数值代数　第 2 次上机作业

李佳 2100010793

1　问题描述

利用五点差分格式近似求解 Dirichlet 边界的 Poisson 方程:

$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial^2 u}{\partial y^2} = f,\ (x,y) \in \Omega \\ \qquad\qquad\qquad u = 0,\ (x,y) \in \partial\Omega \end{cases}$$

其中区域 $\Omega = (0,1) \times (0,1)$, 源函数 $f = \sin(\pi x)\sin(\pi y)$.

对区域 Ω 进行均匀剖分, 沿 x 轴和 y 轴均以 $h = h_x = h_y = \frac{1}{N}$ 的间距进行等分. 记 $x_i = (i-1)h_x, y_i = (i-1)h_y$, 以及 $u_{i,j} = u(x_i, y_j), f_{i,j} = f(x_i, y_j), 1 \le i, j \le N+1$.

当 (x_i, y_j) 为 Ω 内部点时, 即 $2 \le i, j \le N$, 利用二阶中心差分格式近似 $\frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}$:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} + u_{i-1,j} - u_{i,j}}{h_x^2}, \frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1} + u_{i,j-1} - u_{i,j}}{h_y^2},$$

再由边界条件, 知:

$$u_{1,j} = u_{N+1,j} = u_{i,1} = u_{i,N+1} = 0,\ 1 \le i, j \le N+1.$$

令 $U_{i,j}$ 为 $u_{i,j}$ 的数值近似, 可以得到如下方程:

$$4U_{i,j} - U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1} = \frac{f_{i,j}}{N^2},\ 2 \le i, j \le N;$$

$$u_{1,j} = u_{N+1,j} = u_{i,1} = u_{i,N+1} = 0,\ 1 \le i, j \le N+1.$$

其中需要求解未知数 $U_{i,j}, 2 \le i, j \le j$. 将 $U_{i,j}, f_{i,j}$ 均按列排成向量:

$$X = (U_{2,2}, U_{2,3}, ..., U_{2,N}, U_{3,2}, ..., U_{N,N})^T,\ F = (f_{2,2}, f_{2,3}, ..., f_{2,N}, f_{3,2}, ..., f_{N,N})^T,$$

从而得到线性方程组

$$AX = b,$$

其中

$$b = \frac{1}{N^2} F,$$

$$A=\begin{bmatrix} M & -I & & & \\ -I & M & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & M & -I \\ & & & -I & M \end{bmatrix}_{(N-1)^2\times(N-1)^2}, M=\begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}_{(N-1)^2\times(N-1)^2}$$

对于 $N=32,64,128,256,512$ 时, 分别用 LDL^T 方法和带状 Guass 方法求解上述线性方程组, 并比较、分析各情况、方法的计算求解时间.

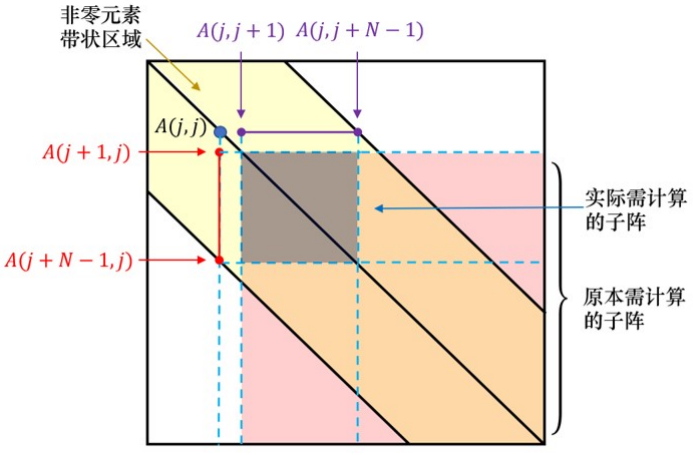
2 数值方法

2.1 一般的 LDL^T 方法

我们记矩阵规模 $(N-1)^2=n$. 注意到矩阵 A 为对称矩阵, 顺序主子式均为正, 知 A 为对称正定矩阵, 可通过 LDL^T 法进行分解, 再依次求解三角矩阵对应的线性方程组 $Lz=b, Dy=z, L^Tx=y$. 该方法的工作量为 $\frac{1}{3}n^3=\frac{1}{3}N^6$.

2.2 带状 Guass 方法

A 的顺序主子式均为正, 因此对 A 可以直接进行 LU 分解. 注意到, 矩阵 A 仅在主对角线和上下各 $N-1$ 条对角线的带状区域上有非 0 元素, 进行一般的 Guass 消元过程中, 只有图示部分的元素可能发生改变. 这就是说, 在前 $n-(N-1)$ 次 Guass 消元中, 不需要对全部的下三角矩阵 $A(i+1:n, i+1:n)$ 操作, 只需计算 $A(i+1:i+N-1, i+1:i+N-1)$ 的变化即可. 这大约可以将工作量压缩为原来的 $(\frac{N-1}{n})^2=\frac{1}{(N-1)^2}$. 工作量大约可达到 $O(N^4)$.



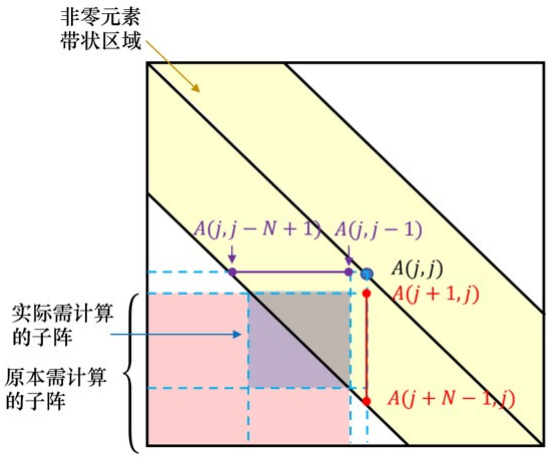
另一方面, 求解三角矩阵对应的线性方程组 $Ly=b, Ux=y$ 时, 通过上述方法求解的 L, U 也仅在主对角线和上下各 $N-1$ 条对角线上有非 0 元素, 因此求解的代入过程也不需要将向量 b 中的全体元素进行更新,(以解 $Ly=b$ 为例) 第 i 步 $(i\le n-(N-1))$ 代入只需计算 $b(i+1:i+N-1)$ 的变化. 因此每一步三角形方程组求解也可以将原有的工作量 n^2 压缩为原来的 $\frac{1}{N-1}$. 工作量大约可达到 $O(N^3)$.

2.3 LDL^T 带状消去方法

在实际进行 LDL^T 方法求解过程中, 发现除对角线附近的带状区域外, 其他区域仍然为 0. 事实上, 在每一次求解 $l_{ij}, i = j+1, j+2, ..., n$ 的过程中, 计算公式为:

$$A(j+1:n,j) = (A(j+1:n,j) - A(j+1:n,1:j-1)v(1:j-1))/A(j,j)$$

$N-1 < j \leq n-(N-1)$ 时, 如图所示, 矩阵 $A(j+1:n, 1:j-1)$ 仅有右上角非 0, 因此作减法、除法的过程仅有 $A(j+1:j+N-1, j)$ 发生变化, 从而该列只有带状区域内有非 0 元素. 由归纳法, 易知只有带状区域有非 0 元素.



因此,$N-1 < j \leq n-(N-1)$ 时, 计算 L 时只需计算 $l_{ij}, i = j+1, j+2, ..., j+N-1$. 同理可知 (如图) 计算 $v(i)$ 时只需计算 $v(i), i = j-N+1, j-N+2, ..., j-1$. 这大约也可以将工作量压缩为原来的 $\frac{1}{(N-1)^2}$. 工作量大约可达到 $O(N^4)$. 解三角形方程组如 2.2 中可进行类似的优化求解.

3 理论分析结果

从对三种方法的工作量估计来看, 问题规模 N 每增大 2 倍, 一般的 LDL^T 方法计算时间增加 $2^6 = 64$ 倍, 带状 Guass 方法和 LDL^T 带状方法各增大 $2^4 = 16$ 倍, 一般的 LDL^T 方法与其余两种方法的耗时之比在 $N = 512$ 时, 与 $N = 32$ 时相比将扩大 $16^2 = 256$ 倍. 另外,LDL^T 带状方法与带状 Guass 方法相比, 计算的元素仅在下三角, 计算耗时理论上是带状 Guass 方法的 $\frac{1}{2}$.

4 具体算法实现

4.1 一般的 LDL^T 方法

```
%% Step 1: LDL^T 分解
for j = 1:n
    for i = 1:j-1
        v(i) = A(j,i)*A(i,i);
    end
    A(j,j) = A(j,j) - A(j,1:j-1)*v(1:j-1);
    A(j+1:n,j) = (A(j+1:n,j) - A(j+1:n,1:j-1)*v(1:j-1))/A(j,j);
end

%% Step 2: 解三角形方程组 Lz = b, Dy = z, L^Tx = y
for i = 1:n-1  % 前代法解 Lz = b
    b(i+1:n) = b(i+1:n) - b(i)* A(i+1:n,i);
end
for i = 1: n   % 解 Dy = z
    b(i) = b(i)/ A(i,i);
end
for j = n:-1:2 % 回代法解 L^Tx = y
    b(1:j-1) = b(1:j-1) - b(j)* A(j,1:j-1)';
end
```

4.2 带状 Guass 方法

```
%% Step 1: 带状 Guass 消去
for i = 1:n-(N-1)      % 前面进行带状高斯消去
    A(i+1:i+N-1,i) = A(i+1:i+N-1,i)/A(i,i); % 只需计算 L 的(N-1)行
    A(i+1:i+N-1,i+1:i+N-1) = A(i+1:i+N-1,i+1:i+N-1) - A(i+1:i+N-1,i)*A(i,i+1:i+N-1); % 只需更新(N-1)*(N-1)的子阵
end
for i = n-(N-1)+1:n-1 % 后面规模更小进行一般的高斯消去
    A(i+1:n,i) = A(i+1:n,i)/A(i,i);
    A(i+1:n,i+1:n) = A(i+1:n,i+1:n) - A(i+1:n,i)*A(i,i+1:n);
end

%% Step 2: 解三角形方程组 Ly = b, Ux = y
for i = 1:n-N+1      % 前代法解 Ly = b 此时只需更新向量 b 的(N-1)行
    b(i+1:i+N-1) = b(i+1:i+N-1) - b(i)* A(i+1:i+N-1,i);
end
for i = n-N+2:n-1    % 此时进行正常的前代法求解
    b(i+1:n) = b(i+1:n) - b(i)* A(i+1:n,i);
end
for j = n:-1:N       % 回代法解 Ux = y 此时只需更新向量 b 的(N-1)行
    b(j) = b(j)/A(j,j);
    b(j-N+1:j-1) = b(j-N+1:j-1) - b(j)* A(j-N+1:j-1,j);
end
for j = N-1:-1:2     % 此时进行正常的回代法求解
    b(j) = b(j)/A(j,j);
    b(1:j-1) = b(1:j-1) - b(j)* A(1:j-1,j);
end
b(1) = b(1)/A(1,1);
```

4.3 LDL^T 带状消去方法

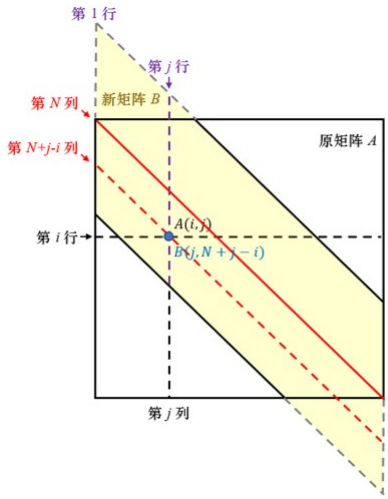
```
%% Step 1: LDL^T 带状消去
for j = 1:n
    if j <= N-1           % 前(N-1)列的计算
        for i = 1:j-1
            v(i) = A(j,i)*A(i,i);
        end
        A(j,j) = A(j,j) - A(j,1:j-1)* v(1:j-1);
        % 只需计算 L 在该列中的 (N-1) 行
        % 此处的处理是为尽量运用矩阵向量运算 , 尽管矩阵是上三角
        % 在第 4 节特殊存储方式的处理中 , 只能用标量计算
        % 但充分利用了上三角性质 , 可以减少运算 . 基本过程类似 , 不再赘述
        A(j+1:j+N-1,j) = (A(j+1:j+N-1,j) - A(j+1:j+N-1,1:j-1)*v(1:j-1))/A(j,j);
    elseif j > n-(N-1)   % 后(N-1)列的计算
        for i = j-N+1:j-1
            v(i-(j-N)) = A(j,i)*A(i,i);  % 向量 v 此时只需计算(N-1)个分量
        end
        A(j,j) = A(j,j) - A(j,j-N+1:j-1) * v(1:N-1);
        A(j+1:n,j) = (A(j+1:n,j) - A(j+1:n,j-N+1:j-1)*v(1:N-1))/A(j,j);
    else                  % 中间列的计算
        for i = j-N+1:j-1
            v(i-(j-N)) = A(j,i)*A(i,i);  % 向量 v 此时只需计算(N-1)个分量
        end
        A(j,j) = A(j,j) - A(j,j-N+1:j-1)*v(1:N-1);
        A(j+1:j+N-1,j) = (A(j+1:j+N-1,j) - A(j+1:j+N-1,j-N+1:j-1)*v(1:N-1))/A(j,j);
        % 只需计算 L 在该列中的(N-1)行
    end
end

%% Step 2: 解三角形方程组 Lz = b, Dy = z, L^Tx = y
for i = 1: n-N+1 % 前代法解 Lz = b
    b(i+1:i+N-1) = b(i+1:i+N-1) - b(i)* A(i+1:i+N-1,i);  % 此时只需更新向量 b 的(N-1)行
end
for i = n-N+2: n-1
    b(i+1:n) = b(i+1:n) - b(i)* A(i+1:n,i);
end
for i = 1: n      % 解 Dy=z
    b(i) = b(i)/ A(i,i);
end
for j = n: -1: N % 回代法解 L^Tx = y
    b(j-N+1:j-1) = b(j-N+1:j-1) - b(j)* A(j,j-N+1:j-1)'; % 此时只需更新向量 b 的(N-1)行
end
for j = N-1: -1: 2
    b(1:j-1) = b(1:j-1) - b(j)* A(j,1:j-1)';
end
b(1) = b(1)/A(1,1);
```

4.4 $N = 256,512$ 时的特殊处理

由于 $N = 256,512$ 时存储整个矩阵所需的内存过大, 因此对矩阵 A 采取了不一样的存储方式 (如图).

这种存储方式与 Matlab 自带函数 spdiags 输出对角线的方式相似, 只存储对角线元素, 存储密度更大, 因此不会超过内存限度. 设这个新矩阵为 B, 其中的元素与原矩阵



中元素的对应关系是:

$$A(i,j) = B(j, N+j-i),\ |i-j| \le N-1.$$

实际计算中, 将原有算法中引用矩阵 A 的地方通过对应关系改成引用 B 中元素即可.

5 数值结果及相应分析

对每种情况的每种方法, 取 5 次计算耗时取均值记入结果 (除 $N = 256$ 时一般的 LDL^T 方法耗时过长, 仅进行 2 次计算; $N = 512$ 时一般的 LDL^T 方法预计时间超过 20 小时, 未进行计算). 通过 Matlab 软件计算得到了如下结果 (单位: 秒):

	一般的 LDL^T 方法	带状 Guass 方法	LDL^T 带状方法
$N = 32$	0.4463	0.0182	0.0175
$N = 64$	38.4076	0.1281	0.0981
$N = 128$	2442.8544	1.7314	1.3270
$N = 256$	14346.1470	129.3184	36.0271
$N = 512$	\	970.2106	513.0896

可以看出,

(1) 在 $N = 64, 128$ 时, 与小一倍的规模情况相比, 一般的 LDL^T 方法计算耗时的增长约为 64 倍, 与预估的相差不大; $N = 256$ 时采用了稀疏矩阵的方法存储, 耗时的增长仅约为 6 倍, 个人猜测可能与稀疏矩阵存储使得内存占用大幅减少, 从而使矩阵元素索引时间的增长比值降低. 但尽管如此,$n = 512$ 时的耗时也至少需要约 80000 秒, 与其他两种方法相比进行了过多的无意义计算, 相比来说已不具有实用性.

(2) 除了 $N = 256$ 外, 带状 Guass 方法和 LDL^T 带状方法的计算耗时增长均约为 16 倍, 与预估的相差不大; $N = 256$ 时采用了新的存储方式 (4.4 节), 一定程度上破坏了原矩阵 A 的结构 (例如无法直接索引 A 的行向量), 因此部分矩阵运算只能通过向量运算实现、部分向量运算只能通过标量运算实现, 这导致了计算时间的大量增加; 但由于 LDL^T 带状方法在特殊存储方式中的具体实现中也进行了优化 (原本的矩阵 -向量运

算实际上只需要计算上三角 (如 2.3 节图所示), 变为标量运算后对这一点进行了优化, 减少了一半运算量), 因此 LDL^T 方法耗时增长低于 $Guass$ 带状方法.

(3) 带状 Guass 方法和 LDL^T 带状方法的耗时比例大约为 $2:1$, 与预估的相差不大.

(4) $N = 256$ 时, 一般的 LDL^T 方法耗时已经达到了带状 Guass 方法的 106 倍,LDL^T 带状方法的 384 倍, $N = 512$ 时比值将继续增大. 由此可见, 对稀疏带状矩阵对应的线性方程求解, 应尽可能利用带状 Guass 方法或 LDL^T 带状方法之类的能将运算集中在带状区域上的算法, 以避免过多的无用计算, 获得更高的运行效率.