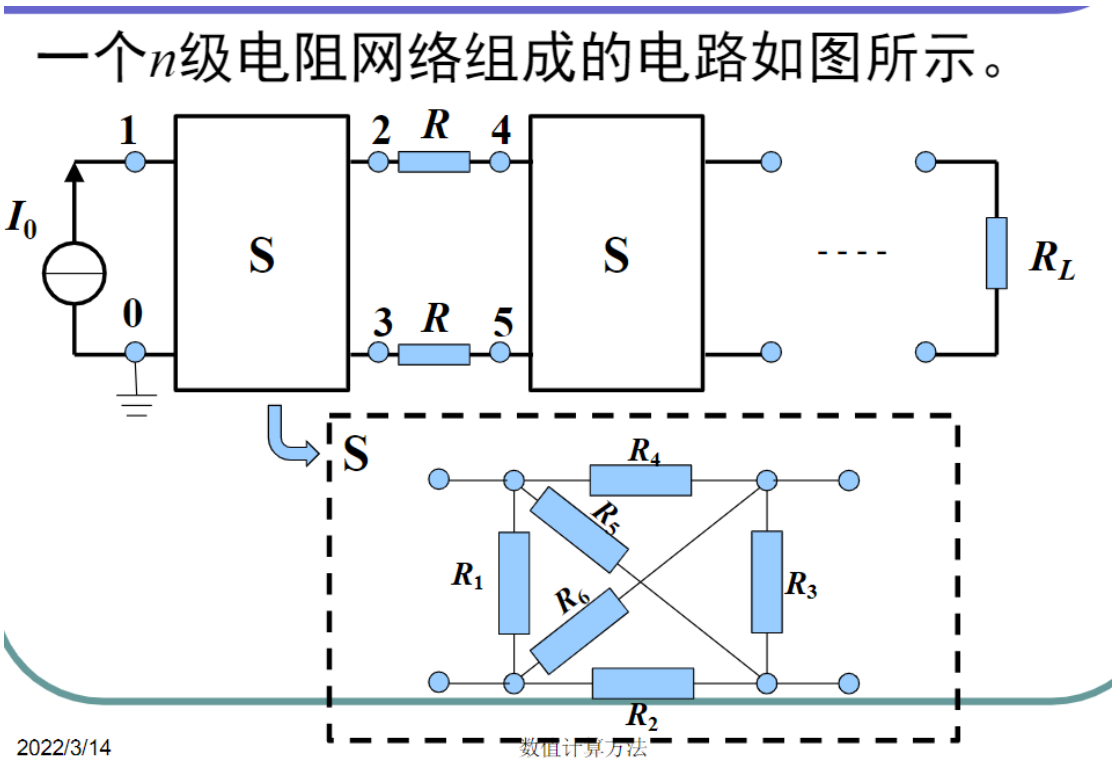
第三章报告

问题描述：



图中，I0 为恒流源，所有电阻阻值为 R，当 n>=1 时，节点数为 4n。

（1）由欧姆定律和基尔霍夫定律建立求解各节点电势 Vi 的线性代数方程组。

（2）若 I0 = 1A，R = 1Ω。请确定 n=1-5 时，每个节点的电势 Vi，请用不同的方法求解并进行对比

问题分析：

当 n=1 时，此时可列写电路方程如下：

节点 0：已知为 0

节点 1：$I_0+\frac{v_2-v_1}{R}+\frac{v_3-v_1}{R}+\frac{v_0-v_1}{R}=0$

节点 2：$\frac{v_1-v_2}{R}+\frac{v_0-v_2}{R}+\frac{v_3-v_2}{R}\times 2=0$

节点 3：$\frac{v_1-v_3}{R}+\frac{v_0-v_3}{R}+\frac{v_2-v_3}{R}\times 2=0$

当 n>1 时，可分为 3 段列写方程

节点 1：同上式即$I_0+\frac{v_2-v_1}{R}+\frac{v_3-v_1}{R}+\frac{v_0-v_1}{R}=0$

设现在处于第 i 段且第 i 段不为最后一段

节点 4*i-4：$\frac{v_{4i-5}-v_{4\cdot i-4}}{R}+\frac{v_{4i-1}-v_{4\cdot i-4}}{R}=\frac{v_{4i-4}-v_{4\cdot i-3}}{R}+\frac{v_{4i-4}-v_{4\cdot i-2}}{R}$

节点 4*i-3：$\frac{v_{4i-3}-v_{4\cdot i-6}}{R}=\frac{v_{4i-4}-v_{4\cdot i-3}}{R}+\frac{v_{4i-2}-v_{4\cdot i-3}}{R}+\frac{v_{4i-1}-v_{4\cdot i-3}}{R}$

节点 4*i-2: $\frac{v_{4i-2}-v_{4\cdot i-3}}{R} = \frac{v_{4i-4}-v_{4\cdot i-2}}{R}+\frac{v_{4i-1}-v_{4\cdot i-2}}{R}+\frac{v_{4i+1}-v_{4\cdot i-2}}{R}$

节点 4*i-1：$\frac{v_{4i-1}-v_{4\cdot i}}{R} + \frac{v_{4i-1}-v_{4\cdot i-2}}{R}+\frac{v_{4i-1}-v_{4\cdot i-3}}{R}+\frac{v_{4i-1}-v_{4\cdot i-4}}{R} = 0$

设现在处于第 i 段且 i 处在最后一段

节点 4*i-4 同上：$\frac{v_{4i-5}-v_{4\cdot i-4}}{R} + \frac{v_{4i-1}-v_{4\cdot i-4}}{R}=\frac{v_{4i-4}-v_{4\cdot i-3}}{R}+\frac{v_{4i-4}-v_{4\cdot i-2}}{R}$

节点 4*i-3 同上：$\frac{v_{4i-3}-v_{4\cdot i-6}}{R} = \frac{v_{4i-4}-v_{4\cdot i-3}}{R}+\frac{v_{4i-2}-v_{4\cdot i-3}}{R}+\frac{v_{4i-1}-v_{4\cdot i-3}}{R}$

节点 4*i-2: $\frac{v_{4i-2}-v_{4\cdot i-3}}{R} = \frac{v_{4i-4}-v_{4\cdot i-2}}{R}+\frac{v_{4i-1}-v_{4\cdot i-2}}{R}+\frac{v_{4i-1}-v_{4\cdot i-2}}{R}$

节点 4*i-1：$\frac{v_{4i-1}-v_{4\cdot i-2}}{R} + \frac{v_{4i-1}-v_{4\cdot i-2}}{R}+\frac{v_{4i-1}-v_{4\cdot i-3}}{R}+\frac{v_{4i-1}-v_{4\cdot i-4}}{R} = 0$

将上述方程组化为矩阵可得：
n=1 时，矩阵为：

A = $\begin{bmatrix} 3 & -1 & -1 \\ -1 & 4 & -2 \\ -1 & -2 & 4 \end{bmatrix}$ b = $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

当 n>1 时，
A 的第一行依旧为[3,-1,-1],其余为 0
当处于第 i 段且不为最后一段时
则可获得一个第 4i-4 行到 4i-1 行的矩阵，矩阵列从 4i-6 到 4i+1 列

$$\begin{pmatrix} 0 & -1 & 4 & -1 & -1 & -1 & 0 & 0 \\ -1 & 0 & -1 & 4 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 4 & -1 & 0 & -1 \\ 0 & 0 & -1 & -1 & -1 & 4 & -1 & 0 \end{pmatrix}$$

当 i 为最后一段时
则矩阵最后两行如下：
包含第 4i-4 到 4i-1 列元素，其余位置元素为 0

$$\begin{pmatrix} -1 & -1 & 4 & -2 \\ -1 & -1 & -2 & 4 \end{pmatrix}$$

通过以上步骤，即可得到系数矩阵 A，而 b 为一个第一个元素为 1，其余元素为 0 的 4i-1*1 的列向量。
方法使用：在求解这些电势大小的线性方程组，下面的程序使用了 Guass 消去法，Guass 列主元消去法，Guass-Jordan 法，DooLittle 分解法，Cholesky 分解法，雅可比迭代法以及 Guass-Seidel 法求解，都能很好的完成任务得到答案。

Matlab 程序:

建立矩阵 CrateMatrix(n,value) n 为电阻网络数，value 为 I_0R 的值
函数返回 A 为系数矩阵，b 为方程右侧值

```
function [A,b] = CreateMatrix(n,value) %建立矩阵
    A = zeros(4*n-1);
    b = zeros(4*n-1,1);
    b(1) = value;
    if n == 1%当n为1时
        A = [3,-1,-1;-1,4,-2;-1,-2,4];
    else%当n>1时
       A(1:1,1:3) = [3,-1,-1]; %矩阵的第一行
        for i = 1:1:n
           if i == 1
               A(2:3,1:5) = [-1,4,-1,0,-1;-1,-1,4,-1,0];
           else
              A(4*i-4:4*i-3, 4*i-6:4*i-1) = [0,-1,4,-1,-1,-1;-1,0,-1,4,-1,-1];%矩阵在每一段电阻网络列写方程时的前两行
              if i ~= n
                   A(4*i-2:4*i-1, 4*i-4:4*i+1) = [-1,-1,4,-1,0,-1;-1,-1,-1,4,-1,0];%不是最后一个电阻网络
              else
                   A(4*i-2:4*i-1, 4*i-4:4*i-1) = [-1,-1,4,-2;-1,-1,-2,4];%最后一个电阻网络
              end
           end
       end
    end
end
```

下面分别为 n 从 1 到 5 所构建的矩阵

A =

3	-1	-1
-1	4	-2
-1	-2	4

A =

3	-1	-1	0	0	0	0
-1	4	-1	0	-1	0	0
-1	-1	4	-1	0	0	0
0	0	-1	4	-1	-1	-1
0	-1	0	-1	4	-1	-1
0	0	0	-1	-1	4	-2
0	0	0	-1	-1	-2	4

A =

3	-1	-1	0	0	0	0	0	0	0	0
-1	4	-1	0	-1	0	0	0	0	0	0
-1	-1	4	-1	0	0	0	0	0	0	0
0	0	-1	4	-1	-1	-1	0	0	0	0
0	-1	0	-1	4	-1	-1	0	0	0	0
0	0	0	-1	-1	4	-1	0	-1	0	0
0	0	0	-1	-1	-1	4	-1	0	0	0
0	0	0	0	0	0	-1	4	-1	-1	-1
0	0	0	0	0	-1	0	-1	4	-1	-1
0	0	0	0	0	0	0	-1	-1	4	-2
0	0	0	0	0	0	0	-1	-1	-2	4

A =

3	-1	-1	0	0	0	0	0	0	0	0	0	0
-1	4	-1	0	-1	0	0	0	0	0	0	0	0
-1	-1	4	-1	0	0	0	0	0	0	0	0	0
0	0	-1	4	-1	-1	-1	0	0	0	0	0	0
0	-1	0	-1	4	-1	-1	0	0	0	0	0	0
0	0	0	-1	-1	4	-1	0	-1	0	0	0	0
0	0	0	-1	-1	-1	4	-1	0	0	0	0	0
0	0	0	0	0	0	-1	4	-1	-1	-1	0	0
0	0	0	0	0	0	0	-1	4	-1	-1	0	0
0	0	0	0	0	0	0	-1	-1	4	-1	0	0
0	0	0	0	0	0	0	0	-1	-1	4	-1	-1
0	0	0	0	0	0	0	0	0	-1	-1	4	-2
0	0	0	0	0	0	0	0	0	0	-1	-1	-2	4

A =

3	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	4	-1	0	-1	0	0	0	0	0	0	0	0	0	0
-1	-1	4	-1	0	0	0	0	0	0	0	0	0	0	0
0	0	-1	4	-1	-1	-1	0	0	0	0	0	0	0	0
0	-1	0	-1	4	-1	-1	0	0	0	0	0	0	0	0
0	0	0	-1	-1	4	-1	0	-1	0	0	0	0	0	0
0	0	0	-1	-1	-1	4	-1	0	0	0	0	0	0	0
0	0	0	0	0	0	-1	4	-1	-1	-1	0	0	0	0
0	0	0	0	0	0	0	-1	4	-1	-1	0	0	0	0
0	0	0	0	0	0	0	-1	-1	4	-1	0	-1	0	0
0	0	0	0	0	0	0	0	-1	-1	4	-1	0	0	0
0	0	0	0	0	0	0	0	0	-1	-1	4	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	-1	-1	4	-1	-1
0	0	0	0	0	0	0	0	0	0	0	-1	-1	4	-2
0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	-2	4

可以发现这些矩阵都是对称的

Guass 顺序消去法 Guass.m 传入 A，n，b

```
%原始高斯消去法

function [X] = Guass(Matrix,n,b)
    %消去过程

    for k = 1:n-1
        for i = k+1:n
            factor = Matrix(i,k)/Matrix(k,k);
            for j  = k+1:n
                Matrix(i,j) = Matrix(i,j) - factor * Matrix(k,j);
            end
            b(i) = b(i) - factor * b(k);
        end
    end

    %回代过程
    X(n) = b(n)/Matrix(n,n);
    for i = n-1:-1:1
        sum = b(i);
        for j = i+1:n
            sum = sum - Matrix(i,j)*X(j);
        end
        X(i) = sum/Matrix(i,i);
    end
end
```

带入 n=1，2，3，4，5 后结果为

```
X1 =

    0.5000    0.2500    0.2500


X1 =

    0.5000    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500


X1 =

    0.5000    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500


X1 =

  列 1 至 11

    0.5000    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500

  列 12 至 15

    0.2500    0.2500    0.2500    0.2500
```

```
X1 =

  列 1 至 11

    0.5000    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500

  列 12 至 19

    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500
```

此方法的时间复杂度为：$\frac{2}{3}n^3 + O(n)$

当方程组规模变大时，计算时间增加很快。浮点操作个数增加接近维数增加量的三次方,其中大部分时间消耗在消去步骤中。

列主元高斯消去法 Guass_Column.m 传入参数 A，n，b

高斯列主元消去法是对高斯顺序消去法的改进，原始的高斯消去法每一个结果都依赖于前面的结果，如果前面的结果很小，这就会引起其他元素数量级的剧增和摄入误差的增长，导致计算结果不可靠，甚至计算不能进行下去（上溢）。

```
%列主元高斯消去法
function [X] = Guass_column(Matrix,n,b)

    %消去过程
    for k = 1:n-1
        a = Matrix(k:n,k); %取出矩阵的第i列向量
        [~,index] = max(abs(a));%得到向量中绝对值最大的元素下标
        index = index + k - 1;%将向量下标换算为矩阵的行
        if index ~= k%比对index是否为现在循环的行数，如果不是则把两行的元素进行交换
            Matrix([k index],:) = Matrix([index k],:);%交换Matrix矩阵的行元素
            temp = b(k);%交换b向量的元素
            b(k) = b(index);
            b(index) = temp;
        end
        for i = k+1:n %顺序消元法
            factor = Matrix(i,k)/Matrix(k,k);
            for j  = k+1:n
                Matrix(i,j) = Matrix(i,j) - factor * Matrix(k,j);
            end
            b(i) = b(i) - factor * b(k);
        end
    end
    %回代过程
    X(n) = b(n)/Matrix(n,n);
    for i = n-1:-1:1
        sum = b(i);
        for j = i+1:n
            sum = sum - Matrix(i,j)*X(j);
        end
        X(i) = sum/Matrix(i,i);
    end
end
```

相比于原始的高斯消去法，在消元过程中额外多了按列选主元的步骤和交换行步骤，其余流程是一致的。时间复杂度在原来高斯消去法的程度上增加了 2 个 O(n)。

此方法由于$\frac{|a_{ik}|}{|a_{kk}|} \leq 1$（i=k+1，…n），列主元消去法有利于控制误差的传播，故具有较好的数值稳定性。
运行结果与高斯顺序消去法一样 V1 = 0.5V，其余节点的 V 均为 0.25V.

高斯约当法 Guass_Jordan.m 传入参数 A，n，b

高斯约当法是高斯消去法的变形，将方程钟所有的未知数都消去，除以主元进行标准化，最终消去的结果是一个单位阵，此方法只需要消去，而不需要回代。

```
function [X] = Guass_Jordan(A,n,b)
    A = [A b];%增广矩阵
    for i=1:n
        x = A(i,i);
        for j = 1:n+1
            A(i,j) = A(i,j)/x;%除以主元
        end

        for j = 1:n%标准化
            y = A(j,i);
            if j~= i
                for k = 1:n+1
                    A(j,k) = A(j,k) - y*A(i,k);
                end
            end
        end
    end
    X = A(:,n+1);%获得增广矩阵的最后一列即为所求
end
```

此方法的时间复杂度为$\frac{n^3}{2}+n^2-\frac{n}{2}$，当 n 增加时，可表示为$\frac{n^3}{2}+O(n^2)$。

高斯约当法的运行结果与前面两种方法相同。

LU 分解

Doolittle 分解法：Doolittle.m 传入 A,n,b
当系数矩阵 A 的各阶顺序主子式均部位 0 时，Doolittle 分解可以实现。此线性方程组的系数矩阵各界顺序主子式均不为 0，所以 Doolittle 分解可以实现。

Doolittle 分解即：把 A 分解为 LU 的形式，L 为一个对角线元素为 1 的下三角矩阵，而 U 为一个上三角矩阵。然后通过 LY=b 先将 Y 求出，再代入 UX = Y，求解 X。

```
function [X] = Doolittle(A,n,b)
    X = zeros(n,1);
    %判断A的各阶顺序主子式是否为0，若都不为0，则满足LU分解条件
    for i=1:n
        if det(A(1:i,1:i)) == 0
            fprintf("A不满足LU分解条件");
            return;
        end
    end

    %A的LU分解
    L = diag(ones(n,1));
    U = zeros(n,n);
    for k = 1:n
        for i = k:n
            sum1 = 0;
            sum2 = 0;
            for j = 1:k-1
                sum1 = sum1 + L(k,j)*U(j,i);
                sum2 = sum2 + L(i,j)*U(j,k);
            end
            U(k,i) = A(k,i) - sum1;
            L(i,k) = (A(i,k)-sum2)/U(k,k);
        end
    end
```

下一页有后续程序

```
    %代入
    %向前带入LY=b
    Y = zeros(n,1);
    for k=1:n
        sum = 0;
        for j = 1:k-1
            sum = sum + L(k,j)*Y(j);
        end
        Y(k) = b(k) - sum;
    end

    %向后带入UX=Y
    for k=n:-1:1
        sum = 0;
        for j = k+1:n
            sum = sum + U(k,j)*X(j);
        end
        X(k) = (Y(k) - sum)/U(k,k);
    end



end
```

Doolittle 分解的结果也与前三种方法相同。

Doolittle 分解的时间复杂度为 分解过程：$\frac{n^3-n}{3}$ 代入：n^2，随着 n 的增加，此方法的时间复杂度与高斯消去法相当。

Doolittle 分解法是从矩阵 A 的元素直接由关系式 A = LU 确定 L 和 U 的元素，不必像 Guass 消去法那样计算那些中间结果，高斯消去法求解方程组时，右端项必须提前知道，而三角分解不需要。当已实现 A = LU 的分解后，解具有相同系数矩阵的方程组 AX=bi 很方便，只需要求解两个三角形方程组，用 n^2次乘除计算即可。

对称正定矩阵的平方根法(Cholesky 法)　Cholesky.m　传入 A,n,b

Cholesky 法：若 A 为 n 阶是对称正定矩阵，则必存在非奇异下三角矩阵 L，使得 $A=LL'$，并且当 L 的主对角元均为正数时分解唯一。此线性方程组的系数矩阵可满足上述条件，所以可以使用 Cholesky 分解法。

```
function [X] = Cholesky(A,n,b)
    %判断是否为对称正定矩阵
    X = zeros(n,1);
    if A ~= A'
        return;
    end
    lamuda = eig(A);
    for i = 1:size(lamuda)
        if lamuda(i) <= 0
            return;
        end
    end

    %LL'分解
    L = zeros(n,n);
    for k = 1:n
        sum = 0;
        for i = 1:k-1
            sum = sum + L(k,i)*L(k,i);
        end
        L(k,k) = sqrt(A(k,k) - sum);
        for j = k+1:n
            sum = 0;
            for i = 1:k-1
                sum = sum + L(k,i)*L(j,i);
            end
            L(j,k) = (A(j,k)-sum)/L(k,k);
        end
    end
```

下一页有后续程序

```
    %带入
    Y = zeros(n,1);
    for k = 1:n
        sum = 0;
        for j = 1:k-1
            sum = sum + L(k,j)*Y(j);
        end
        Y(k) = (b(k) - sum)/L(k,k);
    end

    for k = n:-1:1
        sum = 0;
        for j = k+1:n
            sum = sum + L(j,k)*X(j);
        end
        X(k) = (Y(k) - sum)/L(k,k);
    end

end
```

程序的分解过程计算量小，只需约$\frac{n^3}{6}$次乘除法，大约是高斯消去法和 Doolittle 分解法的二分之一，而且数值稳定，存储量小。
但是由于存在开方运算，所以可能会出现根号下负数。

程序运行结果也和前述方法均一致。

迭代法

雅可比迭代法 Jacobi.m 传入 A,n,b

设 A 为非奇异矩阵，且 $a_{ii}\neq 0$，选 M=D,N=D-A = L+U。
通过$x^{(k+1)} = Jx^k + f$， $J = D^{-1}(L+U)$ ，$f = D^{-1}b$来迭代得到结果
雅可比迭代法收敛条件为：ρ（J）<1。

雅可比迭代法，每迭代一次主要是计算一次矩形乘向量，计算过程中，原始数据 A 保持不变，计算中需要两组工作单元来保存 X。

```
%雅可比迭代法（异步迭代法）
function [X] = Jacobi(Matrix,n,b)
    X = zeros(n,1);
    ea = ones(n,1);
    M = diag(diag(Matrix));
    J = inv(M)*(M-Matrix);
    if max(eig(J)) > 1
        fprintf("雅可比迭代法不收敛");
        return
    end
    f = inv(M)*b;
    while check(ea,n)%check函数检测是否达到容限，迭代过程如下
        X1 = J*X + f;
        for j = 1:n
            ea(j) = abs((X1(j) - X(j))/X1(j));
        end
        X = X1;
    end
end
```

雅可比迭代法也能成功的完成运算，结果与前述结果一致。

G-S 迭代法 Guass_Seidel.m 代入 A，n，b
选取 M=D-L，N=M-A=U
迭代$x^{(k+1)} = Gx^{(k)} + f$，G=$(D-L)^{-1}U$，f=$(D-L)^{-1}b$
G-S 迭代法收敛条件为：ρ（G）<1.

G-S 迭代法每一次迭代也主要是计算一次矩阵乘向量，计算$x^{(k+1)}$的第 i 个分量$x_i^{(k+1)}$时利用已计算出的最新分量。计算中只需要一组工作单元来保存 X。

```
%高斯-赛德尔迭代法（异步迭代法）
function [X] = Guass_Seidel(Matrix,n,b)
    X = zeros(n,1);
    ea = ones(n,1);
    M = tril(Matrix);
    G = inv(M)*(M-Matrix);
    if max(eig(G)) > 1
        fprintf("高斯-赛德尔迭代法不收敛");
        return
    end
    f = inv(M)*b;
    while check(ea,n)%check函数检测是否达到容限，迭代过程如下
        for j = 1:n
            sum = 0;
            for k=1:n
                sum = sum +  G(j,k)*X(k);
            end
            temp = X(j);
            X(j) = sum + f(j);
            ea(j) = abs((X(j) - temp)/X(j));
        end
    end
end
```

G-S 迭代法也能成功的完成运算，结果与前述结果一致

SOR 逐次超松弛迭代法：

此方法为在 G-S 方法上的修改，引入 w 松弛因子概念：

当 w=1 时，SOR 方法即为 G-S 方法

0<w<1，结果为当前迭代结果和上一次迭代结果的加权平均法。

1<w<2,超松弛方法

隐含假设：新值沿正确方向向真实解移动，但是移动速度很慢

用于加速已知是收敛的方程组的收敛速度

根据经验确定 w 值。

文件 导航 编辑 断点

```
%SOR（逐次超松弛迭代法）
function [X] = SOR(A,n,b)
    X = zeros(n,1);
    ea = ones(n,1);
    w = 1.5;%松弛因子

    while check(ea,n)%check函数检测是否达到容限，迭代过程如下
        for j = 1:n
            sum = 0;
            for k=1:n
                sum = sum + A(j,k)*X(k);
            end
            temp = X(j);
            X(j) = (b(j) - sum)*w/A(j,j) + X(j);
            ea(j) = abs((X(j) - temp)/X(j));
        end
    end
```

SOR 方法也得到了和上述方法相同的结果。

SOR 方法的矩阵形式如下：$\boldsymbol{x^{(k+1)} = Gwx^{(k)} + f}$

其中$\boldsymbol{G\omega = (D - \omega L)^{-1}[(1 - w)^{D} + wU]}$ $\boldsymbol{f = \omega(D - \omega L)^{-1}b}$

SOR 方法收敛充要条件为ρ(Gw)<1

下面进行几种迭代法的对比：

以上程序均在所有解的近似相对误差$\varepsilon_a = \frac{|x^{(k+1)} - x^{(k)}|}{|x^{(k+1)}|} < 5e-10$ 时结束

雅可比迭代法

矩阵 n×n	3	7	11	15	19
迭代次数	66	306	701	1241	1921

G-S 法

矩阵 n×n	3	7	11	15	19
迭代次数	29	115	260	459	710

从上述表格可以看出，G-S 方法的迭代次数要比 Jacobi 法更少，运算更快。

SOR 法松弛因子探讨：

松弛因子 w	迭代矩阵				
	n = 3	n = 7	n = 11	n = 15	n = 19
1.1	27	130	298	530	823
1.2	18	105	244	435	676
1.3	23	82	196	352	550
1.4	29	62	154	280	439
1.5	38	40	116	216	341
1.6	51	49	79	156	252
1.7	72	70	68	96	168
1.8	114	111	108	105	107
1.9	240	232	228	227	225

从以上表格可以看出，不同大小不同疏密程度的矩阵的最适松弛因子 w 是不同的，n=3 时，w=1.3 最好，n>3 之后，矩阵变得更加疏密，n=7 时，w=1.5 最好；n=11 和 n=15 时，w=1.7 最好，而 n=19 时，w=1.8 最好。
SOR 迭代法，选择好最适 w 之后，能够大幅度的降低 G-S 法的迭代次数。