

### 6.3 应用: Poisson 方程求解

Poisson 方程是一类典型的偏微分方程, 经过差分离散后, 转化为一个线性方程组. 我们以这个线性方程组为例, 讨论 Jacobi, G-S 和 SOR 迭代法的相关性质.

#### 6.3.1 一维 Poisson 方程

首先介绍一维 Poisson 方程的离散. 考虑如下带 Dirichlet 边界条件的一维 Poisson 方程

$$\begin{cases} -\frac{d^2u(x)}{dx^2} = f(x), & 0 < x < 1, \\ u(0) = a, u(1) = b, \end{cases} \quad (6.16)$$

其中  $f(x)$  是给定的函数,  $u(x)$  是需要计算的未知函数.

#### 差分离散

取步长  $h = \frac{1}{n+1}$ , 网格节点为  $x_i = ih, i = 0, 1, 2, \dots, n+1$ . 我们采用二阶中心差分来近似二阶导数, 即

$$-\frac{d^2u(x)}{dx^2} \Big|_{x_i} = \frac{2u(x_i) - u(x_{i-1}) - u(x_{i+1}))}{h^2} + O\left(h^2 \cdot \left\| \frac{d^4u}{dx^4} \right\|_{\infty}\right), \quad i = 1, 2, \dots, n.$$

将其代入 (6.16), 舍去高阶项后得到 Poisson 方程在  $x_i$  点的近似离散方程

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i,$$

其中  $f_i = f(x_i)$ ,  $u_i$  为  $u(x_i)$  的近似. 令  $i = 1, 2, \dots, n$ , 则可得  $n$  个线性方程. 写成矩阵形式为

$$T_n u = f, \quad (6.17)$$

其中

$$T_n = \begin{bmatrix} 2 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & -1 & \\ & & -1 & 2 & \end{bmatrix} \triangleq \text{tridiag}(-1, 2, -1), \quad u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix}, \quad f = \begin{bmatrix} f_1 + u_0 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n + u_{n+1} \end{bmatrix}. \quad (6.18)$$

由边界条件可知  $u_0 = a, u_{n+1} = b$ .

#### 系数矩阵 $T_n$ 的性质

易知  $T_n$  是三对角对称矩阵, 因此其特征值都是实数. 事实上, 我们有下面的结论.

**引理 6.24**  $T_n$  的特征值和对应的特征向量分别为

$$\lambda_k = 2 - 2 \cos \frac{k\pi}{n+1},$$

$$z_k = \sqrt{\frac{2}{n+1}} \left[ \sin \frac{k\pi}{n+1}, \sin \frac{2k\pi}{n+1}, \dots, \sin \frac{nk\pi}{n+1} \right]^T, \quad k = 1, 2, \dots, n,$$

即  $T_n = Z \Lambda Z^T$ , 其中  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  是对角矩阵,  $Z = [z_1, z_2, \dots, z_n]$  是正交矩阵.



(留作课外自习, 直接代入验证即可)

由此可知,  $T_n$  是对称正定的. 更一般地, 我们有下面的结论.

**推论 6.25** 设  $T = \text{tridiag}(a, b, c) \in \mathbb{R}^{n \times n}$ ,  $ac > 0$ , 则  $T$  的特征值为

$$\lambda_k = b - 2\sqrt{ac} \cos \frac{k\pi}{n+1}, \quad k = 1, 2, \dots, n,$$

对应的特征向量为  $z_k$ , 其第  $j$  个分量为

$$z_k(j) = \left(\frac{a}{c}\right)^{\frac{j}{2}} \sin \frac{jk\pi}{n+1}.$$

特别地, 若  $a = c = 1$ , 则对应的单位特征向量为

$$z_k = \sqrt{\frac{2}{n+1}} \left[ \sin \frac{k\pi}{n+1}, \sin \frac{2k\pi}{n+1}, \dots, \sin \frac{nk\pi}{n+1} \right]^T.$$

(留作练习)

由引理 6.24 可知,  $T_n$  的最大特征值为

$$2 \left( 1 - \cos \frac{n\pi}{n+1} \right) = 4 \sin^2 \frac{n\pi}{2(n+1)} \approx 4,$$

最小特征值为

$$2 \left( 1 - \cos \frac{\pi}{n+1} \right) = 4 \sin^2 \frac{\pi}{2(n+1)} \approx \left( \frac{\pi}{n+1} \right)^2.$$

因此, 当  $n$  很大时,  $T_n$  的谱条件数约为

$$\kappa_2(T_n) \approx \frac{4(n+1)^2}{\pi^2}.$$

### 6.3.2 二维 Poisson 方程

考虑二维 Poisson 方程的离散. 同样是带 Dirichlet 边界条件, 即

$$\begin{cases} -\Delta u(x, y) = -\frac{\partial^2 u(x, y)}{\partial x^2} - \frac{\partial^2 u(x, y)}{\partial y^2} = f(x, y), & (x, y) \in \Omega, \\ u(x, y) = u_0(x, y), & (x, y) \in \partial\Omega, \end{cases} \quad (6.19)$$

其中  $\Omega = [0, 1] \times [0, 1]$  为求解区域,  $\partial\Omega$  表示  $\Omega$  的边界.

#### 五点差分离散

为了简单起见, 我们在  $x$ -方向和  $y$ -方向取相同的步长  $h = \frac{1}{n+1}$ , 网格节点为  $x_i = ih, y_j = jh, i, j = 0, 1, 2, \dots, n+1$ . 在  $x$ -方向和  $y$ -方向同时采用二阶中心差分近似, 可得

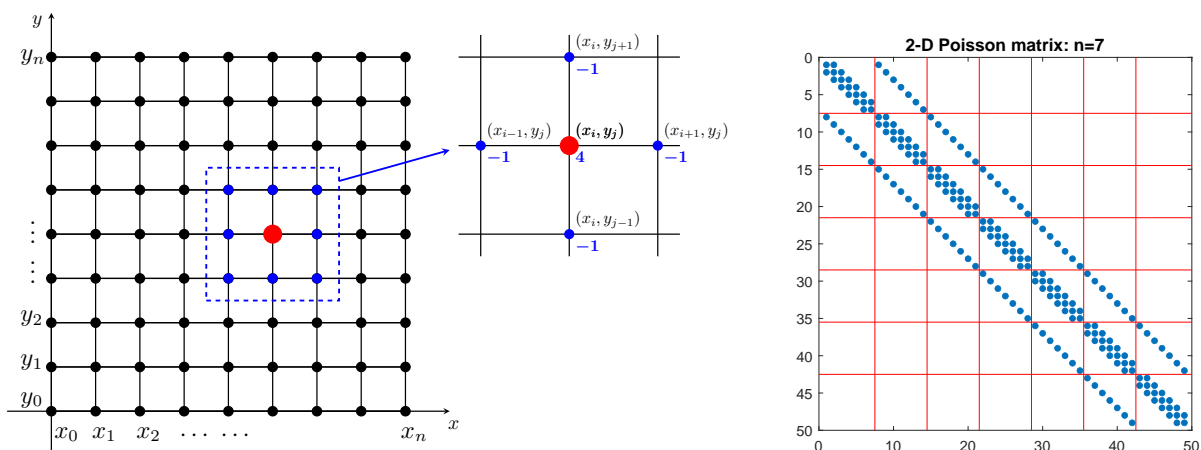
$$\begin{aligned} -\frac{\partial^2 u(x, y)}{\partial x^2} \Big|_{(x_i, y_j)} &\approx \frac{2u(x_i, y_j) - u(x_{i-1}, y_j) - u(x_{i+1}, y_j)}{h^2}, \\ -\frac{\partial^2 u(x, y)}{\partial y^2} \Big|_{(x_i, y_j)} &\approx \frac{2u(x_i, y_j) - u(x_i, y_{j-1}) - u(x_i, y_{j+1})}{h^2}. \end{aligned}$$

代入 (6.19) 后, 就可以得到二维 Poisson 方程在  $(x_i, y_j)$  点的近似离散方程

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{i,j},$$

其中  $f_{i,j} = f(x_i, y_j)$ ,  $u_{i,j}$  为  $u(x_i, y_j)$  的近似. 这个离散格式可以用下面的图 (左图) 来描述.





按自然顺序排列以上方程 (即先沿  $x$ -方向从左到右, 然后沿  $y$  方向从下往上), 可得矩阵形式

$$T_N u = h^2 f, \quad (6.20)$$

其中

$$T_N \triangleq I \otimes T_n + T_n \otimes I, \quad u = [u_{1,1}, \dots, u_{n,1}, u_{1,2}, \dots, u_{n,2}, \dots, u_{1,n}, \dots, u_{n,n}]^T.$$

这里  $\otimes$  表示 Kronecker 乘积,  $T_n$  为一维 Poisson 方程离散后的系数矩阵, 即 (6.18). 上图 (右图) 为  $N = 7^2$  时的矩阵示例图.

🔴 需要注意的是, 系数矩阵  $T_N$  与网格点的排序有关, 不同的排序方式会得到不同的  $T_N$ .

🔴 类似地, 如果对三维 Poisson 方程进行中心差分离散, 则对应的系数矩阵为

$$T_n \otimes I \otimes I + I \otimes T_n \otimes I + I \otimes I \otimes T_n.$$

### 系数矩阵 $T_N$ 的性质

**定理 6.26** 设  $T_n = Z\Lambda Z^T$ , 其中  $Z = [z_1, z_2, \dots, z_n]$  为正交阵,  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  为对角阵, 则  $T_N$  的特征值分解为

$$T_N = (Z \otimes Z)(I \otimes \Lambda + \Lambda \otimes I)(Z \otimes Z)^T,$$

即  $T_N$  的特征值为  $\lambda_i + \lambda_j$ , 对应的特征向量为  $z_i \otimes z_j$ ,  $i, j = 1, 2, \dots, n$ .

由于  $T_N$  对称正定, 所以其条件数为

$$\kappa(T_N) = \frac{\lambda_{\max}(T_N)}{\lambda_{\min}(T_N)} = \frac{1 - \cos \frac{n\pi}{n+1}}{1 - \cos \frac{\pi}{n+1}} = \frac{\sin^2 \frac{n\pi}{2(n+1)}}{\sin^2 \frac{\pi}{2(n+1)}} \approx \frac{4(n+1)^2}{\pi^2}.$$

故当  $n$  越来越大时,  $\kappa(T_N)$  也越来越大, 即  $T_N$  越来越病态.

### 求解二维离散 Poisson 方程的 Jacobi 迭代法



**算法 6.5.** 求解二维离散 Poisson 方程的 Jacobi 迭代法

```

1: Given an initial guess  $u^{(0)} \in \mathbb{R}^N$ 
2: while not converge do
3:   for  $i = 1$  to  $n$  do
4:     for  $j = 1$  to  $n$  do
5:        $u_{i,j}^{(k+1)} = \frac{1}{4} \left( h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)} \right)$ 
6:     end for
7:   end for
8: end while

```

**求解二维离散 Poisson 方程的 G-S 迭代法**

我们可以直接用 Gauss-Seidel 迭代法对基于自然顺序的离散 Poisson 方程 (6.20) 进行数值求解, 即

```

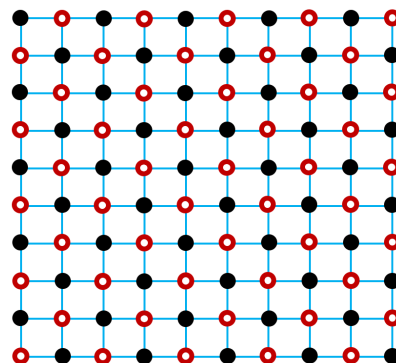
1: Given an initial guess  $u^{(0)} \in \mathbb{R}^N$ 
2: while not converge do
3:   for  $i = 1$  to  $n$  do
4:     for  $j = 1$  to  $n$  do
5:        $u_{i,j}^{(k+1)} = \frac{1}{4} \left( h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} \right)$ 
6:     end for
7:   end for
8: end while

```

但此时迭代解的更新必须按自然顺序进行 (即更新  $u_{i,j}^{(k+1)}$  时必须先更新  $u_{i-1,j}^{(k+1)}$  和  $u_{i,j-1}^{(k+1)}$  的值), 因此不适合并行计算.

下面我们介绍一种适合并行计算的排序方法: **红黑排序**, 即将二维网格点依次做红黑记号, 如右图所示.

在计算过程中, 对  $u_{i,j}^{(k+1)}$  的值进行更新时, 我们可以先更新红色节点上的值, 此时所使用的只是黑色节点的数据, 然后再更新黑色节点上的值, 这时使用的是红色节点的数据. 由于在更新红点时, 各个红点之间是相互独立的, 因此可以并行计算. 同样, 在更新黑点时, 各个黑点之间也是相互独立的, 因此也可以并行计算.

**算法 6.6.** 求解二维离散 Poisson 方程的红黑排序 G-S 迭代法

```

1: Given an initial guess  $u^{(0)} \in \mathbb{R}^N$ 
2: while not converge do

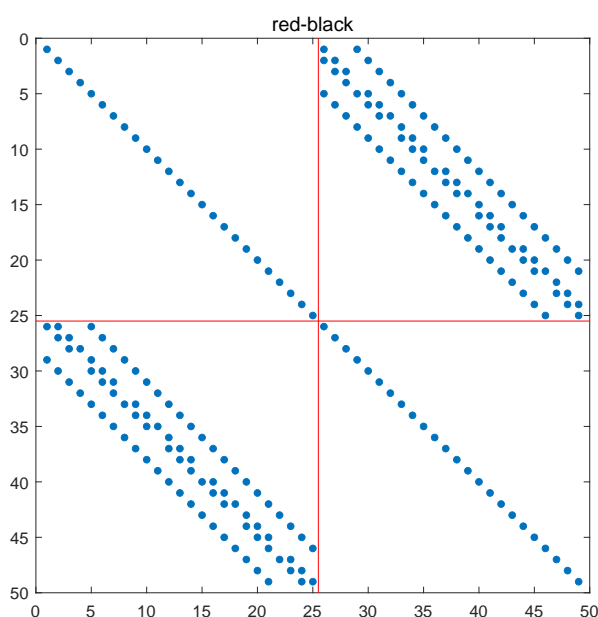
```

```

3:   for (i, j) 为红色节点 do
4:      $u_{i,j}^{(k+1)} = \frac{1}{4} (h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)})$    % 使用旧数据
5:   end for
6:   for (i, j) 为黑色节点 do
7:      $u_{i,j}^{(k+1)} = \frac{1}{4} (h^2 f_{i,j} + u_{i+1,j}^{(k+1)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k+1)} + u_{i,j-1}^{(k+1)})$    % 使用新数据
8:   end for
9: end while

```

我们记基于红黑排序所得到的系数矩阵为  $T_N^{\text{RB}}$ , 它可以写成  $2 \times 2$  分块形式, 且两个对角块都是对角矩阵, 如下图所示 ( $N = 7^2$ )



事实上,  $T_N^{\text{RB}}$  可以通过对  $T_N$  进行行置换和列置换得到, 即存在置换矩阵  $P$  使得

$$T_N^{\text{RB}} = PT_N P^T.$$

### 求解二维离散 Poisson 方程的 SOR 迭代法

**算法 6.7.** 求解二维离散 Poisson 方程的 SOR 迭代法


```

1: Given an initial guess  $u^{(0)} \in \mathbb{R}^N$  and a parameter  $\omega$ 
2: while not converge do
3:   for  $i = 1$  to  $n$  do
4:     for  $j = 1$  to  $n$  do
5:        $u_{i,j}^{(k+1)} = (1 - \omega)u_{i,j}^{(k)} + \frac{\omega}{4} (h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)})$ 
6:     end for
7:   end for

```



8: **end while**

 上面的 SOR 算法是基于自然排序, 基于红黑排序的 SOR 算法请读者自己完成.

**例 6.2** 已知二维 Poisson 方程

$$\begin{cases} -\Delta u(x, y) = -1, & (x, y) \in \Omega \\ u(x, y) = \frac{x^2 + y^2}{4}, & (x, y) \in \partial\Omega \end{cases}$$

其中  $\Omega = (0, 1) \times (0, 1)$ . 该方程的解析解是  $u(x, y) = \frac{x^2 + y^2}{4}$ . 用五点差分格式离散后得到一个线性方程组.

(1) 分别用 Jacobi, G-S 和 SOR 迭代法计算该方程组的解.

(2) 分别用 SOR 和 SSOR 迭代法求解该方程组, 观察参数  $\omega$  对算法收敛的影响.

**解.** (1) MATLAB 程序参见 [Poisson\\_Jacobi\\_GS\\_SOR.m](#). 定义近似解的相对误差:

$$\text{relerr}_k \triangleq \frac{\|u^{(k)} - u_*\|_2}{u_*},$$

其中  $u_*$  表示精确解. 下图画出了  $n = 16$  (即  $N = 256$ ) 时三种方法的近似解相对误差的下降过程 (前 100 个迭代步).

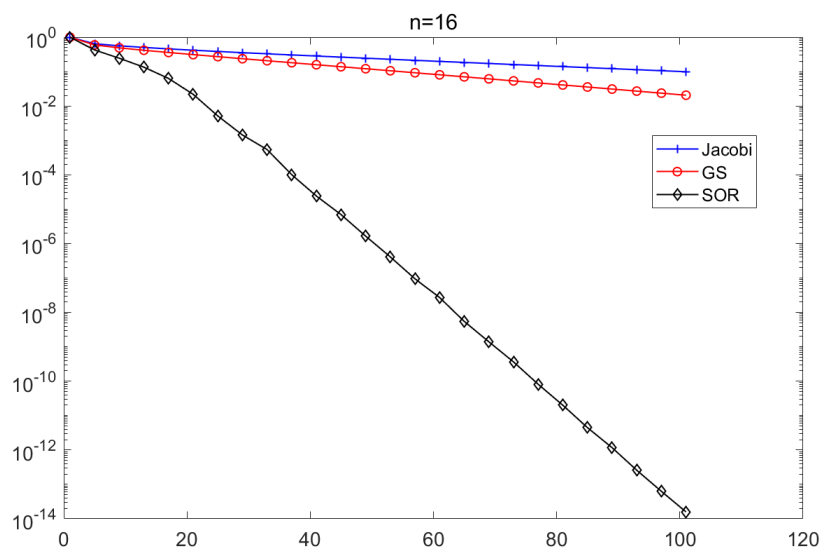


图 6.1. Jacobi, G-S 和 SOR 的近似解相对误差的下降曲线.

(2) MATLAB 程序参见 [Poisson\\_SOR\\_omega.m](#) 和 [Poisson\\_SSOR\\_omega.m](#). 下图中画出了  $n = 8$  (即  $N = 64$ ) 时, SOR 和 SSOR 收敛结果与参数  $\omega$  取值之间的关系.

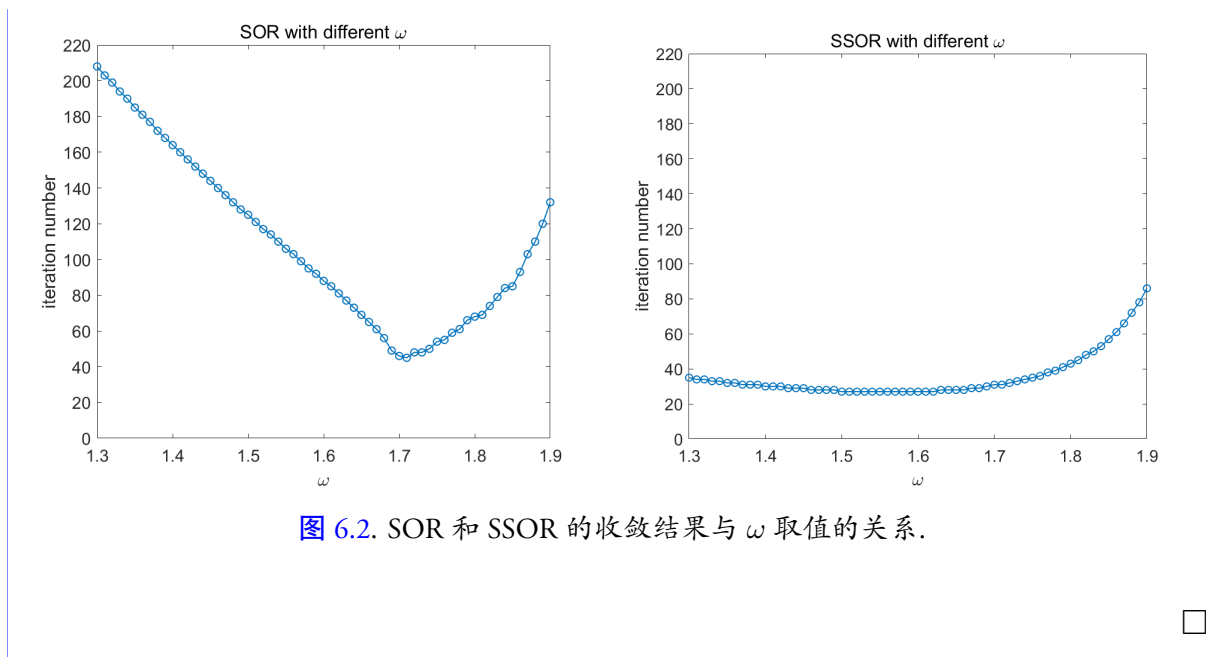


图 6.2. SOR 和 SSOR 的收敛结果与  $\omega$  取值的关系.

### 6.3.3 收敛性分析

对于基本迭代法, 其收敛的充要条件是迭代矩阵的谱半径小于 1. 当谱半径不可求时, 我们可以根据迭代矩阵的范数来判断, 即如果迭代矩阵的某个范数小于 1, 则迭代法收敛.

本小节考虑求解二维离散 Poisson 方程的 Jacobi, G-S 和 SOR 的收敛性. 考虑这些方法的收敛性, 只需研究相应的迭代矩阵的谱半径即可. 对于二维离散 Poisson 方程, 系数矩阵为

$$A = T = I \otimes T_n + T_n \otimes I.$$

故 Jacobi 迭代法的迭代矩阵为

$$G_J = D^{-1}(L + U) = (4I)^{-1}(4I - T) = I - T/4. \quad (6.21)$$

由于  $T$  的特征值为

$$\lambda_i + \lambda_j = 2 \left( 1 - \cos \frac{\pi i}{n+1} \right) + 2 \left( 1 - \cos \frac{\pi j}{n+1} \right) = 4 - 2 \left( \cos \frac{\pi i}{n+1} + \cos \frac{\pi j}{n+1} \right),$$

所以  $G_J$  的特征值为

$$1 - (\lambda_i + \lambda_j)/4 = \frac{1}{2} \left( \cos \frac{\pi i}{n+1} + \cos \frac{\pi j}{n+1} \right).$$

故

$$\rho(G_J) = \frac{1}{2} \max_{i,j} \left\{ \left| \cos \frac{\pi i}{n+1} + \cos \frac{\pi j}{n+1} \right| \right\} = \cos \frac{\pi}{n+1} < 1,$$

即 Jacobi 迭代法是收敛的.

注意当  $n$  越来越大时,  $\kappa(T) \rightarrow \infty$ , 即  $T$  越来越病态, 此时  $\rho(G_J) \rightarrow 1$ , 即 Jacobi 迭代法收敛越来越慢.

 通常, 问题越病态就越难求解, 舍入误差对解的影响也越大.

关于 G-S 和 SOR, 我们有下面的结论.



**定理 6.27** 设  $G_{GS}$  和  $G_{SOR}$  分别表示求解基于红黑排序的二维 Poisson 方程的 G-S 和 SOR 的迭代矩阵, 则有

$$\rho(G_{GS}) = \rho(G_J)^2 = \cos^2 \frac{\pi}{n+1} < 1$$

$$\rho(G_{SOR}) = \frac{\cos^2 \frac{\pi}{n+1}}{\left(1 + \sin \frac{\pi}{n+1}\right)^2} < 1, \quad \omega = \frac{2}{1 + \sin \frac{\pi}{n+1}}.$$

**证明.** 由于此时的系数矩阵  $T_N$  具有性质 A, 因此直接利用定理 6.23 即可. □

在上述结论中, SOR 中的  $\omega$  是最优参数, 即此时的  $\rho(G_{SOR})$  最小. 由 Taylor 公式可知, 当  $n$  很大时, 有


$$\rho(G_J) = \cos \frac{\pi}{n+1} \approx 1 - \frac{\pi^2}{2(n+1)^2} = 1 - O\left(\frac{1}{n^2}\right),$$

$$\rho(G_{SOR}) = \frac{\cos^2 \frac{\pi}{n+1}}{\left(1 + \sin \frac{\pi}{n+1}\right)^2} \approx 1 - \frac{2\pi}{n+1} = 1 - O\left(\frac{1}{n}\right).$$


由于当  $n$  很大时有

$$\left(1 - \frac{1}{n}\right)^k \approx 1 - \frac{k}{n} = 1 - \frac{kn}{n^2} \approx \left(1 - \frac{1}{n^2}\right)^{kn},$$

即 SOR 迭代  $k$  步后误差的减小量与 Jacobi 迭代  $kn$  步后误差减小量差不多. 因此, 对于二维离散 Poisson 方程, 当 SOR 取最优参数时, 收敛速度大约是 Jacobi 的  $n$  倍.

 需要指出的是, 对于一般线性方程组, 上述结论不一定成立.

由于  $\rho(G_{GS}) = \rho(G_J)^2$ , 因此, 对于二维离散 Poisson 方程, G-S 的收敛速度大约是 Jacobi 的 2 倍.

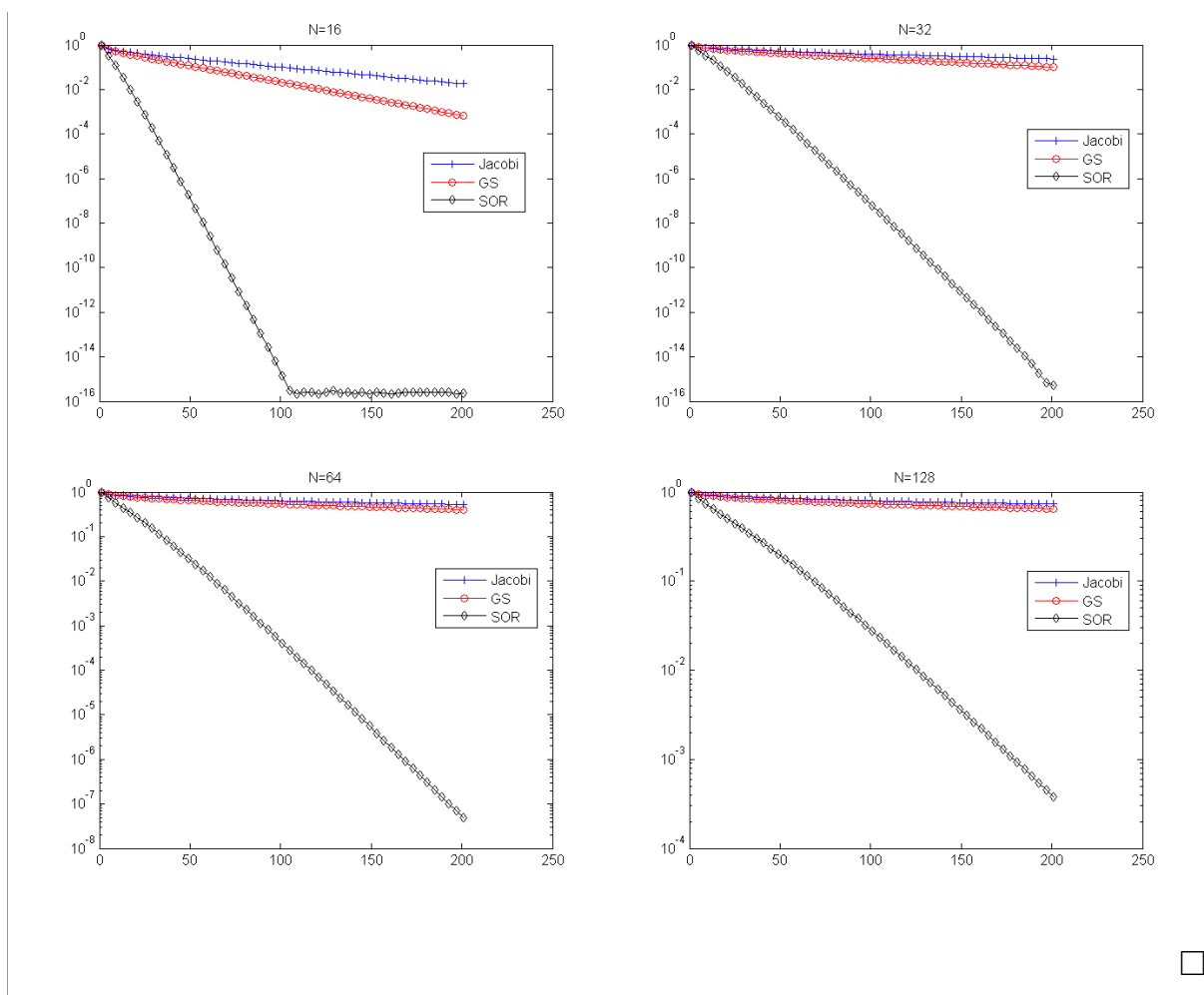
 事实上, 当  $n$  很大时, 这三个方法的收敛速度都很慢.

**例 6.3** 同例 ??, 分别对不同的  $n$ , 观察 Jacobi, Gauss-Seidel 和 SOR 迭代法的收敛情况.

**解.** 参见 MATLAB 程序 `Poisson_Jacobi_GS_SOR_convergence.m`. 下图中画出了  $N = 16, 32, 64, 128$  时, 这三种方法的相对误差下降情况.







事实上, 由于二维离散 Poisson 方程的系数矩阵  $T_N$  是不可约对角占优的, 因此根据定理 6.8, Jacobi 和 G-S 都收敛. 另外,  $T_N$  是对称正定的, 故当  $0 < \omega < 2$  时, SOR 迭代法收敛.

### 6.3.4 快速求解方法

如果已经知道矩阵  $A$  的特征值分解  $A = X\Lambda X^{-1}$ , 则  $Ax = b$  的解可表示为

$$x = A^{-1}b = X\Lambda^{-1}X^{-1}b.$$

如果  $A$  是正规矩阵, 即  $X$  是酉矩阵, 则

$$x = A^{-1}b = X\Lambda^{-1}X^*b.$$

一般来说, 我们不会采用这种特征值分解的方法来解线性方程组, 因为计算特征值分解通常比解线性方程组更困难. 但在某些特殊情况下, 我们可以由此得到快速方法.

考虑二维离散 Poisson 方程

$$Tu = h^2 f, \quad (6.22)$$



其中

$$T = I \otimes T_n + T_n \otimes I, \quad T_n = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{n \times n}.$$

由定理 6.26 可知

$$T = (Z \otimes Z)(I \otimes \Lambda + \Lambda \otimes I)(Z \otimes Z)^T,$$

其中  $Z = [z_1, z_2, \dots, z_n]$  是正交矩阵. 这里

$$z_k = \sqrt{\frac{2}{n+1}} \left[ \sin \frac{k\pi}{n+1}, \sin \frac{2k\pi}{n+1}, \dots, \sin \frac{nk\pi}{n+1} \right]^T, \quad k = 1, 2, \dots, n.$$

所以, 方程 (6.22) 的解为

$$u = T^{-1}h^2f = [(Z \otimes Z)(I \otimes \Lambda + \Lambda \otimes I)^{-1}(Z \otimes Z)^T]h^2f.$$

因此, 主要的运算是  $Z \otimes Z$  与向量的乘积, 以及  $(Z \otimes Z)^T$  与向量的乘积. 而这些乘积可以通过快速 Sine 变换来实现.

### 快速 Fourier 变换

**快速 Fourier 变换 (FFT)** 是用来计算离散 Fourier 变换 (DFT) 矩阵与向量乘积的一种快速方法.


设  $x = [x_0, x_1, \dots, x_{n-1}]^T \in \mathbb{C}^n$ , 其 DFT 定义为  $y = \text{DFT}(x) = [y_0, y_1, \dots, y_{n-1}]^T \in \mathbb{C}^n$ , 其中

$$y_k = \sum_{j=0}^{n-1} \omega_n^{kj} x_j, \quad k = 0, 1, \dots, n-1.$$

这里

$$\omega_n = e^{-\frac{2\pi i}{n}} = \cos(2\pi/n) - \mathbf{i} \sin(2\pi/n)$$

是 1 的一个  $n$  次本原根 (primitive  $n$ -th root of unity),  $\mathbf{i}$  是虚部单位.

 这里说  $\omega_n$  是 primitive  $n$ -th root of unity 是指  $\omega_n^n = 1$  且

$$\omega_n^k \neq 1, \quad k = 1, 2, \dots, n-1.$$

构造矩阵  $F_n = [f_{kj}] \in \mathbb{C}^{n \times n}$ , 其中  $f_{kj} = \omega_n^{kj} = e^{-\frac{2kj\pi i}{n}} = \cos(2kj\pi/n) - \mathbf{i} \sin(2kj\pi/n)$ , 即

$$F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix},$$

则有

$$y = \text{DFT}(x) = F_n x.$$

我们称矩阵  $F_n$  为 DFT 矩阵. 易知 DFT 矩阵具有以下性质:

- (1)  $F_n$  是对称矩阵 (但不是 Hermitian);
- (2)  $F_n^* F_n = nI$ , 所以  $\frac{1}{\sqrt{n}} F_n$  是酉矩阵.



相应的离散 Fourier 反变换定义为  $x = \text{IDFT}(y)$ , 其中

$$x_j = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{-jk} y_k, \quad j = 0, 1, \dots, n-1.$$


写成矩阵形式为

$$x = \frac{1}{n} F^* y.$$

DFT 和 IDFT 满足下面的性质:

$$\text{IDFT}(\text{DFT}(x)) = x,$$

$$\text{DFT}(\text{IDFT}(y)) = y.$$

 在 MATLAB 中, 计算 DFT 和 IDFT 的函数分别为 `fft` 和 `ifft`, 即: `y=fft(x)`, `x=ifft(y)`.  
(测试代码见 `FFT_test.m`)

## 离散 Sine 变换

**离散 Sine 变换 (DST)** 有多种定义, 我们这里只介绍与求解 Poission 方程有关的一种定义, 其它定义可参见 [17].

设  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$ , 其离散 Sine 变换定义为  $y = \text{DST}(x) = [y_1, y_2, \dots, y_n]^T \in \mathbb{R}^n$ , 其中

$$y_k = \sum_{j=1}^n x_j \sin\left(\frac{kj\pi}{n+1}\right), \quad k = 1, 2, \dots, n.$$


对应的离散 Sine 反变换记为 IDST, 即  $x = \text{IDST}(y)$ , 其中

$$x_j = \frac{2}{n+1} \sum_{k=1}^n y_k \sin\left(\frac{jk\pi}{n+1}\right), \quad j = 1, 2, \dots, n.$$

DST 和 IDST 满足下面的性质:

$$\text{IDST}(\text{DST}(x)) = x,$$

$$\text{DST}(\text{IDST}(y)) = y.$$

 在 MATLAB 中, 计算 DST 和 IDST 的函数分别为 `dst` 和 `idst`, 即: `y=dst(x)`, `x=idst(y)`.  
(测试代码见 `DST_test.m`)

## Poission 方程与 DST

我们首先考虑矩阵  $Z$  与一个任意给定向量  $b$  的乘积. 设  $y = Zb$ , 则

$$y_k = \sum_{j=1}^n Z(k, j) b_j = \sqrt{\frac{2}{n+1}} \sum_{j=1}^n b_j \sin\left(\frac{kj\pi}{n+1}\right) = \sqrt{\frac{2}{n+1}} [\text{DST}(b)]_k,$$



其中  $[\text{DST}(b)]_k$  表示第  $k$  个分量. 因此, 乘积  $y = Zb$  可以通过 DST 来实现. 类似地, 乘积  $y = Z^T b = Z^{-1}b$  可以通过离散 Sine 反变换 IDST 实现, 即

$$y = Z^T b = Z^{-1}b = \left( \sqrt{\frac{2}{n+1}} \right)^{-1} \text{IDST}(b).$$

所以对于一维离散 Poisson 方程, 其解为

$$u = T_n^{-1}(h^2 f) = (Z\Lambda^{-1}Z^T)(h^2 f) = h^2 Z\Lambda^{-1}Z^T f = h^2 \text{DST}(\Lambda^{-1} \text{IDST}(b)).$$

可以通过公式  $T_n = Z\Lambda Z^T$  来计算一维离散 Poisson 矩阵的特征值: 记  $[\alpha_1, \alpha_2, \dots, \alpha_n]^T$  和  $[\beta_1, \beta_2, \dots, \beta_n]^T$  分别为  $Z^T T_n$  和  $Z^T$  的第一列, 则  $T_n$  的特征值为

$$\lambda_i = \frac{\alpha_i}{\beta_i}.$$

对应的 MATLAB 代码为: `Lam=idst([2,-1,zeros(1,n-2)]') ./ idst(eye(n,1))`

而对于二维离散 Poisson 方程, 我们需要计算  $(Z \otimes Z)b$  和  $(Z^T \otimes Z^T)b$ . 它们对应的是二维离散 Sine 变换和二维离散 Sine 反变换.

设  $b = [b_1^T, b_2^T, \dots, b_n^T]^T \in \mathbb{R}^{n^2}$ , 其中  $b_k \in \mathbb{R}^{n \times n}$ . 令  $B = [b_1, b_2, \dots, b_n] \in \mathbb{R}^{n \times n}$ , 则由 Kronecker 乘积的性质可知

$$(Z \otimes Z)b = (Z \otimes Z)\text{vec}(B) = \text{vec}(ZBZ^T) = \text{vec}\left((Z(ZB)^T)^T\right).$$

因此, 我们仍然可以使用 DST 来计算  $(Z \otimes Z)b$ . 类似地, 我们可以使用 IDST 来计算  $(Z^T \otimes Z^T)b$ .

#### 算法 6.8. 二维离散 Poisson 方程的快速方法

- 1: 计算  $b = h^2 f$
- 2:  $B = \text{reshape}(b, n, n)$
- 3:  $B_1 = (Z^T B)^T = (\text{IDST}(B))^T$
- 4:  $B_2 = (Z^T B_1)^T = (\text{IDST}(B_1))^T$
- 5:  $b_1 = (I \otimes \Lambda + \Lambda \otimes I)^{-1} \text{vec}(B_2)$
- 6:  $B_3 = \text{reshape}(b_1, n, n)$
- 7:  $B_4 = (Z B_3)^T = (\text{DST}(B_3))^T$
- 8:  $B_5 = (Z B_4)^T = (\text{DST}(B_4))^T$
- 9:  $u = \text{reshape}(B_5, n^2, 1)$

MATLAB 程序见 `Poisson_DST.m`

### 6.3.5 求解方法小结

由于 Poisson 方程的特殊结构和性质, 除了常规求解方法以外, 人们还设计出了一些特殊的快速算法. 下表列出了求解二维 Poisson 方程的不同方法的比较 [30], 这里假定网格剖分为  $n \times n$ , 并记  $N = n^2$ .

	方法	串行时间	存储空间
直接法	稠密 Cholesky 分解	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
	显式求逆	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$
	带状 Cholesky 分解	$\mathcal{O}(N^2)$	$\mathcal{O}(N^{3/2})$
	稀疏 Cholesky 分解	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N \log N)$
基本迭代法	Jacobi	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$
	Gauss-Seidel	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$
	SOR	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N)$
	带 Chebyshev 加速的 SSOR	$\mathcal{O}(N^{5/4})$	$\mathcal{O}(N)$
Krylov 子空间迭代	CG (共轭梯度法)	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N)$
	CG (带修正 IC 预处理)	$\mathcal{O}(N^{5/4})$	$\mathcal{O}(N)$
快速方法	DST (快速 Sine 变换)	$\mathcal{O}(N \log N)$	$\mathcal{O}(N)$
	块循环约化	$\mathcal{O}(N \log N)$	$\mathcal{O}(N)$
	Multigrid	$\mathcal{O}(N)$	$\mathcal{O}(N)$

