

# 第六讲 线性方程组的定常迭代解法

- 1 常用定常迭代方法
- 2 应用: Poisson 方程求解
- 3 收敛性分析
- 4 加速方法
- 5 交替方向与 HSS 算法

# 线性方程组的数值求解

- **直接法** PLU 分解,  $LDL^T$  分解, Cholesky 分解等
- **迭代法**
  - 经典 (定常, 不动点) 迭代法: Jacobi, Gauss-Seidel, SOR, SSOR, ...
  - 现代 (Krylov 子空间) 迭代法: CG, MINRES, GMRES, BiCGStab, ...
- **快速算法** (基于特殊结构和性质)
  - 基于各类快速变换, 如 FFT, DCT, DST 等
  - 代数多重网格法 (Algebraic multigrid)
  - 快速多极子算法 (Fast multipole)
  - Hierarchical Matrices



### 注记

有些方法可能只是对某类方程有效, 比如快速算法.

在实际应用中, 这些方法经常结合使用, 如混合算法, 预处理算法等.

本讲主要介绍经典 (定常, 不动点) 迭代方法



更多迭代方法可参见: **Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods**, SIAM, 1994.



# 1 | 常用定常迭代方法

1.1 矩阵分裂迭代方法

1.2 Jacobi 迭代

1.3 Gauss-Seidel 迭代

1.4 SOR 迭代

1.5 SSOR 迭代方法

1.6 AOR 迭代

1.7 Richardson 算法

1.8 分块迭代方法



## 为什么迭代法

当直接求解方程组  $Ax = b$  较困难时, 我们可以求解一个近似方程组

$$Mx = b$$

其中  $M$  是  $A$  的某个近似, 且该方程组较容易求解.

设其解为  $x^{(1)}$ . 易知它与真解之间的误差满足

$$A(x_* - x^{(1)}) = b - Ax^{(1)}$$

如果  $x^{(1)}$  已经满足精度要求, 则停止计算, 否则需要修正.



设修正量为  $\Delta x$ . 显然  $\Delta x$  满足方程  $A\Delta x = b - Ax^{(1)}$ . 但由于直接求解该方程比较困难, 因此我们还是求解近似

$$M\Delta x = b - Ax^{(1)}.$$

于是得到修正后的近似解

$$x^{(2)} = x^{(1)} + \Delta x = x^{(1)} + M^{-1}(b - Ax^{(1)})$$

若  $x^{(2)}$  已经满足精度要求, 则停止计算, 否则继续按以上的方式进行修正.



不断重复以上步骤, 于是, 我们就得到一个序列

$$x^{(1)}, x^{(2)}, \dots, x^{(k)}, \dots$$

满足以下递推关系

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \quad k = 1, 2, \dots$$

由于每次迭代的格式是一样的, 因此称为 **定常迭代**.

通常, 构造一个好的定常迭代, 需要考虑以下两点:

- (1) 以  $M$  为系数矩阵的线性方程组必须要比原线性方程组更容易求解;
- (2)  $M$  应该是  $A$  的一个很好的近似, 或者迭代序列  $\{x_k\}$  要收敛.



本小节我们介绍几个常见的基于矩阵分裂的定常迭代方法:

- Jacobi 算法
- Gauss-Seidel 算法
- SOR (Successive Over-Relaxation) 算法
- SSOR (Symmetric SOR) 算法
- AOR (Accelerated over-relaxation) 算法





# 1.1 矩阵分裂迭代方法

## 迭代方法的基本思想

给定一个迭代初始值  $x^{(0)}$ , 通过一定的迭代格式生成一个迭代序列

$$x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(k)}, \dots$$

使得

$$\lim_{k \rightarrow \infty} x^{(k)} = x_* \triangleq A^{-1}b$$



**定义 (矩阵分裂 Matrix splitting)** 设  $A \in \mathbb{R}^{n \times n}$  非奇异, 称

$$A = M - N \quad (6.1)$$

为  $A$  的一个矩阵分裂, 其中  $M$  非奇异.

原方程组等价于  $Mx = Nx + b$ . 于是我们就可以构造迭代格式

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b \triangleq Gx^{(k)} + g, \quad k = 0, 1, \dots, \quad (6.2)$$

其中  $G = M^{-1}N$  称为该迭代格式的**迭代矩阵**.



## 1.2 Jacobi 迭代

将矩阵  $A$  分裂为

$$A = D - L - U,$$

其中  $D$  为  $A$  的对角线部分,  $-L$  和  $-U$  分别为  $A$  的严格下三角和严格上三角部分.

在矩阵分裂  $A = M - N$  中取  $M = D$ ,  $N = L + U$ , 则可得 Jacobi 迭代算法:

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b, \quad k = 0, 1, 2, \dots \quad (6.3)$$


迭代矩阵为

$$G_J = D^{-1}(L + U).$$



写成分量形式即为

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n.$$

 由于 Jacobi 迭代中  $x_i^{(k+1)}$  的更新顺序与  $i$  无关, 即可以按顺序  $i = 1, 2, \dots, n$  计算, 也可以按顺序  $i = n, n-1, \dots, 2, 1$  计算, 或者乱序计算. 因此 Jacobi 迭代非常适合并行计算.



### 算法 1.1 求解线性方程组的 Jacobi 迭代方法

- 1: Choose an initial guess  $x^{(0)}$
- 2: **while** not converge **do**
- 3:     **for**  $i = 1$  to  $n$  **do**
- 4:         
$$x_i^{(k+1)} = \left( b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right) / a_{ii}$$
- 5:     **end for**
- 6: **end while**

我们也可以将 Jacobi 迭代格式写为

$$x^{(k+1)} = x^{(k)} + D^{-1}(b - Ax^{(k)}) = x^{(k)} + D^{-1}r_k, \quad k = 0, 1, \dots,$$

其中  $r_k \triangleq b - Ax^{(k)}$  是  $k$  次迭代后的残量.



## 1.3 Gauss-Seidel 迭代

取  $M = D - L$ ,  $N = U$ , 即可得 Gauss-Seidel (G-S) 迭代算法:

$$x^{(k+1)} = (D - L)^{-1}Ux^{(k)} + (D - L)^{-1}b \quad (6.4)$$

迭代矩阵为

$$G_{GS} = (D - L)^{-1}U$$



将 G-S 迭代改写为

$$Dx^{(k+1)} = Lx^{(k+1)} + Ux^{(k)} + b,$$

即可得分量形式

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right), \quad i = 0, 1, \dots, n.$$



## 算法 1.2 求解线性方程组的 G-S 迭代方法

1: Choose an initial guess  $x^{(0)}$

2: **while** not converge **do**

3:     **for**  $i = 1$  to  $n$  **do**

4:         
$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right)$$

5:     **end for**

6: **end while**

📖 G-S 算法的主要优点是充分利用了已经获得的最新数据

📖 但 G-S 算法中未知量的更新是按自然顺序进行的, 不适合并行计算





## 1.4 SOR 迭代


在 G-S 算法的基础上, 我们可以通过引入一个松弛参数  $\omega$  来加快收敛速度. 这就是 SOR (Successive Overrelaxation) 算法:

$$x^{(k+1)} = (1 - \omega)x^{(k)} + \omega \left( D^{-1}(Lx^{(k+1)} + Ux^{(k)}) + D^{-1}b \right).$$

整理后即为

$$x^{(k+1)} = (D - \omega L)^{-1} \left( (1 - \omega)D + \omega U \right) x^{(k)} + \omega(D - \omega L)^{-1}b,$$

其中  $\omega$  称为**松弛参数**.

 注意, 这里是对向量进行整体松弛, 而不是对分量进行松弛.



- (1) 当  $\omega = 1$  时, SOR 即为 G-S 算法,
- (2) 当  $\omega < 1$  时, 称为低松弛 (under relaxation) 算法,
- (3) 当  $\omega > 1$  时, 称为超松弛 (over relaxation) 算法.

👉 SOR 算法曾经在很长一段时间内是科学计算中求解大规模线性方程组的首选方法.

👉 在大多数情况下, 当  $\omega > 1$  时会取得比较好的收敛效果.



SOR 的迭代矩阵为

$$G_{\text{SOR}} = (D - \omega L)^{-1} ((1 - \omega)D + \omega U),$$

对应的矩阵分裂为

$$M = \frac{1}{\omega}D - L, \quad N = \frac{1 - \omega}{\omega}D + U.$$

SOR 迭代的分量形式为

$$\begin{aligned} x_i^{(k+1)} &= (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right) \\ &= x_i^{(k)} + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^n a_{ij}x_j^{(k)} \right) \end{aligned}$$



### 算法 1.3 求解线性方程组的 SOR 迭代方法

1: Choose an initial guess  $x^{(0)}$  and parameter  $\omega$

2: **while** not converge **do**

3:     **for**  $i = 1$  to  $n$  **do**

4:         
$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right)$$

5:     **end for**

6: **end while**

📖 SOR 算法最大的优点是引入了松弛参数  $\omega$ , 通过选取适当的  $\omega$  可以大大提高算法的收敛速度.

📖 但是 SOR 算法最大的难点就是如何选取最优的参数.



## 1.5 SSOR 迭代方法

将 SOR 算法中的  $L$  和  $U$  相交换, 即可得迭代格式

$$x^{(k+1)} = (D - \omega U)^{-1} ((1 - \omega)D + \omega L) x^{(k)} + \omega(D - \omega U)^{-1} b,$$

将这个迭代格式与 SOR 相结合, 就可以得到下面的两步迭代方法

$$\begin{cases} x^{(k+\frac{1}{2})} = (D - \omega L)^{-1} [(1 - \omega)D + \omega U] x^{(k)} + \omega(D - \omega L)^{-1} b \\ x^{(k+1)} = (D - \omega U)^{-1} [(1 - \omega)D + \omega L] x^{(k+\frac{1}{2})} + \omega(D - \omega U)^{-1} b \end{cases}$$

这就是 **SSOR 迭代** (对称超松弛) 算法, 相当于将  $L$  与  $U$  同等看待, 交替做两次 SOR 迭代.



消去中间迭代向量  $x^{(k+\frac{1}{2})}$ , 可得

$$x^{(k+1)} = G_{\text{SSOR}}x^{(k)} + g,$$

其中迭代矩阵

$$G_{\text{SSOR}} = (D - \omega U)^{-1}[(1 - \omega)D + \omega L](D - \omega L)^{-1}[(1 - \omega)D + \omega U].$$

对应的矩阵分裂为

$$\begin{aligned} M &= \frac{1}{\omega(2 - \omega)} [D - \omega(L + U) + \omega^2 LD^{-1}U] \\ &= \frac{1}{\omega(2 - \omega)} (D - \omega L)D^{-1}(D - \omega U), \\ N &= \frac{1}{\omega(2 - \omega)} [(1 - \omega)D + \omega L]D^{-1}[(1 - \omega)D + \omega U]. \end{aligned}$$



✎ 对于某些特殊问题, SOR 算法不收敛, 但仍然可能构造出收敛的 SSOR 算法.

✎ 一般来说, SOR 算法的渐进收敛速度对参数  $\omega$  比较敏感, 但 SSOR 对参数  $\omega$  不太敏感.

([Poisson\\_SOR\\_omega.m](#), [Poisson\\_SSOR\\_omega.m](#))



## 1.6 AOR 迭代

Hadjidimos 于 1978 年提出了 AOR (Accelerated over-relaxation, 快速松弛) 算法, 迭代矩阵为

$$G_{\text{AOR}} = (D - \gamma L)^{-1} [(1 - \omega)D + (\omega - \gamma)L + \omega U],$$

其中  $\gamma$  和  $\omega$  为松弛参数. 对应的矩阵分解为

$$M = \frac{1}{\omega}(D - \gamma L), \quad N = \frac{1}{\omega}[(1 - \omega)D + (\omega - \gamma)L + \omega U].$$

- (1) 当  $\gamma = \omega$  时, AOR 算法即为 SOR 算法;
- (2) 当  $\gamma = \omega = 1$  时, AOR 算法即为 G-S 算法;
- (3) 当  $\gamma = 0, \omega = 1$  时, AOR 算法即为 Jacobi 算法.

 与 SSOR 类似, 我们也可以定义 SAOR 算法.





## 1.7 Richardson 算法

Richardson 算法是一类形式非常简单的算法, 其迭代格式为

$$x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots$$

对应的矩阵分裂和迭代矩阵分别为

$$M = \frac{1}{\omega}I, \quad N = \frac{1}{\omega}I - A, \quad G_R = I - \omega A.$$

如果在每次迭代时取不同的参数, 即

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots,$$

则称为 nonstationary Richardson 算法.



**定理** 设  $A \in \mathbb{R}^{n \times n}$  是对称正定矩阵,  $\lambda_1$  和  $\lambda_n$  分别是  $A$  的最大和最小特征值, 则 Richardson 算法收敛当且仅当

$$0 < \omega < \lambda_1^{-1}.$$

最优参数为

$$\omega_* = \arg \min_{\omega} \rho(G_R) = \frac{2}{\lambda_1 + \lambda_n},$$

即当  $\omega = \omega_*$  时, 迭代矩阵的谱半径达到最小, 且有

$$\rho(G_R) = \begin{cases} 1 - \omega \lambda_n & \text{if } \omega \leq \omega_* \\ \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} = \frac{\kappa(A) - 1}{\kappa(A) + 1} & \text{if } \omega = \omega_* \\ \omega \lambda_1 - 1 & \text{if } \omega \geq \omega_*. \end{cases}$$

(证明见讲义, 留作自习)



## 1.8 分块迭代方法

前面介绍的迭代方法可以推广到分块情形.

将  $A$  写成如下的分块形式:

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{pp} \end{bmatrix}.$$

$A_{11}$					
	$A_{22}$				
					$A_{pp}$

设  $A = D - L - U$ , 其中  $D, -L, -U$  分别是  $A$  的块对角, 块严格下三角和块严格上三角矩阵, 则可得相应的分块 Jacobi, 分块 Gauss-Seidel 和分块 SOR 算法。



### 分块 Jacobi 迭代

$$A_{ii}\mathbf{x}_i^{(k+1)} = \mathbf{b}_i - \sum_{j=1, j \neq i}^p A_{ij}\mathbf{x}_j^{(k)}, \quad i = 1, 2, \dots, p.$$

### 分块 Gauss-seidel 迭代

$$A_{ii}\mathbf{x}_i^{(k+1)} = \mathbf{b}_i - \sum_{j=1}^{i-1} A_{ij}\mathbf{x}_j^{(k+1)} - \sum_{j=i+1}^p A_{ij}\mathbf{x}_j^{(k)}, \quad i = 1, 2, \dots, p.$$

### 分块 SOR 迭代

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} + \omega A_{ii}^{-1} \left( \mathbf{b}_i - \sum_{j=1}^i A_{ij}\mathbf{x}_j^{(k+1)} - \sum_{j=i+1}^p A_{ij}\mathbf{x}_j^{(k)} \right),$$
$$i = 1, 2, \dots, p.$$



## 2 | 应用: Poisson 方程求解

### 2.1 一维 Poisson 方程

### 2.2 二维 Poisson 方程

在本讲中, 我们以一个典型的线性方程组为例, 逐个介绍各种迭代方法, 并比较它们之间的性能. 这个方程组就是二维 Poisson 方程经过五点差分离散后得到的线性方程组.



## 2.1 一维 Poisson 方程

考虑如下带 Dirichlet 边界条件的一维 Poisson 方程

$$\begin{cases} -\frac{d^2 u(x)}{dx^2} = f(x), & 0 < x < 1, \\ u(0) = a, u(1) = b, \end{cases} \quad (6.5)$$

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  $h = \frac{1}{n+1}$ , 节点为  $x_i = ih$ ,  $i = 0, 1, 2, \dots, n+1$ .

我们采用中心差分离散, 可得 ( $i = 1, 2, \dots, n$ )

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

代入 (6.5), 舍去高阶项后可得 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

$$T_n u = f, \quad (6.6)$$

其中

$$T_n = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix}, \quad f = \begin{bmatrix} f_1 + u_0 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n + u_{n+1} \end{bmatrix}. \quad (6.7)$$





## 系数矩阵 $T_n$ 的性质

**引理**  $T_n$  的特征值和对应的特征向量分别为

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

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

即  $T_n = Z\Lambda Z^\top$ , 其中  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ ,  $Z = [z_1, z_2, \dots, z_n]$ .



**引理** 更一般地, 设  $T = \text{tridiag}(a, b, c) \in \mathbb{R}^{n \times n}$ , 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

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

(证明留作练习)



由前面的结论可知,  $T_n$  是对称正定的, 其最大特征值为

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

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

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



矩阵  $T_n$  可以分解为  $T_n = DD^T$ , 其中

$$D = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{n \times (n+1)}.$$

矩阵  $D$  也通常称为 **差分矩阵**. 需要注意的是,  $D$  不是方阵, 因此不能用这个分解来求解线性方程组  $T_n x = b$ .



## 2.2 二维 Poisson 方程

现在考虑二维 Poisson 方程

$$\begin{cases} -\Delta u(x, y) = -\frac{\partial^2 u(x, y)}{\partial x^2} - \frac{\partial^2 u(x, y)}{\partial y^2} = f(x, y), & (x, y) \in \Omega, \\ u(x, y) = u_0(x, y), & (x, y) \in \partial\Omega \end{cases} \quad (6.8)$$

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  $h = \frac{1}{n+1}$ , 节点设为  $x_i = ih$ ,  $y_j = jh$ ,  $i, j = 0, 1, 2, \dots, n$ . 在  $x$ -方向和  $y$ -方向同时采用中心差分离散可得

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

代入 (6.8), 即得二维 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  $f_{ij} = f(x_i, y_j)$ ,  $u_{i,j}$  为  $u(x_i, y_j)$  的近似.




写成矩阵形式即为

$$T_N u = h^2 f, \quad (6.9)$$

其中

$$T_N \triangleq I \otimes T_n + T_n \otimes I, \quad N = n^2,$$

$$u = [u_{1,1}, \dots, u_{n,1}, u_{1,2}, \dots, u_{n,2}, \dots, u_{1,n}, \dots, u_{n,n}].$$

 在后面介绍算法时, 我们以二维离散 Poisson 方程 (6.9) 为例.



## 系数矩阵 $T$ 的性质

因为  $T_N = I \otimes T_n + T_n \otimes I$ , 由 Kronecker 乘积的性质即得

**定理** 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  $T$  的特征值分解为

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

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

条件数

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





## 二维离散 Poisson 方程的 Jacobi 迭代方法

### 算法 2.1 求解二维离散 Poisson 方程的 Jacobi 迭代方法

- 1: Choose an initial guess  $v^{(0)}$
- 2: **while** not converge **do**
- 3:     **for**  $i = 1$  to  $N$  **do**
- 4:         **for**  $j = 1$  to  $N$  **do**
- 5:              $u_{i,j}^{(k+1)} = \left( h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)} \right) / 4$
- 6:             **end for**
- 7:     **end for**
- 8: **end while**

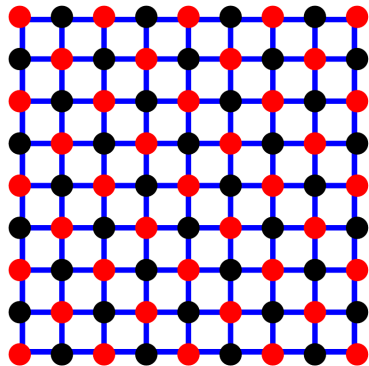


## G-S 算法的并行计算: 红黑排序

下面我们介绍一种适合并行计算的更新顺序:  
**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, 对未知量的值进行更新时, 我们可以先更新红色节点, 此时所使用的只是黑色节点的数据, 然后再更新黑色节点, 这时使用的是红色节点的数据. 于是我们得到**红黑排序 G-S 迭代方法**.

由于在更新红点时, 各个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, 各个点之间也是相互独立的, 因此也可以并行计算.





## 算法 2.2 求解二维离散 Poisson 方程的红黑排序 G-S 迭代方法

---

- 1: Choose an initial guess  $v^{(0)}$
  - 2: **while** not converge **do**
  - 3:     **for**  $(i, j)$  为红色节点 **do**
  - 4:         
$$u_{i,j}^{(k+1)} = \frac{1}{4} \left( h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)} \right)$$
  - 5:     **end for**
  - 6:     **for**  $(i, j)$  为黑色节点 **do**
  - 7:         
$$u_{i,j}^{(k+1)} = \frac{1}{4} \left( h^2 f_{i,j} + u_{i+1,j}^{(k+1)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k+1)} + u_{i,j-1}^{(k+1)} \right)$$
  - 8:     **end for**
  - 9: **end while**
-



### 算法 2.3 求解二维离散 Poisson 方程的红黑排序 SOR 迭代方法

- 1: Choose an initial guess  $v^{(0)}$  and parameter  $\omega$
- 2: **while** not converge **do**
- 3:     **for**  $(i, j)$  为红色节点 **do**
- 4:          $u_{i,j}^{(k+1)} = (1 - \omega)v_{i,j}^{(k)} + \omega(h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)})/4$
- 5:     **end for**
- 6:     **for**  $(i, j)$  为黑色节点 **do**
- 7:          $u_{i,j}^{(k+1)} = (1 - \omega)v_{i,j}^{(k)} + \omega(h^2 f_{i,j} + u_{i+1,j}^{(k+1)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k+1)} + u_{i,j-1}^{(k+1)})/4$
- 8:     **end for**
- 9: **end while**

## 二维离散 Poisson 方程的常用算法

	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	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N \log N)$
经典迭代	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)	$\mathcal{O}(N^{5/4})$	$\mathcal{O}(N)$
快速算法	FFT (快速 Fourier 变换)	$\mathcal{O}(N \log N)$	$\mathcal{O}(N)$
	块循环约化	$\mathcal{O}(N \log N)$	$\mathcal{O}(N)$
	Multigrid	$\mathcal{O}(N)$	$\mathcal{O}(N)$



# 3 | 收敛性分析

3.1 定常迭代方法的收敛性

3.2 二维离散 Poisson 方程情形

3.3 不可约对角占优矩阵情形

3.4 对称正定矩阵情形

3.5 相容次序矩阵



## 3.1 定常迭代方法的收敛性

### 向量序列的收敛

设  $\{x^{(k)}\}_{k=1}^{\infty}$  是  $\mathbb{C}^n$  中的一个向量序列, 如果存在  $x \in \mathbb{C}^n$ , 使得

$$\lim_{k \rightarrow \infty} x_i^{(k)} = x_i, \quad i = 1, 2, \dots, n,$$

则称  $\{x^{(k)}\}$  (按分量) 收敛到  $x$ , 记为  $\lim_{k \rightarrow \infty} x^{(k)} = x$ .

**定理 (收敛性判断)** 设  $\|\cdot\|$  是  $\mathbb{C}^n$  上的任意一个向量范数, 则  $\lim_{k \rightarrow \infty} x^{(k)} = x$  的充要条件是


$$\lim_{k \rightarrow \infty} \|x^{(k)} - x\| = 0.$$



**定义 (迭代方法的收敛性)** 如果对任意的初始向量  $x^{(0)}$ , 都有

$$\lim_{k \rightarrow \infty} x^{(k)} \rightarrow x_*,$$

则称迭代格式 (6.2) 是**收敛**的, 否则就称其为**发散**的.

 基于矩阵分裂的迭代方法, 其收敛性取决于迭代矩阵的谱半径.





## 矩阵谱半径

设  $A \in \mathbb{R}^{n \times n}$ , 则称

$$\rho(A) \triangleq \max_{\lambda \in \sigma(A)} |\lambda|$$

为  $A$  的谱半径, 其中  $\sigma(A)$  表示  $A$  的所有特征值组成的集合.

**引理 (谱半径与范数的关系)** 设  $G \in \mathbb{R}^{n \times n}$ , 则

- (1) 对任意算子范数, 有  $\rho(G) \leq \|G\|$ ;
- (2) 反之, 对任意  $\varepsilon > 0$ , 都存在一个算子范数  $\|\cdot\|_\varepsilon$ , 使得  $\|G\|_\varepsilon \leq \rho(G) + \varepsilon$ , 其中范数  $\|\cdot\|_\varepsilon$  依赖于  $G$  和  $\varepsilon$ . 所以, 若  $\rho(G) < 1$ , 则存在算子范数  $\|\cdot\|_\varepsilon$ , 使得  $\|G\|_\varepsilon < 1$ ;



由此, 我们可以立即得到下面的结论.

**定理** 设矩阵  $G \in \mathbb{R}^{n \times n}$ , 则  $\lim_{k \rightarrow \infty} G^k = 0$  当且仅当  $\rho(G) < 1$ .

下面的结论是谱半径与算子范数之间的一个非常重要的性质.

**引理** 设  $G \in \mathbb{R}^{n \times n}$ , 则对任意算子范数  $\|\cdot\|$ , 有

$$\rho(G) = \lim_{k \rightarrow \infty} \|G^k\|^{\frac{1}{k}}.$$




## 迭代方法收敛性判断

首先给出一个迭代方法收敛的充分条件.

**引理** 若存在算子范数  $\|\cdot\|$ , 使得  $\|G\| < 1$ , 则迭代方法 6.2 收敛. (板书)



 我们记  $e^{(k)} \triangleq x^{(k)} - x_*$  为第  $k$  步迭代解  $x^{(k)}$  的误差向量.

**定理 (收敛性定理)** 对任意迭代初始向量  $x^{(0)}$ , 迭代方法 6.2 收敛的充要条件是  $\rho(G) < 1$ . (板书)




**定义** 设  $G$  是迭代矩阵, 则迭代方法 6.2 的**平均收敛速度**定义为

$$R_k(G) \triangleq -\ln \|G^k\|^{\frac{1}{k}},$$

**渐进收敛速度**定义为

$$R(G) \triangleq \lim_{k \rightarrow \infty} R_k(G) = -\ln \rho(G).$$

 平均收敛速度与迭代步数和所用的范数有关, 但渐进收敛速度只依赖于迭代矩阵的谱半径.




**定理** 考虑算法 6.2. 如果存在某个算子范数  $\|\cdot\|$  使得  $\|G\| = q < 1$ , 则

$$(1) \|x^{(k)} - x_*\| \leq q^k \|x^{(0)} - x_*\|;$$

$$(2) \|x^{(k)} - x_*\| \leq \frac{q}{1-q} \|x^{(k)} - x^{(k-1)}\|;$$

$$(3) \|x^{(k)} - x_*\| \leq \frac{q^k}{1-q} \|x^{(1)} - x^{(0)}\|.$$

(板书)

 一般来说, 好的迭代方法应该满足:

(1)  $\rho(G)$  很小;

(2) 以  $M$  为系数矩阵的线性方程组比较容易求解.



## 3.2 二维离散 Poisson 方程情形

**充要条件** : 迭代矩阵的谱半径小于 1.

**充分条件** : 迭代矩阵的某个算子范数小于 1.

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 算法的迭代矩阵为

$$G_J = D^{-1}(L + U) = (4I)^{-1}(4I - T) = I - \frac{1}{4}T. \quad (6.10)$$

由于  $T$  的特征值为

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




所以  $G_J$  的特征值为

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

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 算法是收敛的.

 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 算法收敛越来越慢.





## G-S 和 SOR

设  $G_{GS}$  和  $G_{SOR}$  分别表示求解二维 Poisson 方程的红黑排序的 G-S 算法和 SOR 算法的迭代矩阵, 则有

$$\rho(G_{GS}) = \rho(G_J)^2 = \cos^2 \frac{\pi}{n+1} < 1 \quad (6.11)$$

$$\rho(G_{SOR}) = \frac{\cos^2 \frac{\pi}{n+1}}{\left(1 + \sin \frac{\pi}{n+1}\right)^2} < 1, \quad \omega = \frac{2}{1 + \sin \frac{\pi}{n+1}}. \quad (6.12)$$

(证明见讲义, 留作自习)

在上述结论中, SOR 算法中的  $\omega$  是最优参数, 即此时的  $\rho(G_{SOR})$  最小.



## Jacobi 和 SOR

由 Taylor 公式可知, 当  $n$  很大时, 有

$$\begin{aligned}\rho(G_J) &= \cos \frac{\pi}{n+1} \approx 1 - \frac{\pi^2}{2(n+1)^2} = 1 - O\left(\frac{1}{n^2}\right), \\ \rho(G_{\text{SOR}}) &= \frac{\cos^2 \frac{\pi}{n+1}}{\left(1 + \sin \frac{\pi}{n+1}\right)^2} \approx 1 - \frac{2\pi}{n+1} = 1 - O\left(\frac{1}{n}\right).\end{aligned}$$

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  $k$  步后的误差下降量与 Jacobi 迭代  $kn$  步后的误差下降量相当, 即带最优参数的 SOR 的收敛速度大约是 Jacobi 的  $n$  倍.



事实上, 当  $n$  很大时, 这三个算法的收敛速度都很慢.

**例** 已知二维 Poisson 方程

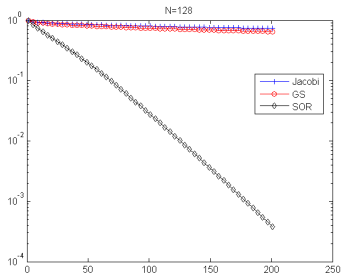
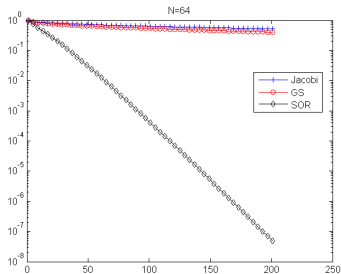
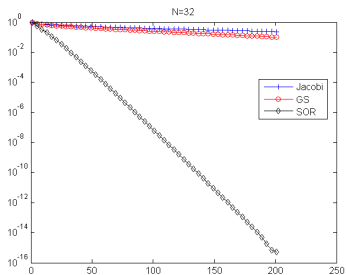
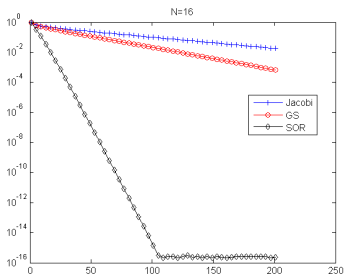
$$\begin{cases} -\Delta u(x, y) = -1, & (x, y) \in \Omega \\ u(x, y) = \frac{x^2 + y^2}{4}, & (x, y) \in \partial\Omega \end{cases}$$

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, 分别用 Jacobi, G-S 和 SOR 算法计算这个方程组的解, 并比较收敛效果.

(Poisson\_Jacobi\_GS\_SOR.m)



下图画出了  $N = 16, 32, 64, 128$  时, 三种算法的相对误差下降情况.





## 3.3 不可约对角占优矩阵情形


这里我们考虑  $A$  是严格对角占优或不可约弱对角占优情形.


**定理** 设  $A \in \mathbb{R}^{n \times n}$ , 若  $A$  严格对角占优, 则 Jacobi 算法和 G-S 算法都收敛, 且

$$\|G_{GS}\|_{\infty} \leq \|G_J\|_{\infty} < 1.$$



**定理** 设  $A \in \mathbb{R}^{n \times n}$ , 若  $A$  是弱对角占优且不可约, 则 Jacobi 算法和 G-S 算法都收敛, 且  $\rho(G_{GS}) < \rho(G_J) < 1$ . (证明留作练习)

 二维离散 Poisson 方程是弱行对角占优且不可约, 故对 Jacobi 算法和 G-S 算法都收敛.

 上述定理中的结论对一般矩阵并不成立: 对某些矩阵, Jacobi 算法收敛, 但 G-S 算法却不一定收敛.



关于 SOR 方法, 我们有下面的结论.

**定理** 设  $A \in \mathbb{R}^{n \times n}$ , 若  $A$  严格对角占优且  $0 < \omega \leq 1$ , 则 SOR 方法收敛.

**定理** 设  $A \in \mathbb{R}^{n \times n}$ , 若  $A$  是弱对角占优且不可约, 且  $0 < \omega \leq 1$ , 则 SOR 方法收敛.  
(证明留作练习)



## 3.4 对称正定矩阵情形

在给出收敛性结论之前, 也介绍两个需要用到的引理.

**引理** 设  $A \in \mathbb{C}^{n \times n}$  Hermite 对称, 且  $A = M - N$  是  $A$  的一个矩阵分裂, 则  $M^* + N$  也是 Hermite 对称, 且对任意  $x \in \mathbb{C}^n$  有

$$x^* Ax - \tilde{x}^* A \tilde{x} = u^* (M^* + N) u,$$

其中  $\tilde{x} = M^{-1}Nx$ ,  $u = x - \tilde{x}$ .

(板书)





**引理** 设  $A \in \mathbb{R}^{n \times n}$  对称, 且  $A = M - N$  是  $A$  的一个矩阵分裂.

- (1) 若  $A$  和  $M^T + N$  都是正定矩阵, 则  $M$  非奇异且  $\rho(M^{-1}N) < 1$ ;
- (2) 如果  $\rho(M^{-1}N) < 1$  且  $M^T + N$  正定, 则  $A$  正定.

(板书)



我们首先给出 SOR 迭代收敛的一个必要条件.


**定理** 对于 SOR 算法, 有  $\rho(G_{\text{SOR}}) \geq |1 - \omega|$ , 故 SOR 算法收敛的必要条件是  $0 < \omega < 2$ .



**定理** 设  $A \in \mathbb{R}^{n \times n}$  对称正定.

- (1) 若  $2D - A$  正定, 则 Jacobi 迭代收敛.
- (2) 若  $0 < \omega < 2$ , 则 SOR 和 SSOR 收敛.
- (3) G-S 迭代收敛.

 若系数矩阵对称正定, 则 SOR 收敛的充要条件是  $0 < \omega < 2$ .

 对于二维离散 Poisson 方程, 其系数矩阵是对称正定的, 故当  $0 < \omega < 2$  时, SOR 算法收敛.



**定理** 设  $A \in \mathbb{R}^{n \times n}$  对称.

- (1) 若  $2D - A$  正定且 Jacobi 迭代收敛, 则  $A$  正定;
- (2) 若  $D$  正定, 且存在  $\omega \in (0, 2)$  使得 SOR (或 SSOR) 收敛, 则  $A$  正定;
- (3) 若  $D$  正定, 且 G-S 迭代收敛, 则  $A$  正定.

(证明留作练习)



## 3.5 相容次序矩阵

针对一类特殊的矩阵, 这三种迭代方法的谱半径之间存在一种特殊关系.

**定义** 设  $A \in \mathbb{R}^{n \times n}$ , 如果存在一个置换矩阵  $P$ , 使得

$$PAP^T = \begin{bmatrix} D_1 & F \\ E & D_2 \end{bmatrix}, \quad (6.13)$$

其中  $D_1, D_2$  为对角矩阵, 则称  $A$  具有**性质 A**.

**例** 对于二维离散 Poisson 方程, 系数矩阵  $T_{N^2}$  具有性质 A. 事实上, 设  $\tilde{T}_{N^2}$  为模型问题采用红黑排序后的系数矩阵, 则  $\tilde{T}_{N^2}$  具有 (6.13) 的结构.



我们首先给出一个性质.

**引理** 设  $B \in \mathbb{R}^{n \times n}$  具有下面的结构

$$B = \begin{bmatrix} 0 & B_{12} \\ B_{21} & 0 \end{bmatrix},$$

令  $B_L$  和  $B_U$  分别表示  $B$  的下三角和上三角部分, 则

- (1) 若  $\mu$  是  $B$  的特征值, 则  $-\mu$  也是  $B$  的特征值;
- (2)  $B(\alpha)$  的特征值与  $\alpha$  无关, 其中

$$B(\alpha) = \alpha B_L + \frac{1}{\alpha} B_U, \quad \alpha \neq 0.$$

(板书)

$B(\alpha) + \beta I$  的特征值也与  $\alpha$  无关, 其中  $\beta$  为任意常数.



👉 该结论可以推广到块三对角形式, 见习题.

设  $A \in \mathbb{R}^{n \times n}$  的对角线元素全不为零, 记  $\tilde{L} = D^{-1}L$ ,  $\tilde{U} = D^{-1}U$ .

**定义** 设  $A \in \mathbb{R}^{n \times n}$  的对角线元素全不为零,  $A = D(I - \tilde{L} - \tilde{U})$ . 若矩阵  $G(\alpha) = \alpha\tilde{L} + \frac{1}{\alpha}\tilde{U}$  的特征值与  $\alpha$  无关, 则称  $A$  具有**相容次序**.

👉 设  $A$  的对角线元素全不为零, 若  $A$  具有性质 A, 则存在置换矩阵  $P$ , 使得  $PAP^T$  具有相容次序.



**定理** 设  $A$  有相容次序且  $\omega \neq 0$ , 则下列命题成立

- (1) Jacobi 迭代矩阵  $G_J$  的特征值正负成对出现;
- (2) 若  $\mu$  是  $G_J$  的特征值且  $\lambda$  满足

$$(\lambda + \omega - 1)^2 = \lambda \omega^2 \mu^2, \quad (6.14)$$

则  $\lambda$  是 SOR 迭代矩阵  $G_{\text{SOR}}$  的一个特征值;

- (3) 反之, 若  $\lambda \neq 0$  是  $G_{\text{SOR}}$  的一个特征值且  $\mu$  满足 (6.14), 则  $\mu$  是  $G_J$  的一个特征值.

(板书)





**推论** 若  $A$  具有相容次序, 则  $\rho(G_{GS}) = \rho(G_J)^2$ , 即当 Jacobi 算法收敛时, G-S 算法比 Jacobi 算法快一倍.

**例** 采用红黑排序的二维离散 Poisson 方程, 系数矩阵  $\tilde{T}_{N^2}$  具有相容次序, 故有  $\rho(G_{GS}) = \rho(G_J)^2$ .



下面是关于 SOR 算法的最优参数选取.

**定理** 设  $A$  具有相容次序,  $G_J$  的特征值全部为实数, 且  $\rho_J = \rho(G_J) < 1$ , 则 SOR 算法的最优参数和对应的谱半径分别为:

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho_J^2}}, \quad \rho(G_{SOR}) = \omega_{opt} - 1 = \frac{\rho_J^2}{\left(1 + \sqrt{1 - \rho_J^2}\right)^2}.$$

进一步, 有

$$\rho(G_{SOR}) = \begin{cases} \omega - 1, & \omega_{opt} \leq \omega \leq 2 \\ 1 - \omega + \frac{1}{2}\omega^2\rho_J^2 + \omega\rho_J\sqrt{1 - \omega + \frac{1}{4}\omega^2\rho_J^2}, & 0 < \omega \leq \omega_{opt} \end{cases}$$

(证明留作练习)



**例** 采用红黑排序的二维离散 Poisson 问题的系数矩阵  $\tilde{T}_{N^2}$  具有相容次序, 且  $G_J$  是对称的, 即  $G_J$  的特征值都是实的. 又由系数矩阵的弱对角占优和不可约性质可知  $\rho(G_J) < 1$ , 故上述定理的条件均满足.



# 4 | 加速方法

## 4.1 外推技术

## 4.2 Chebyshev 加速

当迭代解  $x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(k)}$  已经计算出来后, 我们可以对其进行组合, 得到一个新的近似解, 这样就可以对原算法进行加速.



## 4.1 外推技术

设原迭代格式为

$$x^{(k+1)} = Gx^{(k)} + b. \quad (6.15)$$

由  $x^{(k)}$  和  $x^{(k+1)}$  加权组合后可得新的近似解

$$x^{(k+1)} = (1 - \omega)x^{(k)} + \omega(Gx^{(k)} + b), \quad (6.16)$$

其中  $\omega$  是参数. 这种加速方法就称为 **外推算法**.

为了使得迭代格式 (6.16) 尽可能快地收敛, 需要选择  $\omega$  使得其迭代矩阵  $G_\omega \triangleq (1 - \omega)I + \omega G$  的谱半径尽可能地小. 假设  $G$  的特征值都是实数, 且最大特征值和最小特征值分别为  $\lambda_1$  和  $\lambda_n$ . 于是

$$\rho(G_\omega) = \max_{\lambda \in \sigma(G)} |(1 - \omega) + \omega\lambda| = \max\{|1 - \omega + \omega\lambda_1|, |1 - \omega + \omega\lambda_n|\}.$$



**定理** 设  $G$  的特征值都是实数, 其最大和最小特征值分别为  $\lambda_1$  和  $\lambda_n$ , 且  $1 \notin [\lambda_n, \lambda_1]$ , 则

$$\omega_* = \arg \min_{\omega} \rho(G_{\omega}) = \frac{2}{2 - (\lambda_1 + \lambda_n)},$$

此时

$$\rho(G_{\omega_*}) = 1 - |\omega_*|d,$$

其中  $d$  是 1 到  $[\lambda_n, \lambda_1]$  的距离, 即当  $\lambda_n \leq \lambda_1 < 1$  时,  $d = 1 - \lambda_1$ , 当  $\lambda_1 \geq \lambda_n > 1$  时,  $d = \lambda_n - 1$ . (证明见讲义, 留作自习)

由定理可知,  $\rho(G_{\omega_*}) = 1 - |\omega_*|d$ , 且当  $\omega_* \neq 1$  时, 外推迭代 (6.16) 比原迭代方法收敛要更快一些.

最优参数依赖于原迭代矩阵  $G$  的特征值, 因此实用性不强. 在实际应用时可以估计特征值所在的区间  $[a, b]$ , 然后用  $a, b$  代替  $\lambda_n$  和  $\lambda_1$ .



## JOR 算法

对 Jacobi 迭代进行外推加速, 则可得 JOR (Jacobi over-relaxation) 算法:

$$\begin{aligned}x^{(k+1)} &= (1 - \omega)x^{(k)} + \omega(D^{-1}(L + U)x^{(k)} + D^{-1}b) \\&= x^{(k)} + \omega D^{-1}(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots\end{aligned}$$

**定理** 设  $A$  对称正定. 若

$$0 < \omega < \frac{2}{\rho(D^{-1}A)},$$

则 JOR 算法收敛.



## 4.2 Chebyshev 加速

本节对外推技巧进行推广.

假定通过迭代格式 (6.15) 已经计算出  $x^{(0)}, x^{(1)}, \dots, x^{(k)}$ , 下面考虑如何将这些近似解进行组合, 以便得到更精确的近似解.

记  $\varepsilon_k = x^{(k)} - x_*$  为第  $k$  步迭代解的误差, 则有

$$\varepsilon_k = G\varepsilon_{k-1} = G^2\varepsilon_{k-2} = \cdots = G^k\varepsilon_0.$$

设  $\tilde{x}^{(k)}$  为  $x^{(0)}, x^{(1)}, \dots, x^{(k)}$  的一个线性组合, 即

$$\tilde{x}^{(k)} = \alpha_0 x^{(0)} + \alpha_1 x^{(1)} + \cdots + \alpha_k x^{(k)}, \quad (6.17)$$

其中  $\alpha_i$  为待定系数, 且满足  $\sum_{i=0}^k \alpha_i = 1$ . 于是

$$\tilde{x}^{(k)} - x_* = \alpha_0 \varepsilon_0 + \alpha_1 G\varepsilon_0 + \cdots + \alpha_k G^k \varepsilon_0 \triangleq p_k(G)\varepsilon_0, \quad (6.18)$$





其中  $p_k(t) = \sum_{i=0}^k \alpha_i t^i$  为  $k$  次多项式, 且满足  $p_k(1) = 1$ .

我们希望通过适当选取参数  $\alpha_i$ , 使得  $\tilde{x}^{(k)} - x_*$  尽可能地小, 即使得  $\tilde{x}^{(k)}$  收敛到  $x_*$  速度远远快于  $x^{(k)}$  收敛到  $x_*$  速度. 这种加速方法就称为**多项式加速**或**半迭代方法** (semi-iterative method) .

**例** 设  $p_n(t)$  为  $G$  的特征多项式, 则  $p_n(G) = 0$ , 所以选取  $\alpha_i$  为  $p_n$  的系数, 则  $\tilde{x}^{(n)} - x_* = 0$ . 但这种选取方法不实用, 原因是:

- (1)  $p_n(t)$  的系数并不知道;
- (2) 我们通常希望收敛所需的迭代步数  $\ll n$ .



下面讨论参数  $\alpha_i$  的较实用的选取方法. 由 (6.18) 可知

$$\|\tilde{x}^{(k)} - x_*\|_2 = \|p_k(G)\varepsilon_0\|_2 \leq \|p_k(G)\|_2 \cdot \|\varepsilon_0\|_2.$$

因此我们需要求解下面的极小化问题

$$\min_{p \in \mathbb{P}_k, p(1)=1} \|p(G)\|_2, \quad (6.19)$$

其中  $\mathbb{P}_k$  表示所有次数不超过  $k$  的多项式组成的集合.

一般来说, 这个问题是非常困难的.

但在一些特殊情况下, 我们可以给出其最优解.



假设迭代矩阵  $G$  是对称的, 即  $G$  存在特征值分解:

$$G = Q\Lambda Q^T,$$

其中  $\Lambda$  是实对角矩阵,  $U$  是正交矩阵. 于是有

$$\begin{aligned}\min_{p \in \mathbb{P}_k, p(1)=1} \|p(G)\|_2 &= \min_{p \in \mathbb{P}_k, p(1)=1} \|p(\Lambda)\|_2 \\ &= \min_{p \in \mathbb{P}_k, p(1)=1} \max_{1 \leq i \leq n} \{|p(\lambda_i)|\} \\ &\leq \min_{p \in \mathbb{P}_k, p(1)=1} \max_{\lambda \in [\lambda_n, \lambda_1]} \{|p(\lambda)|\},\end{aligned}\tag{6.20}$$

其中  $\lambda_1, \lambda_n$  分别表示  $G$  的最大和最小特征值.

这是带归一化条件的多项式最佳一致逼近问题 (与零的偏差最小).

该问题的解与著名的 **Chebyshev 多项式** 有关.



## Chebyshev 多项式

Chebyshev 多项式  $T_k(t)$  可以通过下面的递推方式来定义:

$$\begin{aligned} T_0(t) &= 1, \quad T_1(t) = t, \\ T_k(t) &= 2tT_{k-1}(t) - T_{k-2}(t), \quad k = 2, 3, \dots, \end{aligned} \quad (6.21)$$

也可以直接由下面的式子定义

$$T_k(t) = \frac{1}{2} \left[ \left( t + \sqrt{t^2 - 1} \right)^k + \left( t - \sqrt{t^2 - 1} \right)^k \right],$$

或者

$$T_k(t) = \begin{cases} \cos(k \arccos t), & |t| \leq 1 \\ \cosh(k \operatorname{arccosh} t), & |t| > 1 \end{cases},$$



Chebyshev 的一个重要性质是下面的最小最大性质.

**定理** 设  $\eta \in \mathbb{R}$  满足  $|\eta| > 1$ , 则下面的最小最大问题

$$\min_{p(t) \in \mathbb{P}_k, p(\eta)=1} \max_{-1 \leq t \leq 1} |p(t)|$$

的唯一解为

$$\tilde{T}_k(t) \triangleq \frac{T_k(t)}{T_k(\eta)}.$$



通过简单的仿射变换, 该定理的结论可以推广到一般区间.

**定理** 设  $\alpha, \beta, \eta \in \mathbb{R}$  满足  $\alpha < \beta$  且  $|\eta| \notin [\alpha, \beta]$ . 则下面的最小最大问题

$$\min_{p(t) \in \mathbb{P}_k, p(\eta)=1} \max_{\alpha \leq x \leq \beta} |p(t)|$$

的唯一解为

$$\hat{T}_k(t) \triangleq \frac{T_k\left(\frac{2t - (\beta + \alpha)}{\beta - \alpha}\right)}{T_k\left(\frac{2\eta - (\beta + \alpha)}{\beta - \alpha}\right)}.$$



## Chebyshev 加速方法

考虑迭代格式  $x^{(k+1)} = Gx^{(k)} + b$ , 我们假定:

- (1) 迭代矩阵  $G$  的特征值都是实数;
- (2) 迭代矩阵谱半径  $\rho = \rho(G) < 1$ , 故  $\lambda(G) \in [-\rho, \rho] \subset (-1, 1)$ .

于是最小最大问题 (6.20) 就转化为

$$\min_{p \in \mathbb{P}_k, p(1)=1} \max_{\lambda \in [-\rho, \rho]} \{|p(\lambda)|\}.$$

由于  $1 \notin [-\rho, \rho]$ , 根据定理 4.4, 上述问题的解为

$$p_k(t) = \frac{T_k(t/\rho)}{T_k(1/\rho)}.$$



## $\tilde{x}^{(k)}$ 的计算

实际计算时, 我们无需先计算  $x^{(0)}, x^{(1)}, \dots, x^{(k)}$ , 然后通过线性组合 (6.17) 来计算  $\tilde{x}^{(k)}$ .

事实上, 我们可以通过 Chebyshev 多项式的三项递推公式 (6.21), 由  $\tilde{x}^{(k-1)}$  和  $\tilde{x}^{(k-2)}$  直接计算出  $\tilde{x}^{(k)}$ .

具体的推导公式如下:

令  $\mu_k = \frac{1}{T_k(1/\rho)}$ , 即  $T_k(1/\rho) = \frac{1}{\mu_k}$ . 由三项递推公式 (6.21) 可得

$$\frac{1}{\mu_k} = \frac{2}{\rho} \cdot \frac{1}{\mu_{k-1}} - \frac{1}{\mu_{k-2}}.$$





所以

$$\begin{aligned}\tilde{x}^{(k)} - x_* &= p_k(G) \varepsilon_0 = \mu_k T_k(G/\rho) \varepsilon_0 \\ &= \mu_k \left[ \frac{2G}{\rho} \cdot \frac{1}{\mu_{k-1}} (\tilde{x}^{(k-1)} - x_*) - \frac{1}{\mu_{k-2}} (\tilde{x}^{(k-2)} - x_*) \right].\end{aligned}$$

整理后可得

$$\tilde{x}^{(k)} = \frac{2\mu_k}{\mu_{k-1}} \cdot \frac{G}{\rho} \tilde{x}^{(k-1)} - \frac{\mu_k}{\mu_{k-2}} \tilde{x}^{(k-2)} + d_k,$$

其中

$$\begin{aligned}d_k &= x_* - \frac{2\mu_k}{\mu_{k-1}} \cdot \frac{G}{\rho} x_* + \frac{\mu_k}{\mu_{k-2}} x_* = x_* - \frac{2\mu_k}{\mu_{k-1}} \cdot \frac{x_* - g}{\rho} + \frac{\mu_k}{\mu_{k-2}} x_* \\ &= \mu_k \left( \frac{1}{\mu_k} - \frac{2}{\rho \mu_{k-1}} + \frac{1}{\mu_{k-2}} \right) x_* + \frac{2\mu_k g}{\mu_{k-1} \rho} = \frac{2\mu_k g}{\mu_{k-1} \rho}.\end{aligned}$$



由此, 我们可以得到迭代格式 (6.15) 的 Chebyshev 加速算法.

---

#### 算法 4.1 Chebyshev 加速算法

---

- 1: Set  $\mu_0 = 1, \mu_1 = \rho = \rho(G), \tilde{x}^{(0)} = x^{(0)}, k = 1$
  - 2: compute  $\tilde{x}^{(1)} = Gx^{(0)} + g$
  - 3: **while** not converge **do**
  - 4:      $k = k + 1$
  - 5:     
$$\mu_k = \left( \frac{2}{\rho} \cdot \frac{1}{\mu_{k-1}} - \frac{1}{\mu_{k-2}} \right)^{-1}$$
  - 6:     
$$\tilde{x}^{(k)} = \frac{2\mu_k}{\mu_{k-1}} \cdot \frac{G}{\rho} \tilde{x}^{(k-1)} - \frac{\mu_k}{\mu_{k-2}} \tilde{x}^{(k-2)} + \frac{2\mu_k}{\mu_{k-1}\rho} \cdot g$$
  - 7: **end while**
- 

该算法的每步迭代的整体运算量与原迭代格式基本相当.



## SSOR 算法的 Chebyshev 加速

SSOR 迭代矩阵为

$$G_{\text{SSOR}} = (D - \omega U)^{-1} [(1 - \omega)D + \omega L] (D - \omega L)^{-1} [(1 - \omega)D + \omega U].$$

当  $A$  对称时, 有  $L = U^T$ , 故

$$\begin{aligned} & (D - \omega U)G_{\text{SSOR}}(D - \omega U)^{-1} \\ &= [(1 - \omega)D + \omega L] (D - \omega L)^{-1} [(1 - \omega)D + \omega L^T] (D - \omega L^T)^{-1} \\ &= [(2 - \omega)D(D - \omega L)^{-1} - I] [(2 - \omega)D(D - \omega L^T)^{-1} - I] \\ &= I - (2 - \omega)D[(D - \omega L)^{-1} + (D - \omega L^T)^{-1}] \\ &\quad + (2 - \omega)^2 D(D - \omega L)^{-1} D(I - \omega L^T)^{-1}. \end{aligned}$$



假定  $D$  的对角线元素全是正的, 则

$$\begin{aligned} & D^{-1/2}(D - \omega U)G_{\text{SSOR}}(D - \omega U)^{-1}D^{1/2} \\ &= I - (2 - \omega)D^{-1/2}[(D - \omega L)^{-1} + (D - \omega L^T)^{-1}]D^{1/2} \\ &\quad + (2 - \omega)^2 D^{-1/2}(D - \omega L)^{-1}D(I - \omega L^T)^{-1}D^{1/2}. \end{aligned}$$

这是一个对称矩阵, 故  $G_{\text{SSOR}}$  具有实特征值. 所以我们可以对其实行 Chebyshev 加速. 但我们需要估计  $G_{\text{SSOR}}$  的谱半径.



# 5 | 交替方向与 HSS 算法

5.1 多步迭代法

5.2 交替方向法

5.3 HSS 方法



## 5.1 多步迭代法

设  $A = M_1 - N_1 = M_2 - N_2$  是  $A$  的两个矩阵分裂, 则可以构造迭代格式

$$\begin{cases} M_1 x^{(k+\frac{1}{2})} = N_1 x^{(k)} + b, \\ M_2 x^{(k+1)} = N_2 x^{(k+\frac{1}{2})} + b, \end{cases} \quad k = 0, 1, 2, \dots \quad (6.22)$$

这就是两步迭代方法, 对应的分裂称为二重分裂.

易知, 两步迭代格式 (6.22) 的迭代矩阵为

$$G = M_2^{-1} N_2 M_1^{-1} N_1.$$

因此, 其收敛的充要条件是  $\rho(M_2^{-1} N_2 M_1^{-1} N_1) < 1$ .

类似地, 我们可以推广到多步迭代方法.



## 5.2 交替方向法

交替方向法 (ADI) 本质上是一个两步迭代方法.

设  $A = A_1 + A_2$ , 则 ADI 迭代格式为

$$\begin{cases} (\alpha I + A_1)x^{(k+\frac{1}{2})} = (\alpha I - A_2)x^{(k)} + b, \\ (\alpha I + A_2)x^{(k+1)} = (\alpha I - A_1)x^{(k+\frac{1}{2})} + b, \end{cases} \quad k = 0, 1, 2, \dots,$$

其中  $\alpha \in \mathbb{R}$  是迭代参数.



易知 ADI 算法的迭代矩阵为

$$G_{\text{ADI}} = (\alpha I + A_2)^{-1}(\alpha I - A_1)(\alpha I + A_1)^{-1}(\alpha I - A_2).$$

**定理** 设  $A \in \mathbb{R}^{n \times n}$  对称正定,  $A = A_1 + A_2$ , 其中  $A_1$  和  $A_2$  中有一个是对称正定, 另一个是对称半正定, 则对任意正数  $\alpha > 0$ , 有  $\rho(G_{\text{ADI}}) < 1$ , 即 ADI 迭代方法收敛.





## 5.3 HSS 方法

设  $A = H + S$ , 其中  $H$  和  $S$  分别是  $A$  的对称与斜对称部分, 即

$$H = \frac{A + A^T}{2}, \quad S = \frac{A - A^T}{2}.$$

该分裂就称为 HS 分裂, 即 HSS.

类似于 ADI 方法, 我们可得下面的 HSS 方法

$$\begin{cases} (\alpha I + H)x^{(k+\frac{1}{2})} = (\alpha I - S)x^{(k)} + b, \\ (\alpha I + S)x^{(k+1)} = (\alpha I - H)x^{(k+\frac{1}{2})} + b, \end{cases} \quad k = 0, 1, 2, \dots$$

**定理** 设  $A \in \mathbb{R}^{n \times n}$  正定, 则对任意  $\alpha > 0$ , HSS 迭代方法都收敛.



## 参数 $\alpha$ 的选取

**定理** 设  $A \in \mathbb{R}^{n \times n}$  正定, 则极小极大问题

$$\min_{\alpha > 0} \max_{\lambda_{\min}(H) \leq \lambda \leq \lambda_{\max}(H)} \left| \frac{\alpha - \lambda}{\alpha + \lambda} \right|$$

的解为

$$\alpha_* = \sqrt{\lambda_{\max}(H)\lambda_{\min}(H)}.$$

此时

$$\sigma(\alpha_*) = \frac{\sqrt{\lambda_{\max}(H)} - \sqrt{\lambda_{\min}(H)}}{\sqrt{\lambda_{\max}(H)} + \sqrt{\lambda_{\min}(H)}} = \frac{\sqrt{\kappa(H)} - 1}{\sqrt{\kappa(H)} + 1}.$$

📖 HSS 推广: PSS, NSS, AHSS 等, 感兴趣的读者可以参考相关文献.



## 6 | 快速 Poisson 算法

如果已经知道  $A$  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, 则  $X$  可以取酉矩阵, 于是

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

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算法.



考虑二维离散 Poisson 方程

$$Tu = h^2 f, \quad (6.23)$$

其中

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

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

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

所以, 方程 (6.23) 的解为

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

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.



## 离散 Sine 变换

离散 Sine 变换有多种定义, 这里介绍与求解 Poisson 方程有关的一种.

设  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$ , 其 **离散 Sine 变换 (DST)** 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:

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

🔗 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`)



## Possion 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

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

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

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

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

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

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 6.1 二维离散 Poisson 方程的快速算法

---

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

MATLAB 程序见 [Poisson\\_DST.m](#)