高等数值算法与应用（六）

Advanced Numerical Algorithms & Applications

计算机科学与技术系 喻文健

内容概要

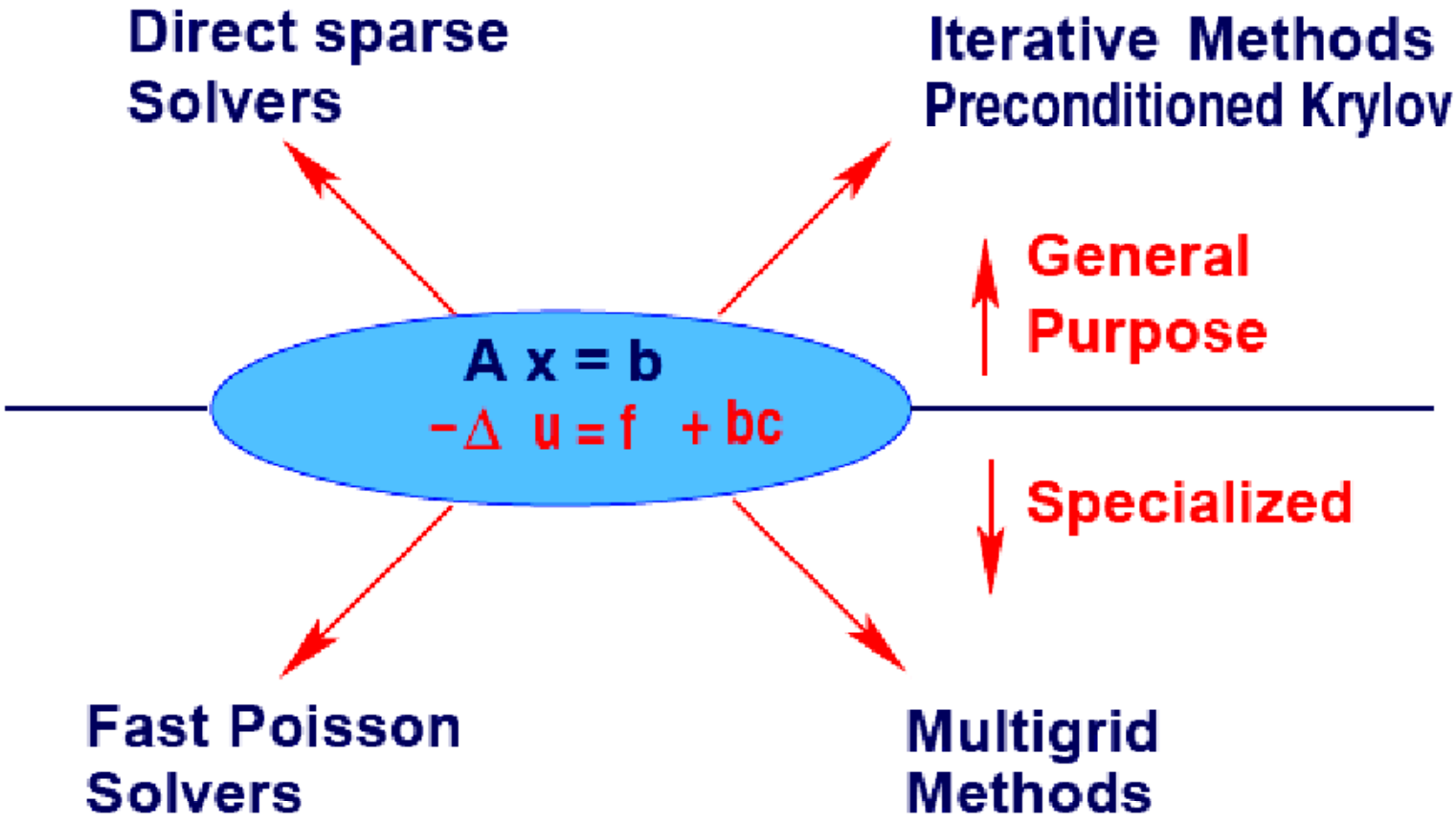
- 线性方程组的迭代解法
  - 概述与经典迭代解法回顾
  - 变分原理导出的非经典迭代解法
    - 最速下降法
    - 共扼梯度(CG)算法及预条件技术
  - **Krylov**子空间迭代法
    - 投影方法的原理
    - **FOM**算法
    - **GMRES**算法

课本第**6.5.2, 6.5.6**小节，第**11.5**节**(**除**11.5.7)**，补充材料

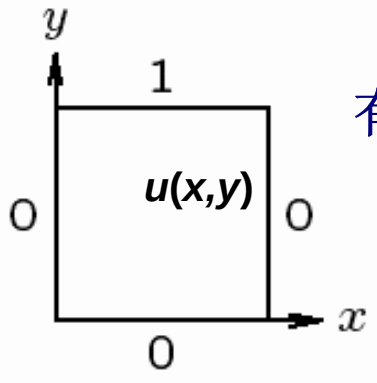
Overview and the Stationary Iterative Methods

Wenjian Yu

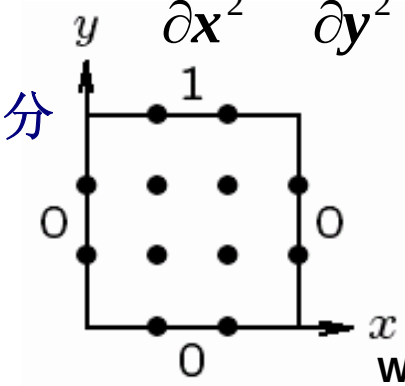
Solving Sparse Systems in 2010



注: $\Delta u \equiv \nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y)$ + 边界条件



有限差分



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

稀疏矩阵线性方程组

经典迭代法 (Stationary Iterative Methods)

- 基本的迭代公式
  - 构造收敛的近似解序列： $x^{(0)}, x^{(1)}, \cdots, x^{(k)}, \cdots$
  - 经典迭代法: 相邻的近似解之间满足某种固定的函数关系
  - 一阶定常迭代公式: $x^{(k+1)}=Bx^{(k)}+f$
  - 构造迭代公式的要求 $Ax=b \Leftrightarrow x=Bx+f$
  - 一阶定常迭代法的收敛性
    - 矩阵的谱半径(spectral radius) $\rho(A)=\max\{|\lambda|:\lambda\in\lambda(A)\}$
    - 定理: 迭代法 $x^{(k+1)}=Bx^{(k)}+f$ 对任意的初始解 $x^{(0)}$ 收敛的充要条件是 $\rho(B)<1$
  - 构造方法: 分裂(splitting)法

    设 $A=M-N$, 则 $Ax=b \Leftrightarrow x=M^{-1}Nx+M^{-1}b$

    构造迭代公式: $x^{(k+1)}=M^{-1}Nx^{(k)}+M^{-1}b$

经典迭代法 (Stationary Iterative Methods)

- **Jacobi迭代法**

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \Rightarrow \begin{cases} x_1^{(k+1)} = -\frac{1}{a_{11}}\left(\quad a_{12}x_2^{(k)} + a_{13}x_3^{(k)} \right) + \frac{b_1}{a_{11}} \\ x_2^{(k+1)} = -\frac{1}{a_{22}}\left(a_{21}x_1^{(k)} \quad + a_{23}x_3^{(k)} \right) + \frac{b_2}{a_{22}} \\ x_3^{(k+1)} = -\frac{1}{a_{33}}\left(a_{31}x_1^{(k)} + a_{32}x_2^{(k)} \quad \right) + \frac{b_3}{a_{33}} \end{cases}$$

  - 计算公式: $x_i^{(k+1)} = \left(b_i - \sum_{j \neq i} a_{ij}x_j^{(k)} \right) / a_{ii}, \quad i = 1, \ldots, n$
  - 矩阵形式:

    设 $A = D - L - U$, 则 $x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b$
  - 收敛的几个充要条件
    - 迭代矩阵的范数<1
    - A和2D–A都对称正定, 且A的对角元>0
    - A严格对角占优, 或不可约的(弱)对角占优

经典迭代法 (Stationary Iterative Methods)

Gauss-Seidel迭代法

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \Rightarrow \begin{cases} x_1^{(k+1)} = -\frac{1}{a_{11}}\left(\quad a_{12}x_2^{(k)} + a_{13}x_3^{(k)} \right) + \frac{b_1}{a_{11}} \\ x_2^{(k+1)} = -\frac{1}{a_{22}}\left(a_{21}x_1^{(k+1)} \quad + a_{23}x_3^{(k)} \right) + \frac{b_2}{a_{22}} \\ x_3^{(k+1)} = -\frac{1}{a_{33}}\left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} \quad \right) + \frac{b_3}{a_{33}} \end{cases}$$

- 公式: 依次计算 $x_i^{(k+1)} = \left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)} \right) / a_{ii}, i = 1, \ldots, n$

解分量按从1到n的顺序更新, 向后G-S迭代

- 矩阵形式:

  设 $A = D - L - U$, 则 $x^{(k+1)} = (D - L)^{-1}Ux^{(k)} + (D - L)^{-1}b$

- 收敛的几个充要条件
  - 矩阵$D^{-1}(L+U)$的1或∞范数<1
  - A严格对角占优, 或不可约的(弱)对角占优
  - A对称正定

经典迭代法 (Stationary Iterative Methods)

SOR迭代法

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \Rightarrow \begin{cases} x_1^{(k+1)} = (1-\omega)x_1^{(k)} + \omega\left(-\frac{a_{12}}{a_{11}}x_2^{(k)} - \frac{a_{13}}{a_{11}}x_3^{(k)} + \frac{b_1}{a_{11}}\right) \\ x_2^{(k+1)} = (1-\omega)x_2^{(k)} + \omega\left(-\frac{a_{21}}{a_{22}}x_1^{(k+1)} - \frac{a_{23}}{a_{22}}x_3^{(k)} + \frac{b_2}{a_{22}}\right) \\ x_3^{(k+1)} = (1-\omega)x_3^{(k)} + \omega\left(-\frac{a_{31}}{a_{33}}x_1^{(k+1)} - \frac{a_{32}}{a_{33}}x_2^{(k+1)} + \frac{b_3}{a_{33}}\right) \end{cases}$$

- G-S法的扩展

$$\begin{cases} \tilde{x}_i^{(k+1)} = (b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k+1)})/a_{ii}, i = 1, \ldots, n \\ x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\tilde{x}_i^{(k+1)} \end{cases}$$

向前SOR, 向后SOR

- 迭代矩阵:

  设 $A = D - L - U$, 则 $B = (D - \omega L)^{-1}\left[(1-\omega)D + \omega U\right]$

- 收敛的几个充要条件
  - ♦必要条件: 0<ω<2 ; ω =1即G-S迭代法
  - ♦ A严格对角占优, 或不可约的(弱)对角占优, 且0<ω≤1
  - ♦ A对称正定, 且0<ω<2

如何选ω的值?

Steepest Descent Method

Wenjian Yu

- 本节主要内容
  - “变分原理”
  - 无约束优化问题——求最小值
  - 最速下降法**(Steepest Descent)**

解线性方程组的变分原理

- 实多元二次函数 $f(x)=\frac{1}{2}x^TAx-b^Tx$，其中$A$对称方阵
  - 取极值的必要条件 $\nabla f(x)=Ax-b=0$
  - 充分性：若A对称正定，取最小值的充分性得到保证
- 充分性的证明

  若x 满足$Ax-b=0$，　$f(y)=f(x)+\frac{1}{2}(y-x)^TA(y-x)$

  由于 $\frac{1}{2}(y-x)^TA(y-x)>0$

  所以，$f(y)>f(x)$ ，$\forall y\neq x$

- 变分原理：若A为实对称正定矩阵，则
  - 求解方程 $Ax=b$ 等价于求 $f(x)=\frac{1}{2}x^TAx-b^Tx$最小值对应的 x

Steepest Descent in 无约束优化问题：求 $f(x)$的最小值

二次连续可微

- Here's one way to minimize a twice continuously differentiable function of several variables $f : \mathbb{R}^n \to \mathbb{R}$
- Pick a starting point x_0
- At each $x^{(k)}$ compute the gradient $\nabla f(x_k)$
- The search direction will be $p_k = -\nabla f(x_k)$
- Find the step length α_k, ie. how far to go along p_k, by minimizing $g_{k+1}(\alpha) := f(x_k + \alpha p_k)$

单变量函数求最小值

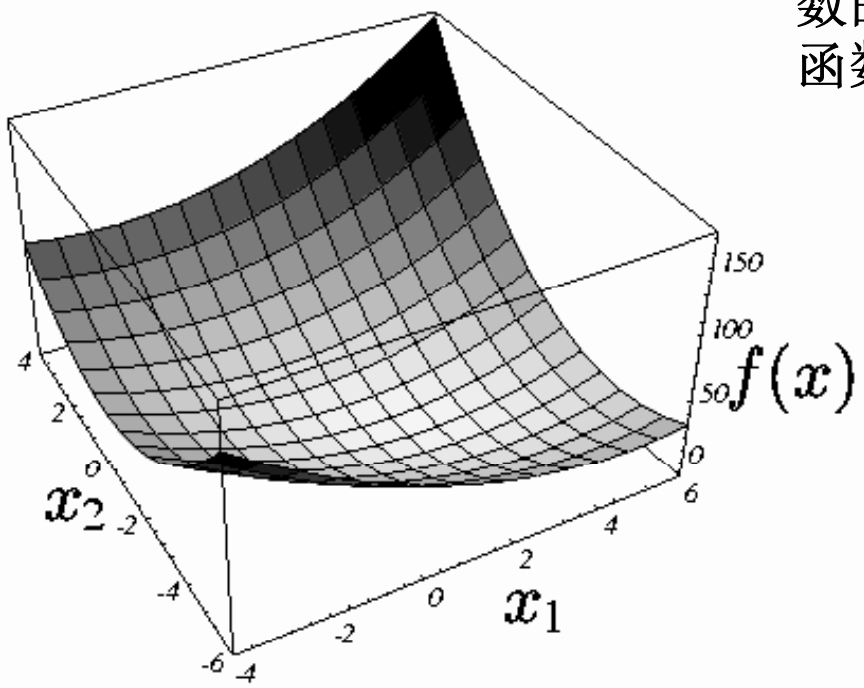
- Set $x_{k+1} = x_k + \alpha_{k+1} p_k$
- With the appropriate conditions (e.g. f is a convex function or x_0 close to the minimizer x), can show that $\lim_{k \to \infty} x_k = x$

为了表达的简便，用 x_k 表示迭代解向量

For symmetric positive definite matrix A

$$f(x) = \frac{3}{2}x_1^2 + 2x_1x_2 + 3x_2^2 - 2x_1 + 8x_2$$

梯度方向 $\nabla f(x)$ 上函数的方向导数最大，即函数值增长速度最快

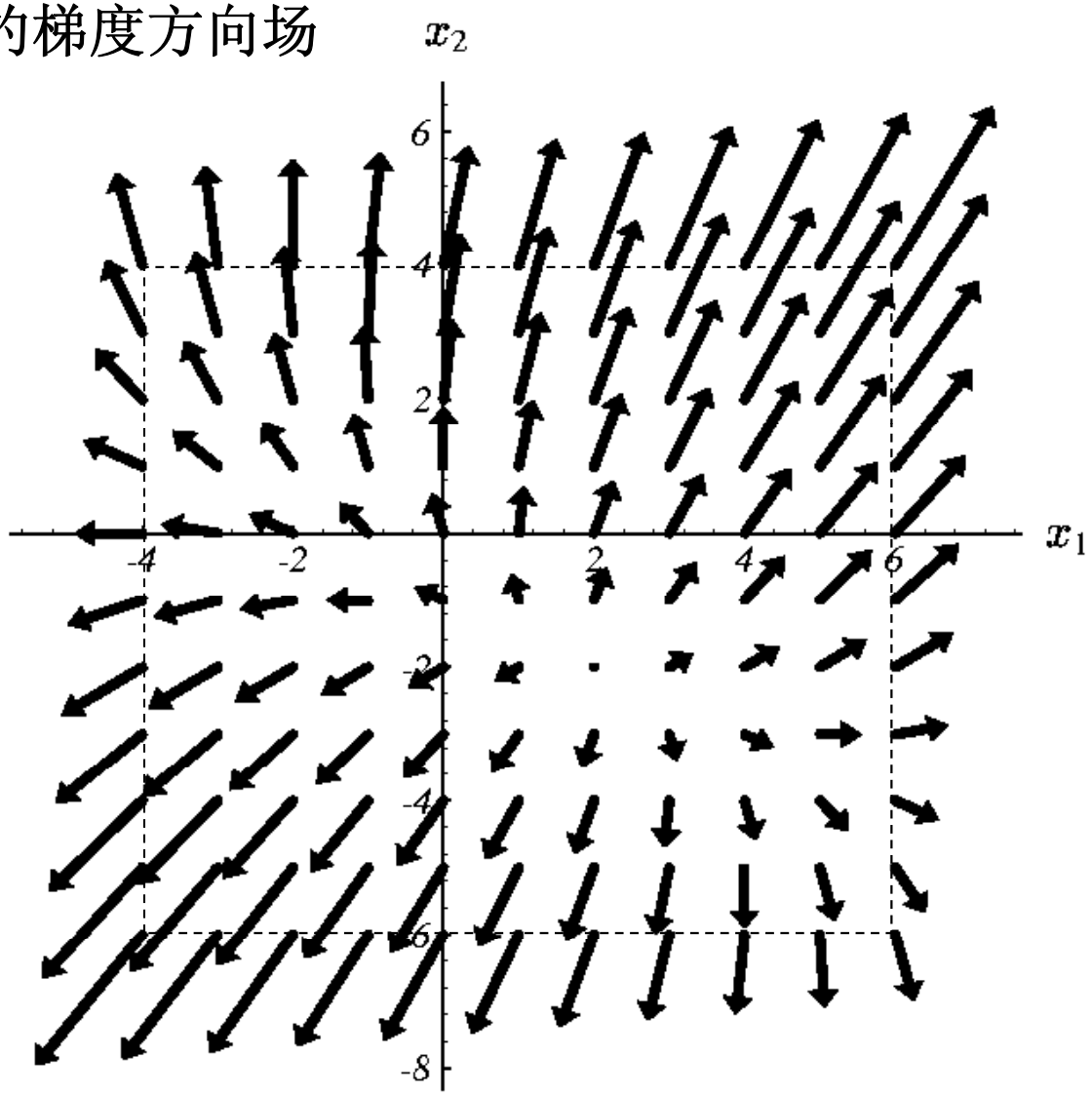


Graph of quadratic form $f(x) = \frac{1}{2}x^T Ax - b^T x$ The minimum point of this surface is the solution to $Ax = b$.

from J. R. Shewchuk "painless CG"

Gradient of quadratic form

前一页例子的梯度方向场



The points in the direction of steepest increase of f(x)

from J. R. Shewchuk "painless CG"

Iterations of Steepest Descent Method

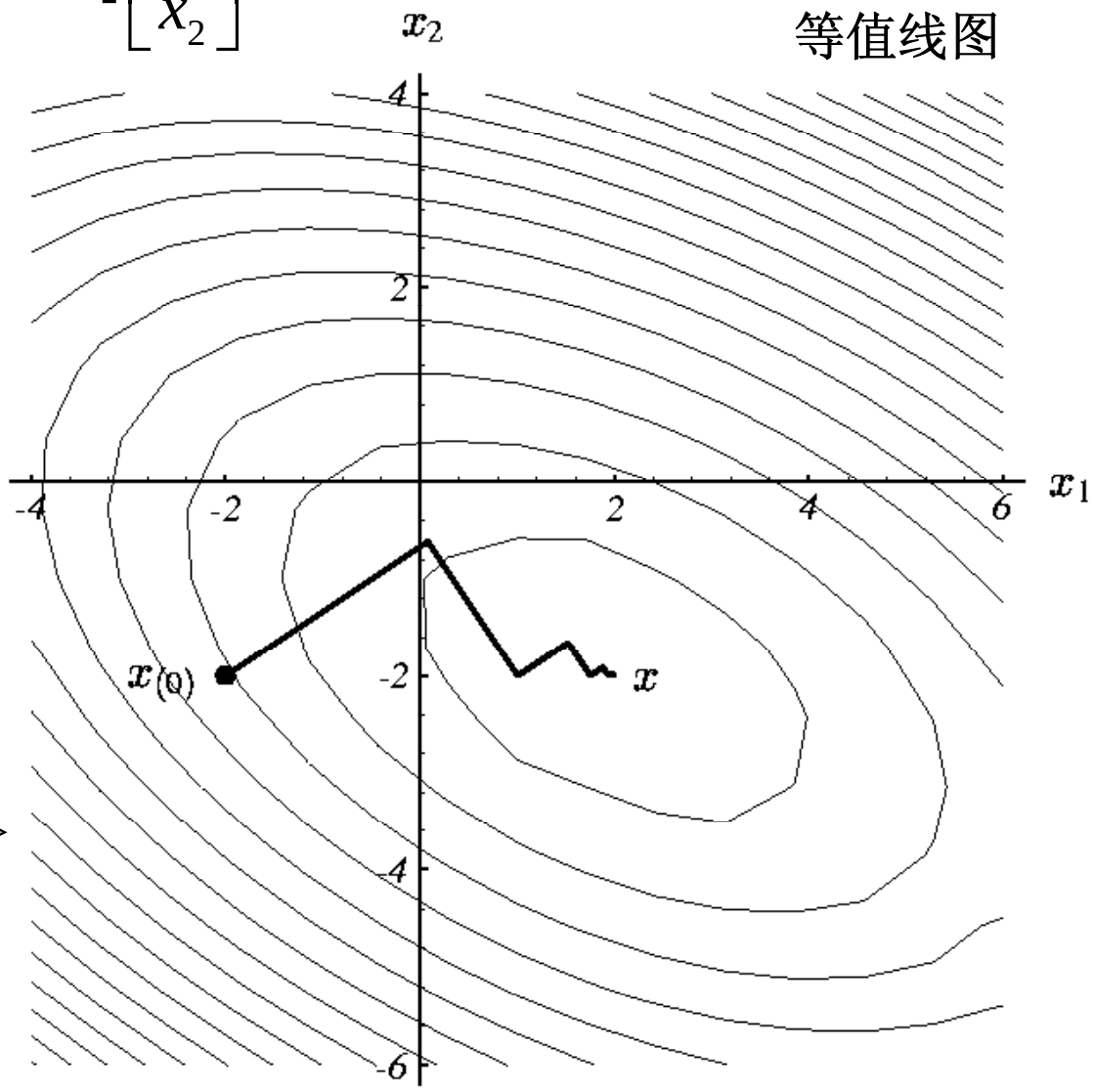
$$f(x)=\frac{1}{2}\begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}3 & 2\\ 2 & 6\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}-\begin{bmatrix}2 & -8\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

$$=\frac{3}{2}x_1^2+2x_1x_2+3x_2^2-2x_1+8x_2$$

求$f(x)$最小值等价于求解

$$\begin{bmatrix}3 & 2\\ 2 & 6\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}2\\ -8\end{bmatrix}$$

设$x_{(0)}=[-2,-2]^T$,最速下降法搜索最小值



from J. R. Shewchuk "painless CG"

Steepest Descent for Solving *Ax = b*

- If $f(x)=\frac{1}{2}x^TAx-b^Tx$, then

负梯度方向即残差向量

$$p_k=-\nabla f(x_k)=b-Ax_k=r_k$$

- Note that the search direction equals the residual
- So $g_{k+1}(\alpha)=f(x_k+\alpha r_k)=\frac{1}{2}(x_k+\alpha r_k)^TA(x_k+\alpha r_k)-b^T(x_k+\alpha r_k)$

求α，使$g_{k+1}(\alpha)$达最小值

一元二次函数

- Differentiating once, we get $g'_{k+1}(\alpha)=\alpha r_k^TAr_k-r_k^Tr_k$
- Differentiating again, we get $g''_{k+1}(\alpha)=r_k^TAr_k>0$
- So the minimum is attained at $g'_{k+1}(\alpha_{k+1})=0$, ie.

$$\alpha_{k+1}=\frac{r_k^Tr_k}{r_k^TAr_k}$$

确定了搜索方向上的步长

Steepest Descent Algorithm

- Note again that $p_k = r_k$

  Input: $x_0 = 0, r_0 = b$

  Output: approximate solution x_N when r_N small enough

  for $k = 1, 2, 3, \ldots$

$$\begin{cases} \alpha_k = \dfrac{r_{k-1}^T r_{k-1}}{r_{k-1}^T A r_{k-1}} & \text{(step length)} \\ x_k = x_{k-1} + \alpha_k r_{k-1} & \text{(approximate solution)} \\ r_k = r_{k-1} - \alpha_k A r_{k-1} & \text{(residual)} \end{cases}$$

- Computational cost:
  - One matrix-vector multiplication per iteration
  - Two inner products per iteration
- Storage:
  - Three vectors of working storage
  - Static storage for ***A*** (ie. no need to modify ***A***)

循环次数=搜索步数，是影响计算量的主要因素

Steepest Descent Algorithm

- 收敛性理论
  - 定理：A为SPD矩阵，其最大、最小特征值为λ_1, λ_n，则最速下降法有：

$$\left\| x_k - x^* \right\|_A \le \left(\frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} \right)^k \left\| x_0 - x^* \right\|_A$$

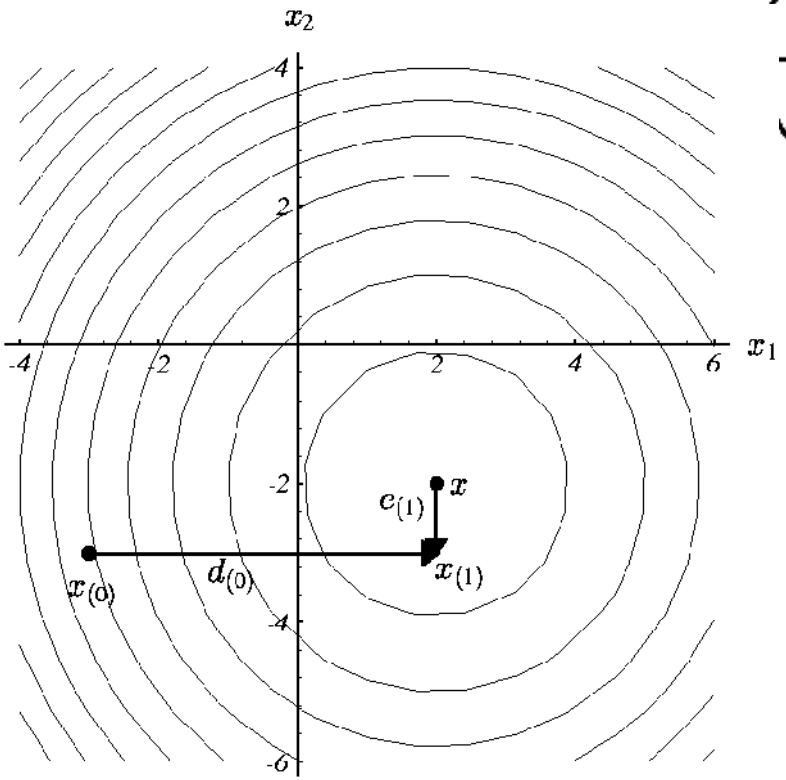
  - 在A范数度量下的收敛因子为 $\frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n}$，可能收敛很慢
- 缺点
  - 每步搜索是局部最优，但整体上可能反复搜索相同方向（迭代收敛慢）
  - 虽然r_k是x_k处的最速下降方向，但当r_k很小时，由于舍入误差，其计算值会严重偏离最速下降方向

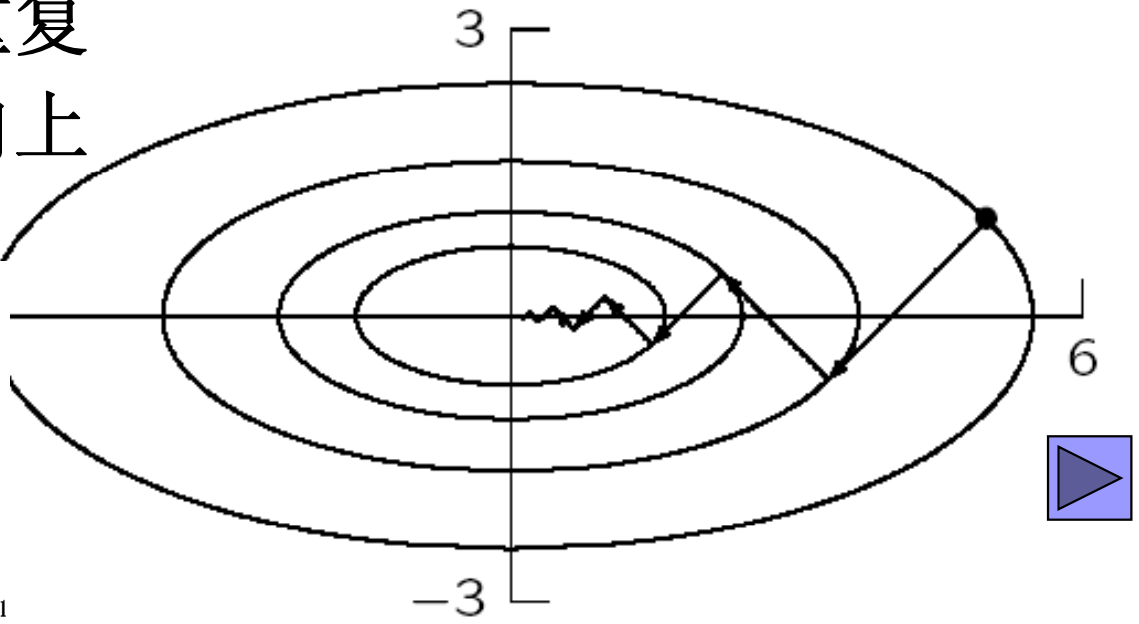
Example using steepest descent method

$$f(x) = 0.5x_1^2 + 2.5x_2^2 = x^T \begin{bmatrix} 0.5 & 0 \\ 0 & 2.5 \end{bmatrix} x$$

初始值 $x_{(0)} = [5,1]^T$

- 整体上，搜索方向重复
- 能否在每个正交方向上仅搜索一次?





从图中观察到：
新解处的梯度(残差)方向与前一搜索方向垂直。这一结论具有普遍性!

Conjugate Gradient Algorithm

Wenjian Yu

- 本节主要内容
  - 共轭梯度法**(Conjugate Gradient)**
  - 迭代法收敛速度与方法比较
  - 采用预条件技术**(Preconditioning)**的**CG**法

共轭梯度法—1

- 如何用较少的步数搜索到$f(x)=\frac{1}{2}x^TAx-b^Tx$的最小值?
- 沿方向d_i的直线搜索----使$f(x_{i+1})$达局部最小值

$$x_{i+1}=x_i+a_id_i$$

$$f(x_{i+1})=\frac{1}{2}(x_i+a_id_i)^TA(x_i+a_id_i)-b^T(x_i+a_id_i) \Rightarrow a_i=\frac{r_i^Td_i}{d_i^TAd_i}$$

  - 新解x_{i+1}对应的残差向量与搜索方向d_i垂直

$$r_{i+1}=r_i-Aa_id_i=r_i-\frac{r_i^Td_i}{d_i^TAd_i}Ad_i \Rightarrow d_i^Tr_{i+1}=0$$

由于r_{i+1}为x_{i+1}处梯度的反方向，所以新解处梯度方向垂直于搜索方向

- 能否沿一组正交基方向搜索?
  - 对n元函数求最小值，最多只需n步搜索

$\{d_0,d_1,\cdots,d_{n-1}\}$ 为一组正交基，将初始误差投影到这组基上

$$e_0=x_0-x^*=\sum_{j=0}^{n-1}\delta_jd_j$$

共轭梯度法—2

- 能否沿一组正交基方向搜索?

$$\boldsymbol{e}_0 = \boldsymbol{x}_0 - \boldsymbol{x}^* = \sum_{j=0}^{n-1} \delta_j \boldsymbol{d}_j \qquad \{\boldsymbol{d}_0, \boldsymbol{d}_1, \cdots, \boldsymbol{d}_{n-1}\} \text{ 为一组正交基} \qquad \boxed{\delta_i = \frac{\boldsymbol{d}_i^T \boldsymbol{e}_0}{\boldsymbol{d}_i^T \boldsymbol{d}_i}}$$

若 $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \delta_i \boldsymbol{d}_i$ ⟹ $\boldsymbol{e}_i = \boldsymbol{x}_i - \boldsymbol{x}^* = \sum_{j=i}^{n-1} \delta_j \boldsymbol{d}_j$, 另一计算公式 $\delta_i = \dfrac{\boldsymbol{d}_i^T \boldsymbol{e}_i}{\boldsymbol{d}_i^T \boldsymbol{d}_i}$

  - 该算法无法实现的原因：$\boldsymbol{e}_i$未知！

- 共轭正交方向搜索
  - $\{\boldsymbol{d}_0, \boldsymbol{d}_1, \cdots, \boldsymbol{d}_{n-1}\}$ 未必要正交，只要线性无关
  - 选择一组“共轭正交”的基向量

初始误差表示为这种基的线性组合，其系数为搜索因子

$$\boldsymbol{e}_0 = \sum_{j=0}^{n-1} \delta_j \boldsymbol{d}_j \text{ , 使 } \alpha_i = -\delta_i, \text{ 则 } \boldsymbol{e}_i = \sum_{j=i}^{n-1} \delta_j \boldsymbol{d}_j, \qquad (\boldsymbol{d}_i, \boldsymbol{d}_j)_A : \begin{cases} = 0, & i \neq j \\ > 0, & i = j \end{cases}$$

线性无关?

$$\alpha_i = -\frac{(\boldsymbol{d}_i, \boldsymbol{e}_i)_A}{(\boldsymbol{d}_i, \boldsymbol{d}_i)_A} = -\frac{\boldsymbol{d}_i^T \boldsymbol{A} \boldsymbol{e}_i}{\boldsymbol{d}_i^T \boldsymbol{A} \boldsymbol{d}_i} = \frac{\boldsymbol{d}_i^T \boldsymbol{r}_i}{\boldsymbol{d}_i^T \boldsymbol{A} \boldsymbol{d}_i}$$

α_i 可计算！符合局部最优的直线搜索

共轭梯度法—3

- 共轭正交方向搜索----四条性质
  - 1.此方法是可行的，搜索n步即达到准确解
  - 2.沿方向$\boldsymbol{d}_i$的搜索步长为 $\alpha_i = \dfrac{\boldsymbol{d}_i^T \boldsymbol{r}_i}{\boldsymbol{d}_i^T \boldsymbol{A}\boldsymbol{d}_i}$,它使此步达局部最小
  - 3. $\boldsymbol{x}_{i+1}$是$\boldsymbol{x}_0$ +span$\{\boldsymbol{d}_0, \boldsymbol{d}_1, \cdots, \boldsymbol{d}_i\}$集合中使$\|\boldsymbol{e}\|_A$达最小的向量

$$\|\boldsymbol{e}_{i+1}\|_A^2 = \boldsymbol{e}_{i+1}^T \boldsymbol{A}\boldsymbol{e}_{i+1} = \left(\sum_{j=i+1}^{n-1} \delta_j \boldsymbol{d}_j\right)^T \boldsymbol{A}\left(\sum_{j=i+1}^{n-1} \delta_j \boldsymbol{d}_j\right) = \sum_{j=i+1}^{n-1}\sum_{k=i+1}^{n-1} \delta_j \delta_k \boldsymbol{d}_j^T \boldsymbol{A}\boldsymbol{d}_k = \sum_{j=i+1}^{n-1} \delta_j^2 \boldsymbol{d}_j^T \boldsymbol{A}\boldsymbol{d}_j$$

$\forall \boldsymbol{x} \in \boldsymbol{x}_0$ +span$\{\boldsymbol{d}_0, \cdots, \boldsymbol{d}_i\}$
考虑其误差

$$\|\boldsymbol{e}\|_A^2 = \left\|\sum_{j=0}^{i} \xi_j \boldsymbol{d}_j + \sum_{j=i+1}^{n-1} \delta_j \boldsymbol{d}_j\right\|_A^2 = \sum_{j=0}^{i} \xi_j^2 \boldsymbol{d}_j^T \boldsymbol{A}\boldsymbol{d}_j + \sum_{j=i+1}^{n-1} \delta_j^2 \boldsymbol{d}_j^T \boldsymbol{A}\boldsymbol{d}_j \geq \|\boldsymbol{e}_{i+1}\|_A^2$$

$$\boldsymbol{e} = \boldsymbol{x}_0 + \sum_{j=0}^{i} \xi_j' \boldsymbol{d}_j - \boldsymbol{x}^* = \sum_{j=0}^{n-1} \delta_j \boldsymbol{d}_j + \sum_{j=0}^{i} \xi_j' \boldsymbol{d}_j$$

  - 4. $\boldsymbol{r}_i^T \boldsymbol{d}_j = 0,\ i > j$, 即$\boldsymbol{r}_i$垂直于前面所有搜索方向(性质2: $\boldsymbol{r}_i \perp \boldsymbol{d}_{i-1}$)

根据 $\boldsymbol{e}_i = \sum_{j=i}^{n-1} \delta_j \boldsymbol{d}_j$, 等号两边左乘 $\boldsymbol{d}_k^T \boldsymbol{A}$, ***k<i*** ⇒ $\boldsymbol{d}_k^T \boldsymbol{A}\boldsymbol{e}_i = 0$

将***k***换成***j***，即得 $\boldsymbol{r}_i^T \boldsymbol{d}_j = 0,\ j < i$ ⇐ $\boldsymbol{d}_k^T \boldsymbol{r}_i = 0$

共轭梯度法—4

- 共轭正交方向的构造
  - 根据一组线性无关向量$\{\boldsymbol{u}_0,\cdots,\boldsymbol{u}_{n-1}\}$构造**A**正交的$\{\boldsymbol{d}_0,\boldsymbol{d}_1,\cdots,\boldsymbol{d}_{n-1}\}$

$$\boldsymbol{d}_0=\boldsymbol{u}_0$$

$$\boldsymbol{d}_i=\boldsymbol{u}_i+\sum_{j=0}^{i-1}\beta_{ij}\boldsymbol{d}_j\ ,\quad i=1,\cdots,n-1$$

类似于**Gram-Schmidt**正交化过程

$$\boldsymbol{d}_j^T\boldsymbol{A}\boldsymbol{d}_i=\boldsymbol{d}_j^T\boldsymbol{A}\boldsymbol{u}_i+\beta_{ij}\boldsymbol{d}_j^T\boldsymbol{A}\boldsymbol{d}_j=0,\quad j<i$$

$$\beta_{ij}=\frac{-\boldsymbol{d}_j^T\boldsymbol{A}\boldsymbol{u}_i}{\boldsymbol{d}_j^T\boldsymbol{A}\boldsymbol{d}_j}\ ,\quad j<i$$

存在的问题：
在迭代过程中需存储每个方向向量；
每一步的计算量较大

  - 性质**4**推论：$\boldsymbol{r}_i^T\boldsymbol{u}_j=0,\ i>j$ ⟸ **4.**$\boldsymbol{r}_i^T\boldsymbol{d}_j=0,\ i>j$, 即$\boldsymbol{r}_i$垂直于前面所有搜索方向

共轭梯度法—5

- 线性无关向量组怎么选取？怎么减小计算量？
  - 是否可取 $\boldsymbol{u}_i = \boldsymbol{r}_i$？
    - $\boldsymbol{r}_i$与前面所有的搜索方向都正交，则与它们线性无关(新的独立方向)
    - 根据性质4推论，$\boldsymbol{r}_i^T \boldsymbol{r}_j = 0,\ i > j$, 也说明了$\{\boldsymbol{u}_0, \cdots, \boldsymbol{u}_{n-1}\}$线性无关
    - 如果某个$\boldsymbol{r}_i$为零，则表明获得准确解，迭代过程可终止
  - 计算量会减小吗？其实 $\boldsymbol{d}_i = \boldsymbol{r}_i + \sum_{j=0}^{i-1} \beta_{ij} \boldsymbol{d}_j$ 中很多 β_{ij} 为0

$$\beta_{ij} = \frac{-\boldsymbol{d}_j^T \boldsymbol{A} \boldsymbol{r}_i}{\boldsymbol{d}_j^T \boldsymbol{A} \boldsymbol{d}_j},\quad j < i$$

主要看分子

$$\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \boldsymbol{A}\alpha_j \boldsymbol{d}_j$$

$$\boldsymbol{r}_i^T \boldsymbol{r}_{j+1} = \boldsymbol{r}_i^T \boldsymbol{r}_j - \alpha_j \boldsymbol{r}_i^T \boldsymbol{A} \boldsymbol{d}_j = \boldsymbol{r}_i^T \boldsymbol{r}_j - \alpha_j \boldsymbol{d}_j^T \boldsymbol{A} \boldsymbol{r}_i$$

若$j < i-1$, $\alpha_j \boldsymbol{d}_j^T \boldsymbol{A} \boldsymbol{r}_i = 0 \Rightarrow \beta_{ij} = 0$

若$j = i-1$, $\boldsymbol{r}_i^T \boldsymbol{r}_i = -\alpha_j \boldsymbol{d}_j^T \boldsymbol{A} \boldsymbol{r}_i \Rightarrow \boldsymbol{d}_j^T \boldsymbol{A} \boldsymbol{r}_i = -\boldsymbol{r}_i^T \boldsymbol{r}_i \cdot \frac{\boldsymbol{d}_j^T \boldsymbol{A} \boldsymbol{d}_j}{\boldsymbol{d}_j^T \boldsymbol{r}_j} \Rightarrow \beta_{ij} = \frac{\boldsymbol{r}_i^T \boldsymbol{r}_i}{\boldsymbol{d}_j^T \boldsymbol{r}_j} = \frac{\boldsymbol{r}_i^T \boldsymbol{r}_i}{\boldsymbol{r}_{i-1}^T \boldsymbol{r}_{i-1}}$

$$\alpha_j = \frac{\boldsymbol{d}_j^T \boldsymbol{r}_j}{\boldsymbol{d}_j^T \boldsymbol{A} \boldsymbol{d}_j}$$

$$\boldsymbol{d}_j^T \boldsymbol{r}_j = (\boldsymbol{r}_j + \sum_{k=0}^{j-1} \beta_{jk} \boldsymbol{d}_k)^T \boldsymbol{r}_j = \boldsymbol{r}_j^T \boldsymbol{r}_j$$

CG算法

为保持和课本一致，将前面推导中的d_k换成p_k

1) 选择x_0, $r_0=b-Ax_0$, 取$p_0=r_0$

2) 对k=0, 1, … 直到 $\|r_k\| \le \varepsilon$ (或 $\|r_k\|/\|b\| \le \varepsilon$)

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$$ 搜索方向的步长

$$x_{k+1} = x_k + \alpha_k p_k$$ 新迭代解

3: $$r_{k+1} = r_k - \alpha_k A p_k$$ 更新剩余向量

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k$$ 新的搜索方向

若让**k=n-1**时结束循环，变成直接解法（总计算量**~6**倍的**Cholesky**分解，且舍入误差使**r**的正交性丧失）

- 计算量
  - 每步迭代，一次矩阵向量乘，两次向量内积
- 存储量
  - **4**个向量工作空间 **(**第**3**行之前：**r, p, Ap, x**，之后：两个**r, p, x)**
  - 矩阵 **A**的静态存储 **(**计算过程中不修改**A)**

CG算法例子

$$f(x)=\frac{1}{2}\begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}3 & 2\\ 2 & 6\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}-\begin{bmatrix}2 & -8\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

$$=\frac{3}{2}x_1^2+2x_1x_2+3x_2^2-2x_1+8x_2$$

$$\boldsymbol{x}_{(0)}=[-2,-2]^T$$

$$\boldsymbol{p}_{(0)}=\boldsymbol{r}_{(0)}=[12,8]^T$$

$$\alpha_{(0)}=\frac{\boldsymbol{r}_{(0)}^T\boldsymbol{r}_{(0)}}{\boldsymbol{p}_{(0)}^T\boldsymbol{A}\boldsymbol{p}_{(0)}}=0.1733$$

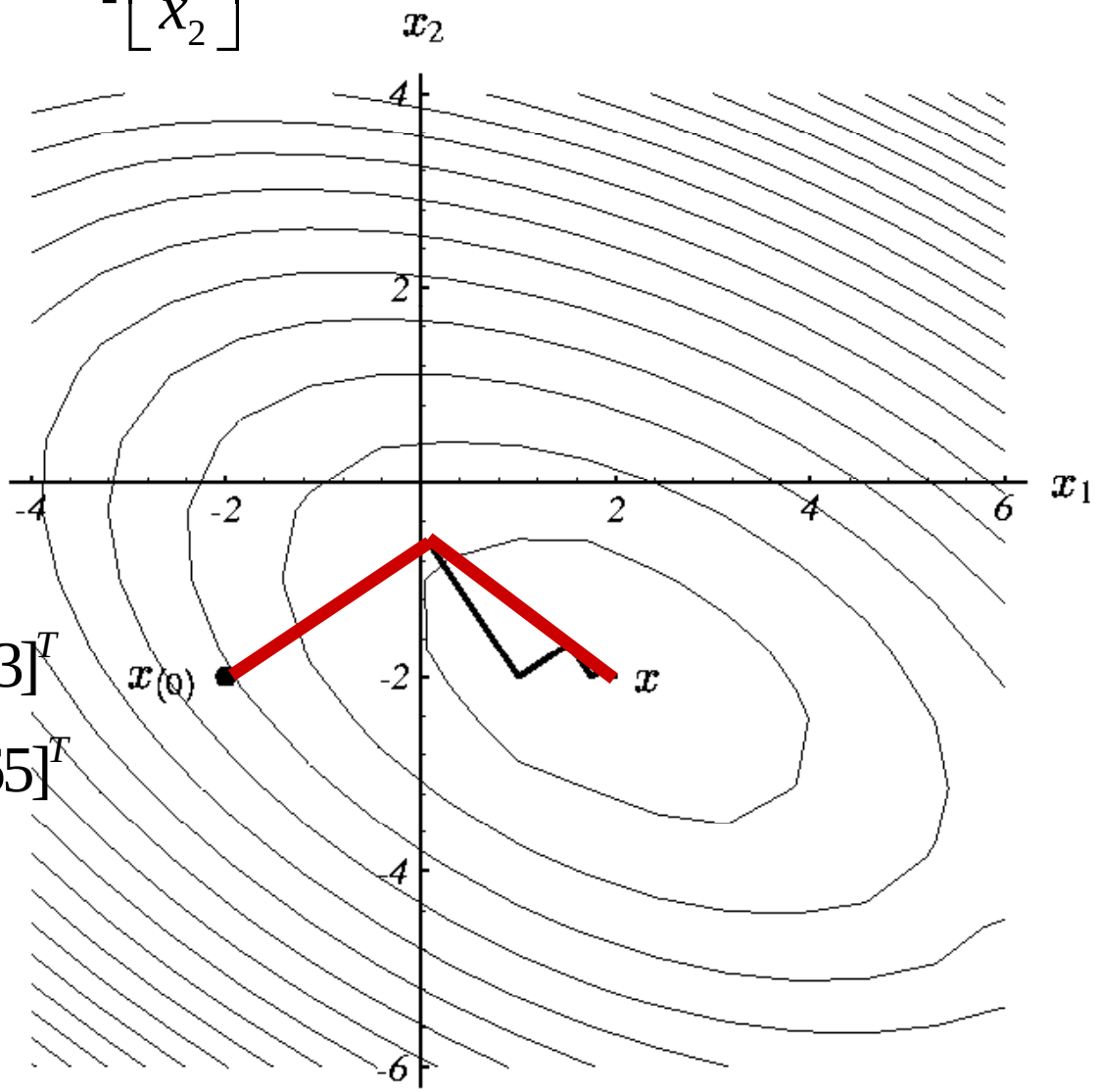
$$\boldsymbol{x}_{(1)}=\boldsymbol{x}_{(0)}+\alpha_{(0)}\boldsymbol{p}_{(0)}=[0.08,-0.6133]^T$$

$$\boldsymbol{p}_{(1)}=\boldsymbol{r}_{(1)}+\beta_{(0)}\boldsymbol{p}_{(0)}=[4.6592,-3.365]^T$$

$$\alpha_{(1)}=0.4121$$

$$\boldsymbol{x}_{(2)}=\boldsymbol{x}_{(1)}+\alpha_{(1)}\boldsymbol{p}_{(1)}=[2,\ -2]^T$$

$$\boldsymbol{r}_{(2)}=\boldsymbol{0},\ \boldsymbol{p}_{(2)}=\boldsymbol{0}$$



共轭梯度法—总结

对2x2矩阵证明？

- (1)理论上，经过n步迭代一定得到准确解。 中断？
- (2)迭代解的残差向量 $\{\boldsymbol{r}_0, \boldsymbol{r}_1, \cdots\}$ 两两正交。
- (3)搜索方向向量 $\{\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots\}$ 两两A正交。
- (4)当前解的残差向量与前面所有搜索方向都正交；与除前一个外的前面所有搜索方向都A正交。
- (5)x_{i+1}是x_0+span$\{\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots, \boldsymbol{p}_i\}$ 集合中使 $\|\boldsymbol{e}\|_A$ 达最小的向量。
- (6)x_{i+1}是x_0+span$\{\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots, \boldsymbol{p}_i\}$ 集合中使$f(\boldsymbol{x})$达最小的向量。

$$\because\ f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{e}\|_A^2 - \frac{1}{2}\|\boldsymbol{x}^*\|_A^2$$

收敛性理论

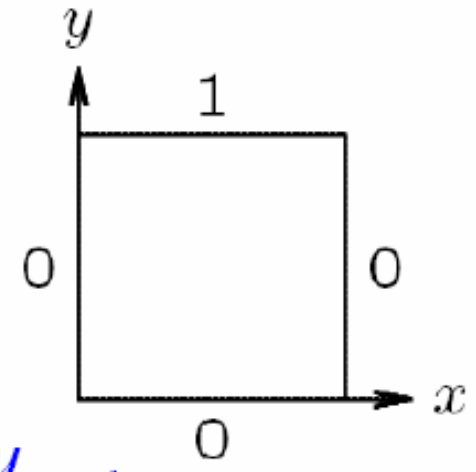
$$\frac{\|e_i\|_A}{\|e_0\|_A} \le 2\left(\frac{\sqrt{\kappa_2}-1}{\sqrt{\kappa_2}+1}\right)^i = 2\left(\frac{\sqrt{\lambda_1}-\sqrt{\lambda_n}}{\sqrt{\lambda_1}+\sqrt{\lambda_n}}\right)^i,\ \kappa_2 = \mathrm{cond}_2(A),\ i = 1, \cdots, n$$

矩阵A对称正定

实际常有“超线性”收敛速度

Example: Laplace equation on unit square

$$u_{xx} + u_{yy} = 0, \quad 0 \le x, y \le 1$$

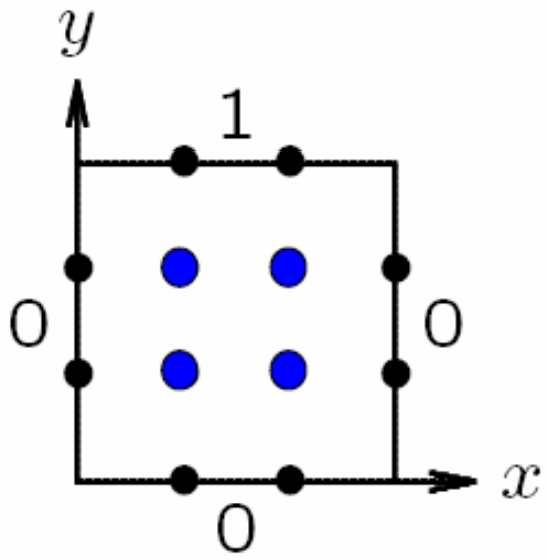


5-point 2nd order centered-difference scheme:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0, \quad i, j = 1, \ldots, n$$

$n = 2$:

$$Ax = \begin{bmatrix} 4 & -1 & -1 & \\ -1 & 4 & & -1 \\ -1 & & 4 & -1 \\ & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{1,2} \\ u_{2,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = b$$



Example: Iterative Methods

We illustrate various iterative methods by using them to solve 4×4 linear system for Laplace equation example

In each case we take $x_0 = 0$ as starting guess

Jacobi method gives following iterates:

k	x_1	x_2	x_3	x_4
0	0.000	0.000	0.000	0.000
1	0.000	0.000	0.250	0.250
2	0.062	0.062	0.312	0.312
3	0.094	0.094	0.344	0.344
4	0.109	0.109	0.359	0.359
5	0.117	0.117	0.367	0.367
6	0.121	0.121	0.371	0.371
7	0.123	0.123	0.373	0.373
8	0.124	0.124	0.374	0.374
9	0.125	0.125	0.375	0.375

Example Continued

Gauss-Seidel method gives following iterates:

k	x_1	x_2	x_3	x_4
0	0.000	0.000	0.000	0.000
1	0.000	0.000	0.250	0.312
2	0.062	0.094	0.344	0.359
3	0.109	0.117	0.367	0.371
4	0.121	0.123	0.373	0.374
5	0.124	0.125	0.375	0.375
6	0.125	0.125	0.375	0.375

SOR method (with optimal $\omega = 1.072$ for this problem) gives following iterates:

k	x_1	x_2	x_3	x_4
0	0.000	0.000	0.000	0.000
1	0.000	0.000	0.268	0.335
2	0.072	0.108	0.356	0.365
3	0.119	0.121	0.371	0.373
4	0.123	0.124	0.374	0.375
5	0.125	0.125	0.375	0.375

Example Continued

CG method converges in only two iterations for this problem:

k	x_1	x_2	x_3	x_4
0	0.000	0.000	0.000	0.000
1	0.000	0.000	0.333	0.333
2	0.125	0.125	0.375	0.375

比较一下几种迭代法的公式

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

G-S: $x^{(k+1)} = (D+L)^{-1}\left(b - Ux^{(k)}\right)$

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

收敛性取决于谱半径 $\rho(G) = \max|\lambda|$

CG: 。。。

(每步迭代，一次矩阵向量乘)

每步迭代的计算量大体相同，计算量取决于迭代步数，也即收敛速度

Rate of Convergence

迭代法的收敛性评价

$$\lim_{k\to\infty}\frac{\|e_{k+1}\|}{\|e_k\|^r}=c$$

r=1, 且**c<1,** 线性收敛
r>1, 超线性收敛
r=2, 平方收敛
… …

根据误差递减情况判断收敛阶：
10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, …
10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, …
10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}, …
10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}, …

Linear, with
c=10^{-1}, 每步获**1**位精度
c=10^{-2} , 每步获**2**位精度

每步获精度位数：$R=-\log_{10}(c)$

c ↘ **R** ↗

For more systematic comparison of methods, we compare them on $k\times k$ model grid problem for Laplace equation on unit square

For this simple problem, spectral radius of iteration matrix for each method can be determined analytically, as well as optimal ω for SOR

Gauss-Seidel is asymptotically twice as fast as Jacobi for this model problem, but for both methods, number of iterations per digit of accuracy gained is proportional to number of mesh points **1/R**

Optimal SOR is order of magnitude faster than either of them, and number of iterations per digit gained is proportional to number of mesh points along one side of grid

Rate of Convergence, continued

Convergence behavior of CG is more complicated, but error is reduced at each iteration by factor of roughly

对称正定矩阵

$$c = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$$

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$$

This estimate is conservative, and CG method may do much better

保守的

If matrix A has only m distinct eigenvalues, then theoretically CG converges in at most m iterations

m个不同的特征值

Detailed convergence behavior depends on entire spectrum of A, not just its extreme eigenvalues, and in practice convergence is often superlinear

Preconditioning

记住：这里的A均对称正定

Convergence rate of CG can often be substantially accelerated by *preconditioning*

Apply CG algorithm to $M^{-1}A$, where M is chosen so that $M^{-1}A$ is better conditioned and systems of form $Mz = y$ are easily solved

M^{-1}与A^{-1}近似

Some choices of preconditioners include:

- Diagonal or block-diagonal
- SSOR
- Incomplete factorization
- Polynomial
- Approximate inverse

实际上为保证对称性，对新矩阵$L^{-1}AL^{-T}$用CG解，其中$M=LL^T$

对原算法进行适当的修改后，不须显式地包含L矩阵，见算法 **11.2, pp. 407**

CG算法 – 预条件

$$\hat{A} = L^{-1} A L^{-T}$$

$$Ax = b \Leftrightarrow \hat{A}\hat{x} = \hat{b}$$

其中 $\hat{b} = L^{-1} b,\ \hat{x} = L^T x$

对 $\hat{A}\hat{x} = \hat{b}$ 应用**CG算法**：

不要生成 $\hat{A}$!

1) 选择x_0, r_0=b-Ax_0

$$\hat{x}_0 = L^T x_0\ ,\ \hat{r}_0 = \hat{b} - \hat{A}\hat{x}_0 = L^{-1} r_0\ ,\ \hat{p}_0 = \hat{r}_0$$

2) 对j=0, 1, … 直到 $\left\|\hat{r}_j\right\| \le \varepsilon$ (或$\left\|\hat{r}_j\right\| \Big/ \left\|\hat{b}\right\| \le \varepsilon$)

$$\alpha_j = \frac{\hat{r}_j^T \hat{r}_j}{\hat{p}_j^T \hat{A} \hat{p}_j} = \frac{\hat{r}_j^T \hat{r}_j}{\hat{p}_j^T L^{-1} A L^{-T} \hat{p}_j} = \frac{\hat{r}_j^T \hat{r}_j}{p_j^T A p_j} \quad (\text{设} p_j = L^{-T} \hat{p}_j)$$

$$\hat{x}_{j+1} = \hat{x}_j + \alpha_j \hat{p}_j \quad \Rightarrow x_{j+1} = x_j + \alpha_j p_j$$

$$\hat{r}_{j+1} = \hat{r}_j - \alpha_j \hat{A} \hat{p}_j \quad \Rightarrow \hat{r}_{j+1} = \hat{r}_j - \alpha_j L^{-1} A p_j$$

$$\beta_j = \frac{\hat{r}_{j+1}^T \hat{r}_{j+1}}{\hat{r}_j^T \hat{r}_j}$$

$$\hat{p}_{j+1} = \hat{r}_{j+1} + \beta_j \hat{p}_j \quad \Rightarrow p_{j+1} = L^{-T} \hat{r}_{j+1} + \beta_j p_j$$

ALGORITHM : Conjugate Gradient with Split Preconditioner

1. **Compute** $r_0 := b - Ax_0$; $\hat{r}_0 = L^{-1}r_0$; **and** $p_0 := L^{-T}\hat{r}_0$.
2. **For** $j = 0, 1, \ldots,$ **until convergence Do:**
3. $\alpha_j := (\hat{r}_j, \hat{r}_j)/(Ap_j, p_j)$
4. $x_{j+1} := x_j + \alpha_j p_j$
5. $\hat{r}_{j+1} := \hat{r}_j - \alpha_j L^{-1}Ap_j$
6. $\beta_j := (\hat{r}_{j+1}, \hat{r}_{j+1})/(\hat{r}_j, \hat{r}_j)$
7. $p_{j+1} := L^{-T}\hat{r}_{j+1} + \beta_j p_j$
8. **EndDo**

$p_0 = M^{-1}r_0$

$M = LL^T$

等号两边乘L

$(L^{-1}r_j, L^{-1}r_j) = (r_j, L^{-T}L^{-1}r_j)$
$= (r_j, M^{-1}r_j)$

$L^{-T}L^{-1}r_{j+1} = M^{-1}r_{j+1}$

令$z_j = M^{-1}r_j$?

1) 相比不带预条件的CG算法，每个迭代步中要额外求解L^{-1}和L^{-T}为系数的线性方程组各一次

2) 算法的形式比较简洁，但要事先得到M的Cholesky分解

M^{-1}与A^{-1}近似

ALGORITHM : *Preconditioned Conjugate Gradient*

1. **Compute** $r_0 := b - Ax_0$, $z_0 = M^{-1}r_0$, **and** $p_0 := z_0$
2. **For** $j = 0, 1, \ldots,$ **until convergence Do:**
3. $\alpha_j := (r_j, z_j)/(Ap_j, p_j)$
4. $x_{j+1} := x_j + \alpha_j p_j$
5. $r_{j+1} := r_j - \alpha_j A p_j$
6. $z_{j+1} := M^{-1} r_{j+1}$
7. $\beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j)$
8. $p_{j+1} := z_{j+1} + \beta_j p_j$
9. **EndDo**

与课本**pp. 407**
算法**11.2**相同

1) 相比不带预条件的CG算法，每次迭代要额外求解*M*为系数的线性方程组一次

2) 算法的形式简洁，不需要事先算*M*的Cholesky分解