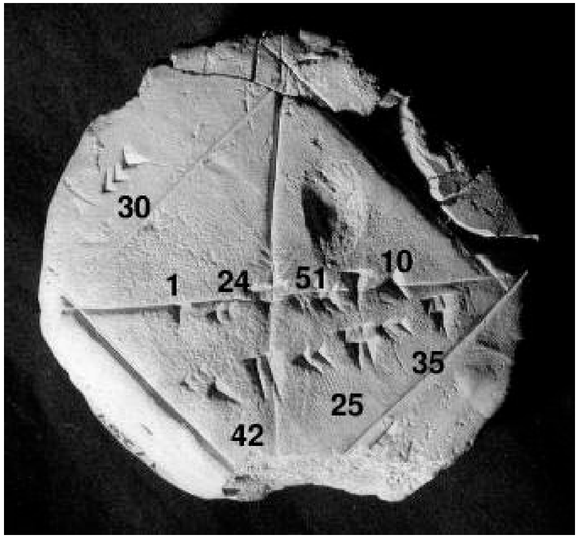
计算方法（A）



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

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 $\mathbf{Ax}=\mathbf{b}$等价变形为 $\mathbf{x}=\mathbf{Mx}+\mathbf{g}$，构造迭代关系式：$\mathbf{x}^{(k+1)}=\mathbf{Mx}^{(k)}+\mathbf{g}$。若向量序列 $\mathbf{x}^{(k)}$ 收敛到 $\mathbf{x}^*$，则

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

- 例如：

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

- 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：

$$\mathbf{x}^{(k+1)}-\mathbf{x}^*=\mathbf{M}\mathbf{x}^{(k)}-\mathbf{M}\mathbf{x}^*=\mathbf{M}(\mathbf{x}^{(k)}-\mathbf{x}^*)$$

$$=\cdots=\mathbf{M}^{k+1}(\mathbf{x}^{(0)}-\mathbf{x}^*)$$

- 向量序列 $\mathbf{x}^{(k)}$ 收敛 $\Leftrightarrow \lim\limits_{k\to\infty}\mathbf{M}^k=0$ 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

$$
\begin{cases}
a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\
\qquad\qquad \vdots \\
a_{n1}x_1 + \cdots + a_{nn}x_n = b_n
\end{cases}
$$

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

Jacobi迭代

- Jacobi迭代格式：

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

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

Jacobi迭代

- Jacobi迭代矩阵：

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{pmatrix} = \begin{pmatrix} 0 & \frac{-a_{12}}{a_{11}} & \cdots & \frac{-a_{1n}}{a_{11}} \\ \frac{-a_{21}}{a_{22}} & 0 & \cdots & \frac{-a_{2n}}{a_{22}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-a_{n1}}{a_{nn}} & \frac{-a_{n2}}{a_{nn}} & \cdots & 0 \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{pmatrix} + \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{pmatrix}, \ \mathbf{D} = \begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix}$$

$$\begin{aligned} \mathbf{Ax} = \mathbf{b} &\Leftrightarrow (\mathbf{D} - (\mathbf{D} - \mathbf{A}))\mathbf{x} = \mathbf{b} \\ &\Leftrightarrow \mathbf{Dx} = (\mathbf{D} - \mathbf{A})\mathbf{x} + \mathbf{b} \\ &\Leftrightarrow \mathbf{x} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b} \end{aligned}$$

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

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 $|a_{ii}|>\sum_{j\neq i}|a_{ij}|,\ i=1,2,\ldots,n.$

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

  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：

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

Jacobi迭代

- Jacobi迭代算法

Algorithm 15 Jacobi's Iteration Algorithm

Input:

n, (a_{ij}), (b_i), (x_i), M, ε

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

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

    $u_i \leftarrow (b_i - \sum_{j \neq i}^{n} a_{ij} x_j)/a_{ii}$;

  end for

  if $||\mathbf{u} - \mathbf{x}|| < \varepsilon$ **then**

    break;

  else

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

      $x_i \leftarrow u_i$;

    end for

  end if

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

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

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

Gauss-Seidel迭代

- **Gauss-Seidel迭代矩阵：**

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

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

$$\begin{aligned} \mathbf{Ax} = \mathbf{b} &\Leftrightarrow (\mathbf{D} + \mathbf{L} + \mathbf{U})\mathbf{x} = \mathbf{b} \\ &\Leftrightarrow (\mathbf{D} + \mathbf{L})\mathbf{x} = -\mathbf{Ux} + \mathbf{b} \\ &\Leftrightarrow \mathbf{x} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{Ux} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b} \end{aligned}$$

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

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 $|a_{ii}|>\sum_{j\neq i}|a_{ij}|,\ i=1,2,\ldots,n.$

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

  (3) $\mathbf{A}$为正定对称矩阵，即 $\left|\mathbf{A}\begin{pmatrix}12\cdots k\\12\cdots k\end{pmatrix}\right|>0,\ k=1,2,\ldots,n.$

  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, ε

for $k = 1$ to M **do**
    for $i = 1$ to n **do**
        $u_i \leftarrow x_i$;
    end for
    for $i = 1$ to n **do**
        $x_i \leftarrow (b_i - \sum_{j \neq i}^{n} a_{ij} x_j)/a_{ii}$;
    end for
    if $||\mathbf{u} - \mathbf{x}|| < \varepsilon$ **then**
        break;
    end if
end for

Output:

(x_i)

SOR迭代

- 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

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

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

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

$$\begin{cases} x_1^{(k+1)}=(1-\omega)x_1^{(k)}+\omega\dfrac{-1}{a_{11}}(a_{12}x_2^{(k)}+\cdots+a_{1n}x_n^{(k)}-b_1) \\ x_2^{(k+1)}=(1-\omega)x_2^{(k)}+\omega\dfrac{-1}{a_{22}}(a_{21}\boxed{x_1^{(k+1)}}+a_{23}x_3^{(k)}+\cdots+a_{1n}x_n^{(k)}-b_2) \\ \quad\vdots \\ x_n^{(k+1)}=(1-\omega)x_n^{(k)}+\omega\dfrac{-1}{a_{nn}}(a_{n1}\boxed{x_1^{(k+1)}+\cdots+a_{n\,n-1}x_{n-1}^{(k+1)}}-b_n) \end{cases}$$

SOR迭代

- SOR迭代矩阵：

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

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

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

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

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

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

SOR迭代

- SOR迭代收敛条件的充分必要条件：

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

- 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

- SOR迭代算法

Algorithm 17 SOR Iteration Algorithm

Input:

n, (a_{ij}), (b_i), (x_i), M, ε

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

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

    $u_i \leftarrow x_i$;

  end for

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

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

  end for

  if $||\mathbf{u} - \mathbf{x}|| < \varepsilon$ **then**

    break;

  end if

end for

Output:

(x_i)

逆矩阵的计算

- 线性代数中，若$n\times n$阶矩阵$\mathbf{A}$的$\det(\mathbf{A})\neq 0$，则矩阵$\mathbf{A}$可逆，且$\mathbf{A}^{-1}=\mathbf{A}^{*}/\det(\mathbf{A})$
- 要计算$\mathbf{A}^{*}$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：

$$\mathbf{A}\mathbf{x}_j=\mathbf{e}_j,\ j=1,2,\ldots,n$$

$$\mathbf{A}^{-1}=[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n]$$

- 常用的方法：
  - 利用Gauss-Jordan消元法：$(\mathbf{A},\mathbf{I})\to(\mathbf{I},\mathbf{A}^{-1})$
  - 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$(i,n+1-i),\ i=1,2,\ldots,n$位置上的元素为$1/2$，其它为零线性方程组：

$$\mathbf{Ax}=\mathbf{b},\ \mathbf{b}=(5/2,3/2,\ldots 3/2,1,1,3/2,\ldots,3/2,5/2)^T$$

$$\Leftrightarrow \mathbf{x}=(1,1,\ldots,1)^T$$

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

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

迭代法

- 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
    - …

迭代法

- 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