UX) = B \Rightarrow \begin{cases} LY = B \\ UX = Y \end{cases}$$

图 4: LU 分解

Matlab 代码：

```
%% 求解
% 矩阵分解
for k=1:n-1
    for i=k+1:n
        a(i,k)=a(i,k)/a(k,k);
        for j=k+1:n
            a(i,j)=a(i,j)-a(i,k)*a(k,j);
        end
    end
end
l=eye(n);
u=zeros(n,n);
for k=1:n
    for i=k:n
        u(k,i)=a(k,i);% 构造U
    end
end
for k=1:n
    for j=1:k-1
```

```
            l(k,j)=a(k,j);% 构造L
      end
end
% 回带求解
y(1)=b(1);
for i=2:n
      for j=1:i-1
            b(i)=b(i)-l(i,j)*y(j);
      end
      y(i)=b(i);
end
x(n)=y(n)/u(n,n);
for i=(n-1):-1:1
      for j=n:-1:i+1
            y(i)=y(i)-u(i,j)*x(j);
      end
      x(i)=y(i)/u(i,i);
end
```

计算结果：以 $n=3$ 为例，$V_1=0, V_2=0.500000000000000, V_3=V_4=0.250000000000000, V_i(i\geq 5)=0.250000000000001$ 以 $n=2$ 为例，其 L 矩阵和 U 矩阵分别为：

$$L=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -0.3333 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -0.3333 & -0.3636 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.2727 & -0.1143 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.3143 & -0.3023 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.2713 & -0.3889 & 1 & 0 \\ 0 & 0 & 0 & 0 & -0.2713 & -0.3889 & -0.8621 & 1 \end{bmatrix}$$

$$U=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3.6667 & -1.3333 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3.1818 & -0.3636 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.6857 & -1.1143 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 3.3488 & -1.3023 & -1.3023 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3.2222 & -2.7778 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8276 \end{bmatrix}$$

经求解，n 取 1 到 5 中任何一个值时，其误差均非常小，最大的与真值相对误差也只有 4.884981308350689e-15。

**由于 A 既不是对称正定矩阵，也不是三对角矩阵，因此无法使用 Cholesky 分解、Thomas 算法。**

### 4.2 迭代法

- 用某种极限过程去逐步逼近线性方程组精确解的方法。
- 迭代法具有占存储单元少，程序设计简单，原始系数矩阵在迭代过程中不变等优点，但计算工作量有时较大。适宜计算系数矩阵为稀疏矩阵的问题。
- 存在收敛性及收敛速度等问题，对方程组的系数矩阵有一定的要求，才能保证迭代过程的收敛。

$$A=\begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix}-\begin{bmatrix} 0 & & & \\ -a_{21} & 0 & & \\ \vdots & \vdots & \ddots & \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix}-\begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & 0 & \cdots & -a_{2n} \\ & & \ddots & \vdots \\ & & & 0 \end{bmatrix}=D-L-U$$

图 5: 迭代法

迭代法基本步骤:

- $A$ 分裂为 $A=M-N$
- 分裂阵 $M$：可选择的非奇异阵，$Mx=d$ 易于求解，$M$ 选为 $A$ 的某种近似
- $Ax=b\rightarrow Mx=Nx+b\rightarrow x=M^{-1}Nx+M^{-1}b$
- $x(0)$ 为初始向量，$x(k+1)=M^{-1}Nx(k)+M^{-1}b$
- 选取不同的 $M$ 阵就得到不同迭代法。

#### 4.2.1 Jacobi 方法

- 设 $A$ 为非奇异矩阵，且 $a_{ii}\neq 0$，选取 $M=D$ 和 $N=D-A=L+U$
- $x^{(k+1)}=Jx^{(k)}+f$，$J=D^{-1}(L+U)$，$f=D^{-1}b$

Matlab 代码:

```
%% 求解
D=diag(diag(a));
N=D-a;
G=inv(D)*N;
f=inv(D)*b;
i_max=1e10; % 最大迭代次数
x0=zeros(n,1);
dtol=5e-11;
for i=1:i_max
    x=G*x0+f;
    if norm(x-x0)/norm(x)<dtol
        break;
    else
    x0=x;
    end
end
```

表 1: Jacobi 方法结果

| $n$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 最大相对误差 | 1.6309e-10 | 1.2050e-09 | 2.9392e-09 | 5.3581e-09 | 8.4680e-09 |
| 迭代次数 | 72 | 336 | 773 | 1375 | 2136 |

#### 4.2.2 Gauss-Seidel 方法

- 选取 $M = D - L$ 和 $N = M - A = U$
- $x^{(k+1)} = Gx^{(k)} + f$，$G = (D-L)^{-1}U$，$f = (D-L)^{-1}b$

Matlab 代码:

```
%% 求解
D=diag(diag(a));
L=-tril(a,-1);
U=-triu(a,1);
G=inv(D-L)*U;
f=inv(D-L)*b;
i_max=1e10; % 最大迭代次数
x0=zeros(n,1);
```

```
dtol=5e-11;
for i=1:i_max
    x=G*x0+f;
    if norm(x-x0)/norm(x)<dtol
        break;
    else
        x0=x;
    end
end
```

表 2: Gauss-Seidel 方法结果

| $n$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 最大相对误差 | 6.2218e-11 | 5.4959e-10 | 1.3654e-09 | 2.5882e-09 | 4.0768e-09 |
| 迭代次数 | 38 | 174 | 400 | 711 | 1106 |

#### 4.2.3 SOR 方法

在高斯-赛得尔方法的基础上进行修改：

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\tilde{x}_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)})$$

图 6: SOR 方法

- $\omega$=1，SOR 方法即为 G-S 方法
- $0 < \omega < 1$，结果为当前迭代结果和上一次迭代结果的加权平均，称为低松弛方法。用于使得非收敛方程组收敛或者克服振荡加速收敛。
- $1 < \omega < 2$，超松弛方法

SOR 方法：

- 隐含假设：新值沿正确方向向真实解移动，但是移动的速度慢。
- 用于加速已知是收敛的方程组的收敛速度
- 根据经验确定 $\omega$ 值

Matlab 代码：

```matlab
%% 求解
D=diag(diag(a));
L=-tril(a,-1);
U=-triu(a,1);
w=1.78;
G=inv(D-L)*U;
f=inv(D-L)*b;
i_max=1e10; % 最大迭代次数
x0=zeros(n,1);
dtol=5e-11;
x=x0;
for i=1:i_max
    for j=1:n
        x(j)=x0(j)+w/a(j,j)*(b(j)-a(j,1:j-1)*x(1:j-1)-a
            (j,j:n)*x0(j:n));
    end
    if norm(x-x0)/norm(x)<dtol
        break;
    else
        x0=x;
    end
end
```

经过调试每种 $n$ 迭代次数最小时 $w$ 值及对应的误差和迭代次数如下：以 $n=5$ 为例，画出迭代次数关于 $w$ 变化的曲线如下：

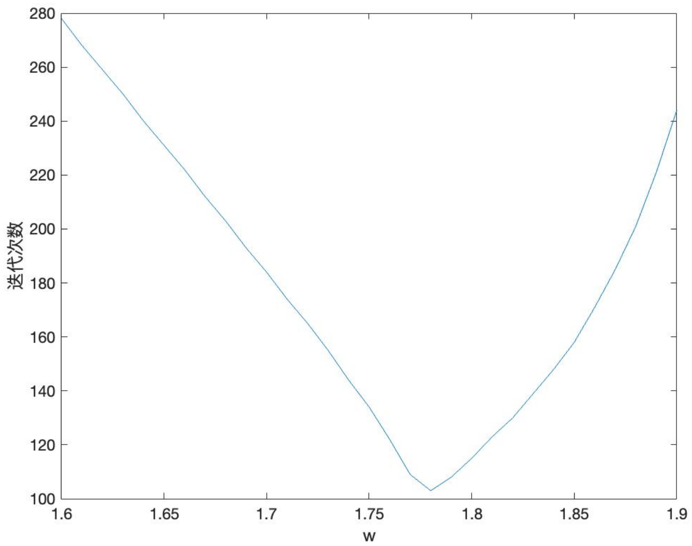


图 7: 迭代次数-$w$

可以看出，在 $w$ 取到某一个点时迭代次数最少，在此之外逐步增加。我们要做的就是通过调试找出每个方程对应的 $w$ 使得迭代次数最少。

表 3: SOR 方法结果

| $n$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $w$ | 1.20 | 1.52 | 1.65 | 1.73 | 1.78 |
| 最大相对误差 | 3.8994e-12 | 3.0131e-11 | 6.6287e-11 | 4.7729e-11 | 4.6872e-11 |
| 迭代次数 | 39 | 41 | 61 | 83 | 103 |

# 5 结果分析

## 5.1 直接法分析

对于 Guass 列主元消去法和 LU 分解法两种非迭代方法，由于 $n$ 取值较小，在本题中计算速度快，结果非常精确，是此问题合适的解法。

## 5.2 迭代法收敛性分析

然而当 $n$ 增大后直接法计算量将大大增加，迭代法将体现出更优的性能，因此迭代法也有很重要的作用。下面讨论迭代法求解的收敛速度及误差大小。在每种迭代法我设定的结束条件误差限均为 5e-11。然而，由于迭代次数不同，实际与真值误差仍然会有明显区别。

一阶定常迭代法收敛性的基本定理：设有方程组 $x = Gx + f$，有迭代法 $x(k+1) = Gx(k) + f$，则对任选初始向量 $x(0)$，迭代法收敛的充要条件是 $\rho(G) < 1$。经

实践认证，该定理符合实际情况：

表 4: $\rho$ 值

| $n$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Jacobi 方法 | 0.7287 | 0.9400 | 0.9746 | 0.9861 | 0.9912 |
| G-S 方法 | 0.5361 | 0.8841 | 0.9499 | 0.9724 | 0.9825 |

### 5.3 迭代法速度分析

下面以 $n = 3$ 为例对三种迭代方法进行对比：

表 5: 迭代方法对比

| 迭代方法 | 最大相对误差（与真值） | 迭代次数 |
|---|---|---|
| Jacobi 方法 | 2.9392e-09 | 773 |
| G-S 方法 | 1.3654e-09 | 400 |
| SOR 方法 | 6.6287e-11 | 61 |

从表中可以很明显地看出层次递进关系。Jacobi 方法迭代次数最多，与真值相对误差也最大；G-S 方法迭代次数中等，与真值相对误差中等；SOR 方法迭代次数最少，与真值相对误差最低。因此，SOR 方法在本题中是迭代法最佳的方法。

## 6 小结

本文对一个实际电路问题列写节点电压方程，将其转换为矩阵形式，从直接法和迭代法两类方法出发，考虑了实际问题的求解。同时，对迭代法中三种典型的方法优劣程度进行了对比分析，得出了一些有意义的结论。