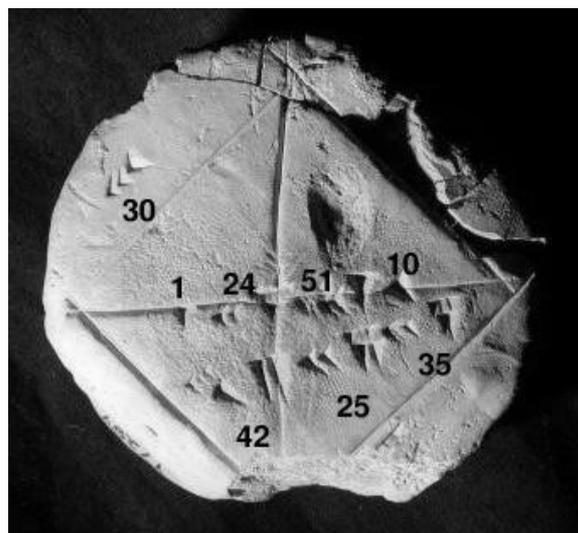


计算方法 (A)



童伟华 管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

中国科学技术大学 数学科学学院

<http://math.ustc.edu.cn/>





第六章 解线性方程组的迭代法

■ 求解线性方程的直接法：

- 时间复杂度： $O(n^3)$
- 空间复杂度： $O(n^2)$
- 理论上可经过有限次四则运算得到准确解，但因数值计算有舍入误差，得到的仍然是近似解
- 适用情况：中等规模

■ 求解线性方程的迭代法：

- 高阶稀疏线性方程组
- 主要运算：矩阵与向量的乘法
- 迭代格式的构造
- 收敛性、收敛速度

迭代法

- 基本思想：将线性方程组 $Ax = b$ 等价变形为 $x = Mx + g$ ，构造迭代关系式： $x^{(k+1)} = Mx^{(k)} + g$ 。若向量序列 $x^{(k)}$ 收敛到 x^* ，则

$$x^* = Mx^* + g \Leftrightarrow Ax^* = b$$

- 例如：

$$A = N - P \Rightarrow x = N^{-1}Px + N^{-1}b$$

- 如何设计迭代格式？
- 收敛性、收敛速度
- 收敛条件（是否与初始值相关）
- 优点：占用存储空间少，程序实现简单，尤其适合于高阶稀疏线性方程组

迭代法



■ 收敛性分析:

$$\begin{aligned}\mathbf{x}^{(k+1)} - \mathbf{x}^* &= \mathbf{M}\mathbf{x}^{(k)} - \mathbf{M}\mathbf{x}^* = \mathbf{M}(\mathbf{x}^{(k)} - \mathbf{x}^*) \\ &= \dots \quad (\mathbf{x}^{(0)} - \mathbf{x}^*)\end{aligned}$$

■ 向量序列 $\mathbf{x}^{(k)}$ 收敛 $\Leftrightarrow \lim_{k \rightarrow \infty} \mathbf{M}^k = \mathbf{0}$ 与初值的选取无关

■ 定理: 求解线性方程组迭代格式 $\mathbf{x}^{(k+1)} = \mathbf{M}\mathbf{x}^{(k)} + \mathbf{g}$ 收敛的充分必要条件是 $\rho(\mathbf{M}) < 1$

■ 推论: 若矩阵 \mathbf{M} 的范数 $\|\mathbf{M}\|_p < 1$, 则 $\mathbf{x}^{(k+1)} = \mathbf{M}\mathbf{x}^{(k)} + \mathbf{g}$ 收敛

■ 常用矩阵范数: $\|\mathbf{M}\|_1$ 或 $\|\mathbf{M}\|_\infty$

■ 注意: 当 $\|\mathbf{M}\|_1 \geq 1$ 或 $\|\mathbf{M}\|_\infty \geq 1$ 时, 不能断定迭代序列发散, 例如

$$\mathbf{M} = \begin{pmatrix} 0.9 & 0 \\ 0.2 & 0.8 \end{pmatrix}$$

Jacobi迭代

- 基本思想：求解第 i 个方程得到第 i 个未知量

$$\Rightarrow \begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n \\ x_1 = \frac{-1}{a_{11}}(a_{12}x_2 + \cdots - b_1) \\ x_2 = \frac{-1}{a_{22}}(a_{21}x_1 + a_{23}x_3 + \cdots - b_2) \\ \vdots \\ x_n = \frac{-1}{a_{nn}}(a_{n1}x_1 + \cdots x_{n-1} - b_n) \end{cases}$$

Jacobi迭代

■ Jacobi迭代格式:

$$\left\{ \begin{array}{l} x_1^{(k+1)} = \frac{-1}{a_{11}} (a_{12}x_2^{(k)} + \dots \quad x_1^{(k)} - b_1) \\ x_2^{(k+1)} = \frac{-1}{a_{22}} (a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + \dots \quad x_2^{(k)} - b_2) \\ \vdots \\ x_n^{(k+1)} = \frac{-1}{a_{nn}} (a_{n1}x_1^{(k)} + \dots \quad x_{n-1}^{(k)} - b_n) \end{array} \right.$$

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

Jacobi迭代

■ Jacobi迭代矩阵:

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{pmatrix} = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & \\ \frac{-a_{21}}{a_{11}} & 0 & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-a_{n1}}{a_{nn}} & \frac{-a_{n2}}{a_{nn}} & \dots & \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{pmatrix} + \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{11}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{pmatrix}$$

$$\mathbf{Ax} = \mathbf{b} \Leftrightarrow (\mathbf{D} - (\mathbf{D} - \mathbf{A}))\mathbf{x} = \mathbf{b}$$

$$\Leftrightarrow \mathbf{Dx} = (\mathbf{D} - \mathbf{A})\mathbf{x} + \mathbf{b}$$

$$\Leftrightarrow \mathbf{x} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b}$$

$$\Rightarrow \mathbf{M} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A}) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \quad , \quad \mathbf{g} = \mathbf{D}^{-1}\mathbf{b}$$

Jacobi迭代

- Jacobi迭代收敛条件的充分必要条件:

$$\rho(\mathbf{M}) < 1$$

- 定理: 若线性方程组 $\mathbf{Ax} = \mathbf{b}$ 的系数矩阵 \mathbf{A} 满足下列条件之一:

(1) \mathbf{A} 为严格行对角占优阵, 即 $|a_{ii}| > \sum_{j \neq i} |a_{ij}|, i = 1, 2, \dots$

(2) \mathbf{A} 为严格列对角占优阵, 即 $|a_{jj}| > \sum_{i \neq j} |a_{ij}|, j = 1, 2, \dots$

则Jacobi迭代收敛

- 通常, 对角元越占优, 收敛速度就越快; 但也有反例, 如:

$$\mathbf{A}_1 = \begin{bmatrix} 1 & -1/2 \\ -1/2 & 1 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} 1 & -3/4 \\ -1/12 & 1 \end{bmatrix}$$



Jacobi迭代

■ Jacobi迭代算法

Algorithm 15 Jacobi's Iteration Algorithm

Input:

$n, (a_{ij}), (b_i), (x_i), M, \varepsilon$

```
1: for  $k = 1$  to  $M$  do
2:   for  $i = 1$  to  $n$  do
3:      $u_i \leftarrow (b_i - \sum_{j \neq i}^n a_{ij}x_j) / a_{ii}$ ;
4:   end for
5:   if  $\|\mathbf{u} - \mathbf{x}\| < \varepsilon$  then
6:     break;
7:   else
8:     for  $i = 1$  to  $n$  do
9:        $x_i \leftarrow u_i$ ;
10:    end for
11:  end if
12: end for
```

Output:

(x_i)

Gauss-Seidel 迭代



- 基本思想：使用最新计算出的分量进行迭代

$$\left\{ \begin{array}{l} x_1^{(k+1)} = \frac{-1}{a_{11}} (a_{12}x_2^{(k)} + \dots + a_{1n}x_n^{(k)} - b_1) \\ x_2^{(k+1)} = \frac{-1}{a_{22}} (a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \dots + a_{2n}x_n^{(k)} - b_2) \\ \vdots \\ x_n^{(k+1)} = \frac{-1}{a_{nn}} (a_{n1}x_1^{(k+1)} + \dots + a_{n,n-1}x_{n-1}^{(k+1)} - b_n) \end{array} \right.$$

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

Gauss-Seidel 迭代



■ Gauss-Seidel 迭代矩阵:

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U},$$

$$\mathbf{D} = \begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix} \quad \mathbf{L} = \begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & & & \dots \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} 0 & a_{12} & \dots & \\ & 0 & \ddots & \\ & & \ddots & \\ & & & 0 & a_{n-1n} \\ & & & & 0 \end{pmatrix}$$

$$\mathbf{Ax} = \mathbf{b} \Leftrightarrow (\mathbf{D} + \mathbf{L} + \mathbf{U})\mathbf{x} = \mathbf{b}$$

$$\Leftrightarrow (\mathbf{D} + \mathbf{L})\mathbf{x} = -\mathbf{U}\mathbf{x} + \mathbf{b}$$

$$\Leftrightarrow \mathbf{x} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}\mathbf{x} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b}$$

$$\Rightarrow \mathbf{M} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}, \quad \mathbf{g} = (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b}$$

Gauss-Seidel 迭代

- Gauss-Seidel 迭代收敛条件的充分必要条件:

$$\rho(\mathbf{M}) < 1$$

- 定理: 若线性方程组 $\mathbf{Ax} = \mathbf{b}$ 的系数矩阵 \mathbf{A} 满足下列条件之一:

(1) \mathbf{A} 为行严格对角占优阵, 即 $|a_{ii}| > \sum_{j \neq i} |a_{ij}|, i = 1, 2, \dots$

(2) \mathbf{A} 为列严格对角占优阵, 即 $|a_{jj}| > \sum_{i \neq j} |a_{ij}|, j = 1, 2, \dots$

(3) \mathbf{A} 为正定对称矩阵, 即 $\mathbf{A} \begin{pmatrix} 12 \dots & & \\ & \ddots & \\ 12 \dots & & \end{pmatrix} k = 1, 2, \dots$

则 Gauss-Seidel 迭代收敛

- 对于行/列严格对角占优阵, 可以证明:

$$\|(\mathbf{D} + \mathbf{L})^{-1} \mathbf{U}\|_{\infty} \leq \|\mathbf{I} - \mathbf{D}^{-1} \mathbf{A}\|_{\infty}$$

故 Gauss-Seidel 迭代法常快于 Jacobi 迭代

Gauss-Seidel迭代



- 对于一般的线性方程组，Jacobi迭代与Gauss-Seidel迭代可能都收敛，也可能都不收敛，还可能Jacobi迭代收敛而Gauss-Seidel迭代不收敛，或者Gauss-Seidel迭代收敛而Jacobi迭代不收敛。
- 在两者都收敛的情况下，收敛的快慢也不一样
- 实际的经验：一般情况下，Gauss-Seidel迭代快于Jacobi迭代

Gauss-Seidel 迭代



■ Gauss-Seidel 算法

Algorithm 16 Gauss-Seidel's Iteration Algorithm

Input:

$n, (a_{ij}), (b_i), (x_i), M, \varepsilon$

```
1: for  $k = 1$  to  $M$  do
2:   for  $i = 1$  to  $n$  do
3:      $u_i \leftarrow x_i$ ;
4:   end for
5:   for  $i = 1$  to  $n$  do
6:      $x_i \leftarrow (b_i - \sum_{j \neq i}^n a_{ij}x_j)/a_{ii}$ ;
7:   end for
8:   if  $\|u - x\| < \varepsilon$  then
9:     break;
10:  end if
11: end for
```

Output:

(x_i)

■ 逐次超松弛迭代 (Successive Over Relaxation)

- 基本思想：当 $\rho(-(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}) \approx 1$ 时，收敛将会很慢，取 $\mathbf{x}^{(k+1)}$ 和 $\mathbf{x}^{(k)}$ 的一个适当的加权平均来改进 Gauss-Seidel 迭代

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega \Delta \mathbf{x}^{(k)}$$

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

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

$$\left\{ \begin{array}{l} x_1^{(k+1)} = (1-\omega)x_1^{(k)} + \omega \frac{-1}{a_{11}} (a_{12}x_2^{(k)} + \dots \quad (k) - b_1) \\ x_2^{(k+1)} = (1-\omega)x_2^{(k)} + \omega \frac{-1}{a_{22}} (a_{21}\boxed{x_1^{(k+1)}} + a_{23}x_3^{(k)} + \dots \quad (k) - b_2) \\ \vdots \\ x_n^{(k+1)} = (1-\omega)x_n^{(k)} + \omega \frac{-1}{a_{nn}} (a_{n1}\boxed{x_1^{(k+1)}} + \dots \quad \boxed{x_{n-1}^{(k+1)}} - b_n) \end{array} \right.$$

SOR迭代



■ SOR迭代矩阵:

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U},$$

$$\mathbf{D} = \begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 0 & & & 0 \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & & \dots & 0 \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} 0 & a_{12} & \dots & l_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & \\ & & & 0 & a_{n-1n} \\ 0 & & & & 0 \end{pmatrix}$$

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

$$\Rightarrow (\mathbf{D} + \omega\mathbf{L})\mathbf{x}^{(k+1)} = [(1 - \omega)\mathbf{D} - \omega\mathbf{U}]\mathbf{x}^{(k)} + \omega\mathbf{b}$$

$$\Rightarrow \mathbf{x}^{(k+1)} = (\mathbf{D} + \omega\mathbf{L})^{-1}[(1 - \omega)\mathbf{D} - \omega\mathbf{U}]\mathbf{x}^{(k)} + \omega(\mathbf{D} + \omega\mathbf{L})^{-1}\mathbf{b}$$

$$\Rightarrow \mathbf{M} = (\mathbf{D} + \omega\mathbf{L})^{-1}[(1 - \omega)\mathbf{D} - \omega\mathbf{U}], \quad \mathbf{g} = \omega(\mathbf{D} + \omega\mathbf{L})^{-1}\mathbf{b}$$

- SOR迭代收敛条件的充分必要条件：

$$\rho(M) < 1$$

- 定理：SOR迭代收敛的必要条件： $0 < \omega < 2$

- 定理：若A为对称正定矩阵，则当 $0 < \omega < 2$ 时，SOR迭代恒收敛

- 如何确定 ω 使得SOR迭代最优？对于一般的线性方程组，比较困难；对于特殊的线性方程组，譬如相容次序的矩阵，有一些理论结果

- SOR迭代

- 亚松弛迭代： $0 < \omega < 1$
- Gauss-Seidel迭代： $\omega = 1$
- 超松弛迭代： $1 < \omega < 2$

■ SOR迭代算法

Algorithm 17 SOR Iteration Algorithm

Input:

$n, (a_{ij}), (b_i), (x_i), M, \varepsilon$

1: **for** $k = 1$ to M **do**

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

3: $u_i \leftarrow x_i$;

4: **end for**

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

6: $x_i \leftarrow (1 - \omega) * u_i + \omega * (b_i - \sum_{j \neq i}^n a_{ij} x_j) / a_{ii}$;

7: **end for**

8: **if** $\|u - x\| < \varepsilon$ **then**

9: **break**;

10: **end if**

11: **end for**

Output:

(x_i)

逆矩阵的计算

- 线性代数中，若 $n \times n$ 阶矩阵 A 的 $\det(A) \neq 0$ ，则矩阵 A 可逆，且 $A^{-1} = A^* / \det(A)$
- 要计算 A^* 需要计算 n^2 个 $n-1$ 阶行列式，计算量很大
- 逆矩阵的计算等价于求解 n 个线性方程组：

$$Ax_j = e_j, j = 1, 2, \dots$$

$$A^{-1} = [x_1, x_2, \dots]$$

- 常用的方法：

- 利用 Gauss-Jordan 消元法： $(A, I) \rightarrow (I, A^{-1})$
- 利用 LU 分解方法
- 利用直接法或迭代法求解一系列线性方程组

为什么用迭代法？

- 设 n 为一偶数，考虑 $n \times n$ 阶矩阵 A ，其对角元为 3，超对角元和次对角元为 -1 ，对除 $i = n/2$ 及 $n/2 + 1$ 外所有的 $(i, n+1-i)$, $i = 1, 2, \dots$ 位置上的元素为 $1/2$ ，其它为零

线性方程组：

$$Ax = b, \quad b = (5/2, 3/2, \dots \quad \dots$$

$$\Leftrightarrow x = (1, 1, \dots$$

$$n = 6, \quad A = \begin{bmatrix} 3 & -1 & & & & 1/2 \\ -1 & 3 & -1 & & & 1/2 \\ & -1 & 3 & -1 & & \\ & & -1 & 3 & -1 & \\ & & & 1/2 & -1 & 3 & -1 \\ 1/2 & & & & -1 & 3 \end{bmatrix}$$

为什么用迭代法？

- 当 $n=100,000$ ，使用double类型存储矩阵A，需要大约：
 $O(n^2) = 8 \times n^2 = 8 \times 10^{10} \approx 80G$ 内存空间，使用Gauss列主元消去法需要： $O(n^3) = n^3 / 3 = 10^{15} / 3$ 次浮点运算，而计算机时钟频率：4GHz，每秒浮点运算次数 (FLOPS) = 108.8GFLOPS，计算时间为： $10^{15} / (3 * 108.8 * 10^9) \approx 3063$ 秒
- 当 $n=100,000$ ，使用double类型及稀疏格式存储矩阵A，需要大约 $O(4n) = 8 \times 4 \times n = 32 \times 10^5 \approx 3M$ 内存空间，使用Jacobi迭代法每次迭代需要： $O(n^2) = n^2 = 10^{10}$ 浮点运算，计算时间为： $10^{10} / (3 * 108.8 * 10^9) \approx 0.03063$ 秒，假设需要100步迭代，总时间为：3.063 秒

■ Krylove子空间方法

- 求解大型系数线性方程组有效的方法
- 对称正定方程组
 - 共轭梯度法 (Conjugate Gradient Method)
 - 预条件共轭梯度法 (Preconditioned Conjugate Gradient Method)
- 对称非定方程组
 - MINRES
 - SYMMLQ
- 最小二乘逼近
 - LSQR
 - LSMR
- 一般稀疏线性方程组
 - GMRES
 - BiCG, CGS, BiCGStab, QMR
 - ...

■ 多重网格法 (Multigrid method)

- 为求解Poisson方程那样的偏微分方程发明的
- 粗网格、细网格交互迭代
- 一类有效加速收敛的技巧

■ 区域分解法 (Domain decomposition methods)

- 基本思想：将一个大问题拆成一些小问题，求解这些小问题，最后将这些小问题的解合成大问题的解
- 无交叠方法 (Nonoverlapping method)
- 交叠方法 (Overlapping method)

■ 分块运算

- 向量、矩阵的分块运算，细分为Level-1 (标量, 向量间的运算), level-2 (矩阵与向量间的运算), level-3 (矩阵与矩阵的运算)
- BLAS库 (Basic Linear Algebra Subprogrammings) : 基于硬件的加速

■ 并行矩阵计算