Chapter 6　线性方程组迭代法

张亚楠*

December 12, 2023

迭代法引入

矩阵的三角或者QR分解之后，解方程的工作量是$\boldsymbol{O}(n^2)$. 如果线性方程组的系数矩阵是大型稀疏矩阵 (例如求微分方程的差分方法或者有限元 方法产生的线性方程组)，迭代一次的工作量与矩阵的稀疏程度有关，例如椭圆算子 的标准五点差分格式，稀疏矩阵的每行只有5个非零元素，迭代一次需要5n次乘法。 若n很大，且迭代收敛较快时，迭代法的运算量只有$\boldsymbol{O}(n)$

记 x^*是

$$Ax = b$$

的精确解

1. 直接法的目标是找到 x^*本身(忽略浮点数计算产生的舍入误差)
2. 迭代法的目标是找到x^* 的有效近似解，例如

$$\|x - x^*\| < \tau = 10^{-9} \quad \text{OR} \quad \|b - Ax^*\|/\|b\| < 10^{-6}$$

则x 即是所求。 只要满足容许的误差τ 即可

如何得到近似解x? 从方程组出发构造迭代格式，迭代格式产生向量序列$\boldsymbol{x}^k$,

$$x^{(0)}, x^{(1)}, \ldots, x^{(k)}, \ldots$$

向量序列收敛即可！！

*ynzhang@suda.edu.cn　苏州大学数学科学学院

1 矩阵范数和条件数

1.1 向量范数

设 $\boldsymbol{x} \in \boldsymbol{R}^n$, 内积空间诱导范数一般称之为2范数。更一般的，可以定义p范数

$$\boldsymbol{x} = (x_1, x_2, ..., x_n)^{\mathrm{T}}; \quad \|\boldsymbol{x}\|_p = \left(\sum_{j=1}^{n} |x_j|^p \right)^{\frac{1}{p}}, \ 1 \le p < +\infty$$

本课程常用三种范数

(1) 1-范数，p = 1,

$$\|\boldsymbol{x}\|_1 = \sum_{j=1}^{n} |x_j|$$

(2) 2-范数，p = 2,

$$\|\boldsymbol{x}\|_2 = \left(\sum_{j=1}^{n} |x_j|^2 \right)^{\frac{1}{2}}$$

(3) ∞-范数， p→∞

$$\|\boldsymbol{x}\|_\infty = \lim_{p\to\infty} \|\boldsymbol{x}\|_p = \lim_{p\to\infty} \left(\sum_{j=1}^{n} |x_j|^p \right)^{\frac{1}{p}} = \max_{1\le j\le n} |x_j|$$

范数具有如下几个性质:

(1) 正定性 $\|\boldsymbol{x}\| \ge 0$ 且 等号成立当且紧当$\boldsymbol{x}$是零元

(2) 齐次性 $\|\alpha \boldsymbol{x}\| = |\alpha| \cdot \|\boldsymbol{x}\|$， $\forall \alpha \in \mathbb{C}$

(3) 三角形不等式 $\|\boldsymbol{x} + \boldsymbol{y}\| \le \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$

范数用来衡量一个向量的大小，长度， 类似于实数或复数的绝对值； 运算结果是一个非负数. 从映射的角度看，n 维向量的范数又是一个n元函数； 可以证明该函数(此时称之为范数) 对每一个自变量(此时称之为向量的分量)均是连续的. 由此，可以证明如下重要结论

定理 1 $\boldsymbol{R}^n$ 空间中任意两个范数 均是等价的. 即存在正数c, 对任意的$\boldsymbol{x} \in \boldsymbol{R}^n$, 有

$$\frac{1}{c}\|\boldsymbol{x}\|_\diamondsuit \le \|\boldsymbol{x}\|_\heartsuit \le c\|\boldsymbol{x}\|_\diamondsuit$$

由等价性可知： 如要说明一个向量序列收敛，则只需要在其中一个范数下说明收敛即可。

定义 1 设 $\{\boldsymbol{x}^{(k)}\}$ 是 $\boldsymbol{R}^n$ 中的向量序列， $\boldsymbol{x}^* \in \boldsymbol{R}^n$, 若对每一个向量分量，都有

$$\lim_{k\to\infty} x_j^{(k)} = x_j^*, \quad 1 \le j \le n$$

则称序列$\boldsymbol{x}^{(k)}$收敛到 $\boldsymbol{x}^*$, 记为

$$\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}^*$$

定理 2 $\boldsymbol{R}^n$ 中序列收敛等价于以任何一种范数收敛

$$\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}^* \Leftrightarrow \lim_{k\to\infty} \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\| = 0$$

1.2 矩阵范数

在误差分析中，矩阵范数大多时候与向量范数会同时出现，因此希望二者之间 有一定的联系和统一性。

定义 2 (从属范数) 任给矩阵 $A \in \boldsymbol{R}^{n\times n}$, 以及向量范数 $\|x\|_{\spadesuit}$,(如： $\spadesuit = 1,2,\infty$), 则

$$\|A\|_{\spadesuit} = \sup_{x\neq 0} \frac{\|Ax\|_{\spadesuit}}{\|x\|_{\spadesuit}} = \max_{x\neq 0} \frac{\|Ax\|_{\spadesuit}}{\|x\|_{\spadesuit}} = \max_{\|x\|_{\spadesuit}=1} \|Ax\|_{\spadesuit}$$

为从属于$\spadesuit$范数的矩阵范数， 也称为矩阵A的算子范数.

矩阵范数满足范数的定义，同时还具有如下性质

$$\|AB\| \leq \|A\| \cdot \|B\|; \quad \|Ax\| \leq \|A\| \cdot \|x\|$$

例 1 证明:

$$\|AB\| \leq \|A\| \cdot \|B\|$$

提示:

$$\|AB\| = \|AB * y\|, \quad \|y\| = 1$$

进而

$$\|AB * y\| \leq \|A\| * \|B * y\| = \|A\| \cdot \|B\|$$

例 2 常用的三种范数 公式

- $\|A\|_{\infty} = \max\limits_{1\leq i\leq n} \sum\limits_{j=1}^{n} |a_{ij}|$
- $\|A\|_{1} = \max\limits_{1\leq j\leq n} \sum\limits_{i=1}^{n} |a_{ij}|$
- $\|A\|_{2} = \sqrt{\lambda_{\max}(A^{\mathrm{T}}A)}$

注意到矩阵的2范数即是最大奇异值(singular value)！ 上述矩阵范数的表达式可按照定义证明, 留作作业.

标注 1 任意矩阵$A \in \mathbb{R}^{m\times n}$, $m \geq n$ 存在正交矩阵$U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$, 对角阵$\Sigma \in \mathbb{R}^{m\times n}$, 满足

$$A = U\Sigma V^{\mathrm{T}}, \quad \Sigma = \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \\ \hline & & & O \end{array}\right]_{m\times n}, \quad r = rank(A), \quad \sigma_j > 0.$$

这里的Σ不是严格意义的对角矩阵, 未列出的地方均为零. 奇异值分解(SVD)是矩阵计算的重要手段，我们这里给出可逆矩阵SVD分解存在性的解释，进一步了解可参考《矩阵论》或者《矩阵计算》等书籍. 设A为n阶可逆方阵，则存在正交矩阵V, 正定对角阵S^2, 满足

$$A^{\mathrm{T}}AV = VS^2 \Rightarrow V^{\mathrm{T}}A^{\mathrm{T}}AV = S^2 \Rightarrow S^{-1}V^{\mathrm{T}}A^{\mathrm{T}}AVS^{-1} = I$$

检验$U = AVS^{-1}$ 为正交矩阵且成立 $A = USV^{\mathrm{T}}$.

思考： 正交矩阵的2范数是多少？ 考察 $\|Qx\|_2^2 = \left(Qx, Qx\right) = x^{\mathrm{T}}Q^{\mathrm{T}}Qx = 1.$

例7：p166

$$A = \begin{pmatrix} 1 & -2 \\ -3 & 4 \end{pmatrix}$$

计算1，2，无穷范数

定理 3 (对称矩阵的2范数) 设 $A \in \mathbb{R}^{n\times n}$,

- 任何一种范数均大于等于矩阵的谱半径: $\rho(A) \le \|A\|$.
- 若 $A = A^{\mathrm{T}}$, 则 $\|A\|_2 = \rho(A)$.

证明：(1) 记$|\lambda| = \rho(A)$, 且 $Ax = \lambda x$, &$\|x\| = 1$, 则

$$|\lambda|\|x\| = \|Ax\| \le \max_{\forall \|x\|=1} \|Ax\| = \|A\|$$

(2) 设$A = Q\Lambda Q^{\mathrm{T}}$, 现证明 $\forall \|x\|_2 = 1$, $\|Ax\|_2 \le \rho(A)$.

$$\|Ax\|_2 = \|Q\Lambda Q^{\mathrm{T}}x\| \le \|Q\|_2\|\Lambda\|_2\|Q^{\mathrm{T}}\|\|x\|_2 = \rho(A)$$

定理 4 若$\rho(B) < 1$, 则 $I \pm B$可逆； 进而还成立

$$\|(I \pm B)^{-1}\| \le \frac{1}{1 - \|B\|}$$

证明：假设$(I \pm B)$ 不可逆，则$(I \pm B)x = 0$ 存在非零解, 即$x \neq 0$. 也即 $\mp x = Bx$, 这与$\rho(B) < 1$矛盾.

$$(I\pm B)^{-1}(I\pm B) = I \rightarrow (I\pm B)^{-1} \pm (I\pm B)^{-1}B = I \rightarrow (I\pm B)^{-1} = I \mp (I\pm B)^{-1}B \rightarrow \|(I\pm B)^{-1}\| \le 1 + \|(I\pm B)^{-1}\|\|B\|$$

标注 2 注意，对 $A, B \in \mathbb{R}^{n\times n}$,

$$\rho(AB) \not\le \rho(A)\rho(B)$$

一般不成立. 反例如下：

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad \rho(A) = 1,\ \rho(B) = \sqrt{2},\ \rho(AB) = 2$$

1.3 矩阵条件数

李庆扬教材 p167 例

例 3 考虑计算目标

$$\begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

若实际计算时右端项存在小的扰动误差，例如

$$\begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2.0001 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

万分之一的右端项扰动得到百分之百的误差？WHY?

右端项有一微小的扰动，所得结果千差万别 。 这种现象称之为“病态”；矩阵

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1.0001 \end{pmatrix}$$

称为病态矩阵；

思考： 为什么右端项小的扰动，解的扰动这么大?

考虑$Ax = b$; 如果右端项有微小扰动δb, 则实际求解问题为

$$A(x + \delta x) = b + \delta b$$

分析解x 的相对误差

$$\frac{\delta x}{x} = ? \quad \text{or} \quad \frac{\|\delta x\|}{\|x\|} = ?$$

$$\delta x = A^{-1} * \delta b \Rightarrow \underline{\|\delta x\| \leq \|A^{-1}\| * \|\delta b\|}$$

$$Ax = b \Rightarrow \|Ax\| = \|b\| \Rightarrow \|A\| * \|x\| \geq \|b\| \Rightarrow \underline{\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}}$$

可得如下关系

$$\frac{\|\delta x\|}{\|x\|} \leq \underline{\|A\| * \|A^{-1}\|} \frac{\|\delta b\|}{\|b\|}$$

右端项扰动所带来的解的扰动完全体现于条件数

$$\text{cond}(A) = \|A^{-1}\| * \|A\|$$

不同的范数对应不同的条件数

条件数的性质：

- 任意矩阵条件数均大于1

$$\text{cond(A)} = \|\text{A}^{-1}\| * \|\text{A}\| \geq \|\text{A}^{-1}\text{A}\| = 1$$

- 任意常数$c \neq 0$

$$\text{cond(cA)} = \text{cond(A)}$$

- 当R为正交矩阵时，

$$\mathrm{cond(R)}_2 = 1; \quad \mathrm{cond(RA)}_2 = \mathrm{cond(AR)}_2 = \mathrm{cond(A)}_2$$

Hint: 条件数的前两条性质按照定义检验即可；证明第三条；

$$R * R^{\mathrm{T}} = I \Leftrightarrow R^{-1} = R^{\mathrm{T}} \rightarrow cond(R) = \|R\|_2 * \|R^{\mathrm{T}}\|_2 = \|R\|_2^2 = 1$$

上式也可按照二范数的奇异值定义进行检验. 同样可以检验 A与AR具有相同的奇异值，因此他们条件数相等.

常用的条件数:

- 无穷条件数

$$\mathrm{cond(A)}_\infty = \|\mathrm{A}^{-1}\|_\infty * \|\mathrm{A}\|_\infty$$

- 谱条件数

$$\mathrm{cond(A)}_2 = \|\mathrm{A}^{-1}\|_2 * \|\mathrm{A}\|_2 = \sqrt{\frac{\lambda_{\max}(\mathrm{A}^{\mathrm{T}}\mathrm{A})}{\lambda_{\min}(\mathrm{A}^{\mathrm{T}}\mathrm{A})}} = \frac{\sigma_1}{\sigma_{\mathrm{n}}}$$

当A为对称矩阵时，

$$\mathrm{cond(A)}_2 = \frac{|\lambda_1|}{|\lambda_{\mathrm{n}}|}$$

计算矩阵的条件数是困难的事情，求矩阵的奇异值或者特征值的计算量一般大于求解线性方程组. 实际应用时，一般是猜测或者估计条件数的大小。例如， 当矩阵的行列式绝对值接近0的时候； 或者 A 的元素间数量级相差很大且无规则，A的条件数可能很大[1]。

李庆扬p171 例9 Hilbert矩阵条件数

2 引例和迭代法一般原理

- 迭代法的一般原理
- 基本迭代法：Jacobi, Gauss-Seidel, SOR
- 最速下降法，共轭梯度法，多重网格，...(本课程略)

李庆扬教材p180例1

[1]只是猜测，未必准确. 在实际计算时判断矩阵的条件数的大小是简单的事情，只需要给出 测试精确解x, 计算$b = Ax$, 再重新计算$Ax = b$, 观察得到的解与x误差大小即可。 读者可测试如下Matlab代码

```
n = 9; A = hilb(n); x = rand(n,1); err = x - A \(A * x); norm(err)
```

例 4 将线性方程组$Ax=b$

$$\begin{cases} 2x+4y-2z &= 2 \\ 4x+9y-3z &= 8 \\ -2x-3y+7z &= 10 \end{cases}$$

改写为

$$\begin{cases} 2x &= 2-\left(4y-2z\right) \\ 9y &= 8-\left(4x-3z\right) \\ 7z &= 10-\left(-2x-3y\right) \end{cases} \rightarrow \begin{cases} x &= \left[2-\left(4y-2z\right)\right]/2 \\ y &= \left[8-\left(4x-3z\right)\right]/9 \\ z &= \left[10-\left(-2x-3y\right)\right]/7 \end{cases}$$

也即是：将$Ax=b$改写成等价形式$x=Bx+f$, 据此构造迭代格式

$$\boldsymbol{x}^{(k+1)}=B\boldsymbol{x}^{(k)}+f, \quad k=0,1,2,...$$

给定初值向量$\boldsymbol{x}^{(0)}$(如零向量)， 反复作用上式可得到一个向量序列

$$\boldsymbol{x}^{(0)},\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},...,\boldsymbol{x}^{(n)}...$$

思考：若随着迭代次数增加，向量序列不再变化了(收敛！！)，是否得到了目标解?

1. $Ax=b \Leftrightarrow x=Bx+f$ 等价形式一定有：$x=x-c*(b-Ax)$
2. 迭代序列 $\boldsymbol{x}^{(k)}$ 啥时候收敛？ 上面的例子就不收敛

定义 3 对上述产生的向量序列， 若

$$\lim_{k\to\infty}\boldsymbol{x}^{(k)}=\boldsymbol{x}^{*}$$

则称迭代法收敛，且$\boldsymbol{x}^{*}$即是方程组的解，否则称为发散.

引进误差向量

$$\boldsymbol{\varepsilon}^{(k)}=\boldsymbol{x}^{(k)}-\boldsymbol{x}^{*}$$

可知迭代序列是否收敛等价于

$$\lim_{k\to\infty}\boldsymbol{\varepsilon}^{(k)}=0$$

注意到 x^{*}满足

$$\boldsymbol{x}^{*}=B\boldsymbol{x}^{*}+f$$

上式与迭代格式对应相减，即得

$$\boldsymbol{\varepsilon}^{(k+1)}=B\boldsymbol{\varepsilon}^{(k)}=...=B^{k+1}\boldsymbol{\varepsilon}^{(0)}$$

观察上式，$\boldsymbol{\varepsilon}^{(0)}$是首次迭代的误差，可以是任意向量；
$\boldsymbol{\varepsilon}^{(k+1)}$是否收敛于0向量, 完全取决于B

问题：什么样的B？ 对任意初值，可使下式成立

$$\lim_{k\to\infty}\boldsymbol{\varepsilon}^{(k+1)}=\lim_{k\to\infty}B^{k+1}\boldsymbol{\varepsilon}^{(0)}=\mathbf{0}$$

或者

$$\lim_{k\to\infty}\|\boldsymbol{\varepsilon}^{(k+1)}\|=\lim_{k\to\infty}\|B^{k+1}\boldsymbol{\varepsilon}^{(0)}\|=0$$

分析：

$$\|B^{k+1}\boldsymbol{\varepsilon}^{(0)}\|\le\|B^{k+1}\|*\|\boldsymbol{\varepsilon}^{(0)}\|\le\|B\|^{k+1}*\|\boldsymbol{\varepsilon}^{(0)}\|$$

结论：若存在某种矩阵范数使得$\|B\|<1$, 则迭代收敛.

说明： 存在不代表一定找得到，利用矩阵范数和谱半径之间的关系

- 对任意矩阵算子范数

$$\rho(B)\le\|B\|$$

- $\forall\varepsilon>0$, 存在一个矩阵算子范数 $\|\cdot\|_\diamond$, 使得

$$\|B\|_\diamond\le\rho(B)+\varepsilon$$

标注 3 上述结论第一条前文已经证明。 第二条性质证明略，可参考[2].

定理 5 设$B\in\boldsymbol{R}^{n\times n}$, 则下面命题等价

- 迭代格式

$$\boldsymbol{x}^{(k+1)}=B\boldsymbol{x}^{(k)}+f,\quad k=0,1,2,\ldots$$

对任意初值均收敛.

- $\rho(B)<1$
- 存在某种$\diamond$范数满足$\|B\|_\diamond<1$
- 对任意矩阵范数

$$\lim_{k\to\infty}\|B^k\|=0$$

即 B^k收敛到零矩阵, i.e.,

$$\lim_{k\to\infty}\left(B^k\right)_{ij}=0,\quad 1\le i,j\le n$$

[2]徐数方，高立，张平文，《数值线性代数》，北京大学出版社.

引理 1 (矩阵的Schur分解) 给定矩阵$A\in\mathbb{C}^{n\times n}$, 则存在上三角矩阵 T, 酉矩阵Q, 使得

$$A=QTQ^{\mathrm{H}},\quad QQ^{\mathrm{H}}=Q^{\mathrm{H}}Q=I$$

其中矩阵上标表示共轭转置 [3].

例 5 对任意给定上三角矩阵$T\in\mathbb{C}^{n\times n}$，且$|T_{ii}|\le\rho<1$, 证明:

$$\lim_{k\to\infty}T^k=0_{n\times n}$$

证明留作课后作业.

例题的结果表明：对任意方形复矩阵, 若其谱半径$\rho(A)<1$, 则当$k\to\infty$时， 其k次幂趋于零矩阵. 反之也成立. 于是这一结果便证明了迭代收敛定理.

如果找到某种范数满足 $\|B\|_\circ=q<1$, 则由迭代格式

$$\boldsymbol{\varepsilon}^{(k+1)}=B*\boldsymbol{\varepsilon}^{(k)}$$

可得(忽略下标)

$$\|\boldsymbol{\varepsilon}^{(k+1)}\|=\|B*\boldsymbol{\varepsilon}^{(k)}\|\le\|B\|*\|\boldsymbol{\varepsilon}^{(k)}\|$$

$$\|\boldsymbol{\varepsilon}^{(k+1)}\|\le\|B\|^{k+1}*\|\boldsymbol{\varepsilon}^{(0)}\|=q^{k+1}\|\boldsymbol{\varepsilon}^{(0)}\|$$

$$\|\boldsymbol{\varepsilon}^{(k+1)}-\boldsymbol{\varepsilon}^{(k)}\|=\|B\boldsymbol{\varepsilon}^{(k)}-\boldsymbol{\varepsilon}^{(k)}\|\ge\|\boldsymbol{\varepsilon}^{(k)}\|-q*\|\boldsymbol{\varepsilon}^{(k)}\|$$

即

$$\|\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\|\ge(1-q)*\|\boldsymbol{\varepsilon}^{(k)}\|$$

进而得到如下误差估计

定理 6 如果存在迭代矩阵的某种范数满足$\|B\|=q<1$, 则有如下估计

- 先验估计 $\|x^*-x^{(k)}\|\le q^k\|x^*-x^{(0)}\|$
- 后验估计 $\|x^*-x^{(k)}\|\le\dfrac{q}{1-q}\|x^{(k)}-x^{(k-1)}\|\le\dfrac{q^k}{1-q}\|x^{(1)}-x^{(0)}\|$

实际计算时上述估计式并不好用，WHY? 因为不知道谱半径和q。实际应用时多用残差(残量, residual) 来判断数值解的精度

$$r_k=b-Ax^{(k)}=Ax^*-Ax^{(k)}=A*\left(x^*-x^{(k)}\right)$$

或者相对残差 $\dfrac{r_k}{r_0}$ 作为迭代停止标准.

[3]利用Jordan标准型和QR分解可得结论.

3 三类简单迭代法

问题： 给定$Ax=b$, 如何选取迭代矩阵B, 且满足收敛性条件？ 不容易！！

$$Ax-b=0 \Leftrightarrow x=x-C*(Ax-b), \quad C \text{ 是与A同阶矩阵}$$

即

$$x=\Big(I-CA\Big)x+Cb$$

得到迭代格式

$$x^{(k+1)}=\Big(I-CA\Big)x^{(k)}+Cb, \quad k=0,1,2,...$$

观察上式，迭代矩阵即为$B=I-CA$, 由前文收敛性分析可知，若存在某种范数 使得$\|I-CA\|=q<1$, 则迭代格式收敛. 此时也可称C为A的广义逆矩阵. $q=0$时，有$CA=I$

NOTICE: 矩阵求逆比解方程组的工作量要大很多，线性方程组数值解必须 关心时效问题； 因此，A的广义逆C必须快速求解， 不可使用$C=A^{-1}$。

3.1 Jacobi迭代

考虑特殊情况，A的对角线元素数量级比该行其它元素都大. 形式上$A\approx D$ 则 $C=D^{-1}$, 进而迭代矩阵

$$B=I-CA=I-D^{-1}*A$$

将系数矩阵写成如下形式

$$A=D-L-U$$

其中D是对角矩阵，L是下三角部分，U是剩余上三角部分. 则

$$B=I-D^{-1}*(D-L-U)=D^{-1}*(L+U)$$

于是Jacobi迭代格式 $x=Bx+f$ 为

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

上述过程也可由 $Ax=(D-L-U)x=b$ 简单推导得到，留作练习。

Matlab 可快速计算矩阵乘法，应用时需要注意计算效率，不可随意取逆。 可按照表达式推导，方便写循环形式的代码。

$$\sum_{j=1}^{n}a_{ij}x_j=b_i, \quad i=1,2,...,n$$

则

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

若A是稠密矩阵，则每迭代一步需要n^2次乘法。

```
function x1 = JacobiIter(A,b,x0, tol, maxI)
% -- jacobi itration for Ax = b
if nargin<1
    n1 = 101;  A = toeplitz([3, -1, zeros(1,n1-2)]);
    b = (1:n1)'; A = sparse(A);
    x0 = b*0;    tol = 1e-6;  maxI = 100;
end
dA = diag(A);   r0 = norm(b - A*x0);
L = tril(A,-1); U = triu(A,1); B = L + U;
for k = 1:maxI
  x1 = (b - B*x0)./dA;
  r1 = norm(b - A*x1);
  if r1/r0 < tol
fprintf('\n itr converge at step %d and rel residual = %8.4e\n', k,r1/r0);
      break
  else
      x0 = x1;
  end
end
if k == maxI
fprintf('\n itr not converge rel residual = %8.4e\n', r1/r0)
end
```

观察Jacobi迭代格式发现，当计算第(k+1)步迭代所对应的第i个分量 $x_i^{(k+1)}$ 时，已经计算出 $x_j^{(k+1)}(j<i)$. 为何不用呢?

3.2 Gauss-Seidel

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

比较可知，Gauss-Seidel迭代和Jacobi迭代工作量几乎一样；GS可以少一个 迭代向量$\boldsymbol{x}$的存储, 实际意义不大，但是直观上GS效率优于Jacobi.

GS迭代的矩阵形式

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

即

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

若记G为系数矩阵A的下三角矩阵(连同对角线), 上式即为

$$x^{(k+1)} = G^{-1}Ux^{(k)} + G^{-1}b$$

也即是此时取A的广义逆C为下三角矩阵G的逆.

NOTICE: 上式仅仅用于分析和书写，不用于 实际计算。因为按照下三角矩阵求逆和乘法的计算量是$\boldsymbol{O}(n^3)$。 而按照表达式顺序计算时工作量与Jacobi一致. 实际计算时每一步迭代均在求解一个下三角方程组， 计算量为$\boldsymbol{O}(n^2)$。记$A=L+U$, 其中L是下三角包含对角线。

$$L*x^{(k+1)}=b-Ux^{(k)}$$

验证迭代矩阵的谱半径是否小于1比较困难，这里给出一般结论:

- 严格对角占优

$$|a_{ii}|>\sum_{j\geq i,\, j=1}^{n}|a_{ij}|,\quad i=1,2,...,n$$

  此时Jacobi , Gauss-Seidel均收敛.

- A对称正定, 则Gauss-Seidel收敛
- A以及$(2D-A)$ 均对称正定，Jacobi收敛

例 6 设 $A\in\mathbb{R}^{n\times n}$ 是严格对角占优矩阵，证明：A可逆(非奇异，满秩，行列式非零).

证明提示：反证法。假设$Ax=0$ 有非零解x, 取出x的最大模对应的分量指标j, 检验Ax 的第j个方程，利用对角占优性质推出矛盾.

例 7 若A为不可约(irreducible) 的弱对角占优：即 $a_{ii}\geq\sum_{j\neq i}a_{ij}$, $i=1,2,...,n$且 至少存在一个指标i使得不等式严格成立，则A可逆.

仿照上例证明，留作研究生作业。

例 8 记$D=\mathrm{diag}(A)$为对角矩阵，$N=A-D$, 则 Jacobi迭代法的迭代矩阵$B=D^{-1}N$, 证明：若A严格对角占优，则$\rho(B)<1$.

证明提示：反证法。假设 $Bx=x\lambda$且 $|\lambda|\geq 1$, 则

$$D^{-1}Nx=\lambda x\quad\rightarrow\quad Nx=\lambda Dx\quad\rightarrow\quad \|Nx\|_1=|\lambda|\|Dx\|_1\geq\|Dx\|_1=|a_{11}||x_1|+...+|a_{nn}||x_n|$$

并利用严格对角占优推出矛盾即可. 关于Gauss-Seidel和Jacobi方法的其它收敛性充分条件，可类似证明.

3.3 Successive Over Relax (SOR)

回忆几个名词：刘辉割圆术的松弛技术，Romberg 积分，Richardson 外推；
共同特点： 加权平均.

对GS迭代的第（k+1）步计算得到$\overline{x_i^{(k+1)}}$, 可采用加权平均技术，令

$$x_i^{(k+1)}=(1-w)x_i^{(k)}+w\overline{x_i^{(k+1)}}=x_i^{(k)}+w\left(\overline{x_i^{(k+1)}}-x_i^{(k)}\right)$$

为下一步迭代值.

需要指出的是：不像梯形公式有具体的收敛阶表达式，可以精确选取 松弛因子w; 若对系数矩阵和迭代矩阵没有足够的分析，好的w只能靠经验 和反复试验得到。

SOR迭代的矩阵形式如下：

$$x^{(k+1)} = (1-w)x^{(k)} + w\left[D^{-1} * \left(Lx^{(k+1)} + Ux^{(k)} + b\right)\right]$$

简单推导可得：

$$\begin{aligned} & Dx^{(k+1)} = (1-w)Dx^{(k)} + w\left(Lx^{(k+1)} + Ux^{(k)} + b\right) \\ \Rightarrow & (D-wL)x^{(k+1)} = \left[(1-w)D + wU\right]x^{(k)} + wb \\ \Rightarrow & x^{(k+1)} = (D-wL)^{-1}\left[(1-w)D + wU\right]x^{(k)} + w(D-wL)^{-1}b \end{aligned}$$

波浪线部分为迭代矩阵B, 系数矩阵A的广义逆

$$C = w(D-wL)^{-1}$$

w取值介于(0,2) 之间，以1为界，分别称为低松弛和超松弛

```
function x1 = GSIter(A,b,x0, tol, maxI)
% -- jacobi itration for Ax = b, maxI
if nargin<1
    n1 = 11;  A = toeplitz([3, -1, zeros(1,n1-2)]);
    b = (1:n1)'; A = sparse(A);
    x0 = b*0;    tol = 1e-6;  maxI = 100;
end
r0 = norm(b - A*x0);   L = tril(A);  U = A-L;  w = 0.15;
for k = 1:maxI
  x1 = L\(b - U*x0);
  r1 = norm(b - A*x1);
  if r1/r0 < tol
 fprintf('\n itr converge at step %d and rel residual = %8.4e\n', k,r1/r0);
      break
  else
      x0 = x1 + w*(x1-x0);
  end
end
if k == maxI
fprintf('\n itr not convergent and rel residual = %8.4e\n', r1/r0)
end
```

4 CG共轭梯度法

本节假设 A 为n阶对称正定矩阵。 why Conjugate & Gradient ?

例 9 设$A=A^{\mathrm{T}}$, 记二次n 元函数

$$f(x)=\frac{1}{2}x^TAx-x^Tb,\quad x\in\mathbf{R}^n$$

检验

- 二次项表达式

$$g(x)=\sum_{k=1}^{n}x_k*(Ax)_k=\sum_{k=1}^{n}x_k*\sum_{j=1}^{n}a_{kj}*x_j=\sum_{j=1}^{n}\sum_{k=1}^{n}a_{kj}x_kx_j$$

- 偏导数

$$\frac{\partial g}{\partial x_1}=\sum_{k=1}^{n}a_{k1}x_k\Big|_{(j=1)}+\sum_{j=1}^{n}a_{1j}x_j\Big|_{(k=1)}=2\sum_{j=1}^{n}a_{1j}x_j=2(A(1,:),x)$$

- 梯度 （一阶导）

$$\nabla g(x)=2*Ax\quad\rightarrow\quad\nabla f=Ax-b$$

- Hessian(二阶导)

$$\frac{1}{2}\frac{\partial^2 g}{\partial x_1\partial x_l}=\frac{\partial}{\partial x_l}\left(\sum_{j=1}^{n}a_{1j}x_j\right)=a_{1l}\quad\rightarrow\quad\text{Hessian}\quad H_f=\nabla^2(f)=A$$

4.1 梯度下降法

由于A正定，则二次函数$f(x)$的极小值点必存在且为驻点，即 $Ax-b=0$

$$f(x^*)=\min f(x)\quad\Leftrightarrow\quad\nabla f(x^*)=Ax^*-b=0$$

想象自己在一个环绕的山腰，准备出发到山底！？

Q-(1) 给定任意初值x_0, 计算residual $r_0=b-Ax_0$. 如何更新下一步?

Q-(2) 注意到r_0 是$f(x_0)$ 的负梯度方向，最速下降方向，不错的选择!

$$x_1=x_0+\alpha*r_0,\quad\text{怎样选择合适的}\alpha$$

这一步走多远? $\alpha=?$

Q-(3) 当然是离极小值点越近越好, 或者说使得f越小越好! 任给一个方向使之沿着P方向更新

$$x_1=x_0+\alpha P$$

在P方向必有一个极小点，也即是合适的步长α.

$$\min f(x_0+\alpha P)\rightarrow\frac{df}{d\alpha}=\Big(\nabla f(x_0+\alpha P),P\Big)=0$$

i.e.,

$$\Big(b-A(x_0+\alpha P),P\Big)=0\rightarrow\Big(r_0-\alpha AP,P\Big)=0\rightarrow\alpha=\frac{(r_0,P)}{\Big(AP,P\Big)}$$

A-1) 给定任意初值x_0, 计算residual $r_0 = b - Ax_0$.

A-2) 选择 $P = r_0$为前进方向， 计算

$$\alpha = \frac{\left(r_0, P\right)}{\left(AP, P\right)} \qquad \text{update} \quad x_1 = x_0 + \alpha P$$

A-3) compute new residual $r_0 - \alpha AP$, repeat

梯度下降法代码：grad_descent.m

标注 4 本小节给出了梯度下降法也称作最速下降法，用于计算 $\min f(x) = \frac{1}{2}x^{\mathrm{T}}Ax - x^{\mathrm{T}}b$. 实际上对于一般的非二次函数$f(x)$, 梯度下降法也是适用的.

1) 选定出发点x_0, 和下降方向$P = -\nabla f(x_0)$(注意P也可选择为其它下降方向，不一定是 负梯度方向)

2) 计算$x_1 = x_0 + \alpha P$, 要求选择步长α使得

$$\frac{df(x_1)}{d\alpha} = \left(\nabla f(x_1), P\right) = \left(\nabla f(x_0 + \alpha P), P\right) = 0$$

由 $\nabla f(x_0 + \alpha P) \approx \nabla f(x_0) + \alpha\left[H_f(x_0)\right]P$, (类似于一元函数的Taylor展开) 得到

$$\left(\nabla f(x_0) + \alpha\left[H_f(x_0)\right]P, P\right) = 0 \rightarrow \alpha = -\frac{\left(\nabla f(x_0), P\right)}{\left(H_f P, P\right)} = -\frac{P^{\mathrm{T}}[\nabla f(x_0)]}{P^{\mathrm{T}}H_f P}$$

当 f取做本节的二次函数，且$P = -\nabla f(x_0)$时，上述过程和本节梯度下降法完全一致; 若P取做下节介绍的“共轭方向”时，上述即是共轭梯度法.

4.2 共轭向量组

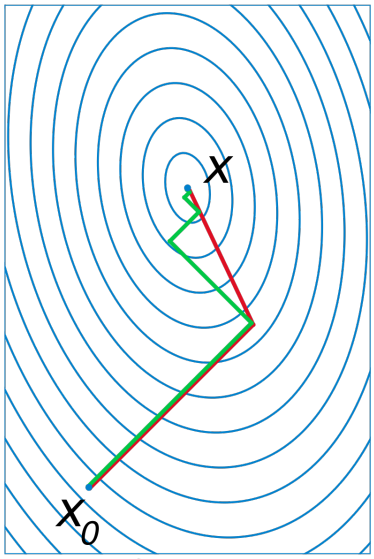
梯度下降法简单且应用广泛[4]， 称为局部最优方法，见wiki给出的图示。 如果选定了前进方向P_k, 最优步长α按照前文方法给出。

$$x_{k+1} = x_k + \alpha * P_k$$

由于α 选取的特殊性(沿着P_k 方向$f(x)$取最小值， $P_k^{\mathrm{T}} \cdot \nabla f(x_{k+1}) = 0$)， 得到两步残差满足

$$r_{k+1} \perp P_k, \rightarrow (r_{k+1}, P_k) = 0 \quad \& \quad \alpha = \frac{(r_k, P_k)}{\left(AP_k, P_k\right)}$$

[4]除求解线性方程组外，也可用于非线性问题



最速下降法是局部最优，每次前进方向到达该方向上$f(x)$的极小，改变方向，沿着负梯度方向前进 到达下一个极小，前后两次是正交的。右图绿线即是最速下降法，但不是全局最优的。下面选择更好的前进方向！

定义 4 A 对称正定，u,v 是n 维向量，则可定义A-内积

$$\langle u,v\rangle_A=(Au,v)=v^TAu=\sum_{j=1}^{n}\sum_{k=1}^{n}a_{jk}u_jv_k$$

若$\langle v,u\rangle_A=0$, 则u,v在$A-$内积意义下正交，称为A共轭。

例 10 检验上述定义的A内积，符合内积定义三要素

1. 对称性

$$\langle u,v\rangle_A=\langle v,u\rangle_A$$

2. 线性运算: 对任意$\alpha,\beta\in\mathbb{R}$, $u,v,w\in\mathbb{R}^n$

$$\langle\alpha u+\beta v,w\rangle_A=\alpha\langle u,w\rangle_A+\beta\langle v,w\rangle_A$$

3. 正定性

$$\langle u,u\rangle_A\geq 0,\quad \langle u,u\rangle_A=0\Leftrightarrow u=\vec{0}$$

例 11 给定一组非零向量 $p_1,p_2,...,p_k,\quad p\in\mathbb{R}^n$, & $k\leq n$, 若

$$\langle p_i,p_j\rangle_A=0,\ i\neq j$$

则$\forall\alpha_l,\ l=1,...,k$

$$\sum_{l=1}^{k}\alpha_lp_l=0\Rightarrow\alpha_l=0$$

即$\{p_j\}$彼此线性无关. 对上式依次与p_i做A内积即得结论。

4.3 CG作为直接法

如何理解$Ax=b$的解存在唯一?

$$\begin{bmatrix}|&|&\cdots&|\\a_1&a_2&\cdots&a_n\\|&|&\cdots&|\end{bmatrix}\begin{bmatrix}x_1\\\vdots\\x_n\end{bmatrix}=\begin{bmatrix}|\\b\\|\end{bmatrix}$$

1. 矩阵列满秩, 列向量可以选做一组基向量

$$span\{a_1,a_2,...,a_n\}=\boldsymbol{R}^n$$

2. 解方程等价于：寻找一组坐标 $x_1,...,x_n$使得b可以由列向量线性表示！！

$$b = x_1 * a_1 + x_2 * a_2 + ... + x_n * a_n$$

若A是正交矩阵或者列向量a_k 彼此正交，则容易计算出系数x_k

$$x_k = \left(a_k, b\right) \Big/ \left(a_k, a_k\right), \quad k = 1,2,...,n$$

思考：已知正交基很有效！如果有一组A共轭的向量组选做一组基，x_k 是不是也容易计算？？ 给定$\boldsymbol{R}^n$ 空间中一组彼此A共轭的向量组（可作为一组基）

$$P_1, P_2, \cdots, P_n$$

则目标向量$x = A^{-1}b$ 可由表示成这组基的线性组合

$$x = \sum_{j=1}^{n} \alpha_j P_j \rightarrow Ax = b = \sum_{j=1}^{n} \alpha_j AP_j \rightarrow P_k^T b = \alpha_k P_k^T AP_k$$

即：

$$\alpha_k = \frac{(P_k, b)}{\langle P_k, P_k \rangle_A} = \frac{P_k^T b}{P_k^T AP_k}$$

如果给定了一组共轭向量组P_k，则可按照上式求出对于系数α_k

主要任务：如何有效的给出一组A共轭向量组P_k??，CG方法即可以顺序给出A共轭向量组，同时又 给出了比最速下降法更有效的前进方向！！为书写方便，下文x_k表示向量在第k步迭代值，不指向量的第k个分量！

A-(1) 给定初值$x_0 = 0$(若无其它有效信息，初值取零向量)；计算残差（residual）

$$r_0 = b - Ax_0$$

注意残差r_0是个好东西，多用于判断迭代停止与否。 同时r_0是函数$f(x)$在点x_0的负梯度方向。前文已判断$f(x)$的驻点一定是极小值点， 负梯度是个好方向，因此可选$P_0 = r_0$

A-(2) 前进方向P_k确定之后，可以按照最速下降法的最优步长， 确定系数$\boxed{\alpha_k}$, 进而更新一次数值解

$$x_{k+1} = x_k + \alpha_k P_k \quad \rightarrow \text{New Residual} \quad r_{k+1} = b - Ax_{k+1}$$

α_k的选择要求满足函数f在P_k方向取极小，即x_{k+1}是该方向驻点

$$\frac{df}{d\alpha}(x_{k+1}) = \nabla f \cdot P_k = \boxed{r_{k+1} \cdot P_k} = (b - Ax_{k+1}, P_k) = (r_k - \alpha AP_k, P_k) = 0 \rightarrow \alpha_k = \frac{(P_k, r_k)}{P_k^T AP_k}$$

A-(3) 新的前进方向 P_{k+1}从哪来？先找一个与P_j, $j=1,2,...,k$线性无关的向量v, 可按照 Cram-Schmidt 正交化的方法

$$P_{k+1}=v-\sum_{j\le k}\frac{\langle P_j,v\rangle_A}{\langle P_j,P_j\rangle_A}P_j=v-\sum_{j\le k}\frac{P_j^TAv}{P_j^TAP_j}P_j$$

A-(4) 上一步可取$v=r_{k+1}$. 理由如下：

(b)-1 根据前进步长α的取法，知道$(r_{k+1},P_k)=0$, 非零向量正交必线性无关。

(b)-2 进一步发现: r_{k+1}与$\{P_0,P_1,\cdots,P_{k-1}\}$均线性无关

(b)-3 巧的是将r_{k+1}在P_k上做共轭投影:

$$P_{k+1}=r_{k+1}-\beta P_k,\quad \beta=\frac{(r_{k+1},AP_k)}{(P_k,AP_k)}$$

所得P_{k+1}满足与P_j, $0\le j\le k-1$也共轭。

标注 5 实际上，新的搜寻方向P_{k+1}的选取是为了满足在span$\{P_k,r_{k+1}\}$平面上找到$f(x)$的极小值点。读者可参考：徐树方、高立、张平文编著的教材《数值线性代数》

定理 7 (算法收敛性证明(A-(4)的回应)) 上述算法产生的向量序列r_j, P_j满足

$$r_l\perp span\{r_0,r_1,\cdots r_{l-1}\} \tag{1}$$

$$P_l\perp^A span\{P_0,P_1,\cdots P_{l-1}\} \tag{2}$$

$$span\{r_0,r_1,\cdots r_{l-1}\}=span\{P_0,P_1,\cdots P_{l-1}\}=span\{r_0,Ar_0,...,A^{l-1}r_0\} \tag{3}$$

Proof: 记号$\perp$表示欧氏正交； $\perp^A$表示A共轭； span{...}表示由向量组生成的子空间；(1)表示r_l正交于子空间中的任一向量. (2)表示P_l与子空间中的任一向量A共轭. 证明过程会用到以下等式

$$r_{k+1}=r_k-\alpha AP_k \qquad \Leftarrow x_{k+1}=x_k+\alpha P_k \tag{4}$$

$$(r_{k+1},P_k)=0 \tag{5}$$

$$P_k=r_k+\beta P_{k-1} \qquad \Leftarrow P_{k+1}\perp^A P_k \tag{6}$$

我们采用归纳法证明：

pf-1) 由 $P_0=r_0$, 验证（易知）

$$r_1\perp P_0,\quad r_1\perp r_0,\quad \&\quad P_1\perp^A P_0$$

pf-2) 注意到(6), 且$r_0=P_0$, r_k,P_k 可以彼此线性表出，则(3)第一个等号明显成立, 只要证明第二个等号即可。归纳假设 $j\le k$时， r_j 相互正交， P_j 彼此共轭, 且

$$\text{span}\{r_0,r_1,\cdots r_{l-1}\}=\text{span}\{r_0,Ar_0,...,A^{l-1}r_0\},\quad 0\le l\le k$$

现在说明

$$r_{k+1}\perp r_l,\quad P_{k+1}\perp P_l,\qquad l=0,...,k$$

- 考查 $l=k$

$$(r_{k+1},r_k)=(r_{k+1},P_k-\beta P_{k-1})=-\beta(r_{k+1},P_{k-1})=-\beta(r_k-\alpha_k AP_k,P_{k-1})=0$$

- 当$l<k$ 时，

$$(r_{k+1},r_l)=(r_k-\alpha AP_k,r_l)=-\alpha(AP_k,r_l)=-\alpha(AP_k,P_l-\beta P_{l-1})=0$$

以上说明$r_{k+1}\perp r_l,\ l\le k$, 则

$$r_{k+1}\perp \text{span}\{r_0,r_1,...,r_k\}=\text{span}\{P_0,P_1,...,P_k\}$$

现在证明P_{k+1}与之前的 P_l 彼此A共轭，

- $\langle P_{k+1},P_k\rangle_A=0$ 是P_{k+1}的生成要求，必然成立。
- 考虑$j<k$时，

$$\langle P_{k+1},P_j\rangle_A=\langle r_{k+1}+\beta P_k,P_j\rangle_A=\langle r_{k+1},P_j\rangle_A=(r_{k+1},AP_j)=\frac{1}{\alpha}(r_{k+1},r_{j+1}-r_j)=0$$

即

$$P_k\perp^A span\{P_0,P_1,\cdots P_{k-1}\}$$

以上证明了(1)-(2)以及(3)的第一个等号.

pf-3) 假设

$$\text{span}\{P_0,P_1,...,P_{k-1}\}=\text{span}\{r_0,r_1,...,r_{k-1}\}=\text{span}\{r_0,Ar_0,...,A^{k-1}r_0\}$$

现证明

$$\mathscr{R}^k=\text{span}\{r_0,r_1,...,r_k\}=\mathscr{K}^k=\text{span}\{r_0,Ar_0,...,A^kr_0\}$$

任意$x\in\mathscr{R}^k$, 将 x 正交分解为两部分

$$x=\underline{y}\in\mathscr{R}^{k-1}+c\underset{\sim}{r_k}=y+c(r_{k-1}-\alpha AP_{k-1})=\underline{\tilde{y}}\in\mathscr{R}^{k-1}+\beta*\underset{\sim}{AP_{k-1}}\in\mathscr{K}^k$$

上式最后一个等号成立因为，$\tilde{y}\in\mathscr{R}^{k-1}=\mathscr{K}^{k-1}$ 且 $P_k\in\mathscr{K}^{k-1}$. 则$x\in\mathscr{K}^k$. 也即是

$$\forall x\in\mathscr{R}^k\Rightarrow x\in\mathscr{K}^k;\quad\rightarrow\mathscr{R}^k\subset\mathscr{K}^k$$

证明 $\mathscr{K}^k\subset\mathscr{R}^k$留作作业。 二集合相互包含，必相等！！证毕！

标注 6 当$k=n$ 时，彼此正交的向量r必定有零向量， 因此最多n 步必得精确解。另外系数β 计算可简化

$$\beta=-\frac{\langle r_{k+1},P_k\rangle_A}{\langle P_k,P_k\rangle_A}=-\frac{(r_{k+1},AP_k)}{\langle P_k,P_k\rangle_A}=-\frac{(r_{k+1},\frac{1}{\alpha_k}(r_k-r_{k+1}))}{\langle P_k,P_k\rangle_A}$$
$$=\frac{1}{\alpha_k}\frac{(r_{k+1},r_{k+1})}{\langle P_k,P_k\rangle_A}=\frac{(r_{k+1},r_{k+1})}{(r_k,r_k)}$$

算法实现

A-1) 输入 $x_0, A, b \rightarrow r_0 = P_0$

A-2) $k = 0, 1, 2, \ldots$

$$\begin{cases} x_{k+1} = x_k + \alpha_k P_k, & \leftarrow \alpha_k = \frac{(r_k, r_k)}{(AP_k, P_k)} \\ r_{k+1} = b - Ax_{k+1} = r_k - \alpha AP_k \\ P_{k+1} = r_{k+1} + \beta_k P_k, & \leftarrow \beta_k = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} = -\frac{(r_{k+1}, AP_k)}{(P_k, AP_k)} \end{cases}$$

考查α的计算表达式

$$\alpha = \frac{\left(r_k, P_k\right)}{\left(AP_k, P_k\right)} = \frac{\left(r_k, r_k\right)}{\left(AP_k, P_k\right)}$$

以及下一步β的计算表达式，可以选择不同的组合，避免重复计算。

定理 8 共轭梯度法必收敛，且产生的第k步迭代值x_k满足如下误差估计

$$\|x_k - x_*\|_A \leqslant 2\left(\frac{\sqrt{\kappa_2} - 1}{\sqrt{\kappa_2} + 1}\right)^k \|x_0 - x_*\|_A$$

其中$\kappa_2 = \mathrm{cond}(A)_2$是矩阵$A$的2条件数。

4.4 预处理共轭梯度法

记M为A的一个有效近似，则M^{-1}称为A的一个广义逆，采用共轭梯度法求解$M^{-1}Ax = M^{-1}b$即为预处理共轭梯度法。在求解的过程中应该避免求逆并关注计算效率。

$$Ax = b, \quad \rightarrow \quad M^{-1}Ax = M^{-1}b$$

以下将引入记号

$$z = M^{-1}\left(b - Ax\right) = M^{-1} * r$$

为新的残量。由于A对称正定，M作为A的近似，可假定M为对称正定；类似于欧几里得内积，我们可以选择A内积或者M内积。

PCG-M-(1) 输入 $x_0, A, M, b \rightarrow$

$$r_0 = b - x_0, \quad z_0 = M^{-1} * r_0 = P_0$$

PCG-M-(2) 循环更新 $k=0,1,2,...$

$$
\begin{cases}
x_{k+1}=x_k+\alpha_k P_k, \quad \Leftarrow \alpha_k=\dfrac{\left(z_k,z_k\right)_{\clubsuit}}{\left(M^{-1}AP_k,P_k\right)_{\clubsuit}} \\
r_{k+1}=b-Ax_{k+1}=r_k-\alpha AP_k \\
z_{k+1}=M^{-1}*r_{k+1} \\
P_{k+1}=z_{k+1}+\beta_k P_k \quad \Leftarrow \beta=\dfrac{\left(z_{k+1},z_{k+1}\right)_{\clubsuit}}{\left(z_k,z_k\right)_{\clubsuit}}
\end{cases}
$$

上式计算系数 α,β 的表达式 没有给出具体的内积形式，不同的内积得到不同的算法，效果基本一样。 Matlab code : pcg2.m　pcg2A.m

```
function [ x0, epn, kk ] = pcg2(A,b,Tol,maxI,M1,x0)
%   Detailed explanation goes here
    epn = 1; kk = 0;
% if (nargin < 6)
% %     warning(message('MATLAB:pcg:NotEnoughInputs'));
%     n1 = length(b); x0 = zeros(n1,1);
% end
    r0 = b - A*x0 ;
    z0 = M1\r0;
    p0 = z0;
while kk < maxI && epn > Tol
    mj = A*p0;  % main cost 1
    alp = dot(r0,z0) ./ dot(mj,p0);
    x0 = x0 + alp*p0;
    r1 = r0 - alp*mj;
    z1 = M1 \ r1; % main cost 2
    epn = norm(z1(:),inf);
    bet  = dot(r1,z1) ./ dot(r0,z0);
    p0 = z1 + bet*p0;
    r0 = r1;   z0 = z1;   kk = kk+1;
end
fprintf('\n pcg method stop at itrstep = %d, and residual = %e\n',kk,epn);
end
```

以上代码以♣取M内积，也最为常见，算法过程只涉及一次矩阵乘法和一次预处理快速除法。 另一种以♣取A内积的代码可类似给出[5].

[5]文献说也是只涉及一次矩阵乘法， 两次乘法我会写，只用一次矩阵乘法 我还不知道怎样写。读者可以自己尝试。

标注 7 当预处理矩阵或者系数矩阵不是对称正定时，α和β计算公式需要采用原始公式，因为此时等价公式不准确. 若完全采用共轭方向也不是最佳方向。例如对于粗糙表面所在区域上求解Cahn-Hilliard方程，采用线性化 稳定格式计算所形成的线性方程组时；可选择PCG算法求解，此时预处理方向选择共轭方向和梯度方向的平均效果更好.

5 Krylov子空间(略)

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

$$\mathbb{K}_m(A, v) = \text{span}\{v, Av, A^2 v, ..., A^{m-1} v\}$$

是$\mathbb{R}^n$空间中的一个m维子空间，称之为Krylov子空间。

CG方法也可看作是Klyrov子空间方法。 取$x_0 = 0$, 迭代k次生成的数值解

$$x_k = 0 + \sum_{j=0}^{k-1} \alpha_j P_j, \quad x_k \in \text{span}\{P_0, P_1, ..., P_{k-1}\} = \text{span}\{r_0, r_1, ..., r_{k-1}\}$$

现说明

$$x_m \in \text{span}\{P_0, P_1, ..., P_{m-1}\} = \text{span}\{b, Ab, ..., A^{m-1} b\} = \mathbb{K}_m\left(A, b\right)$$

1) 已知$P_0 = r_0 = b$, 即

$$\text{span}\{b\} = \text{span}\{P_0\} = \text{span}\{r_0\}$$

2) 归纳假设 j = m之前上式成立，考虑j = m + 1时, 利用

$$r_{m+1} = r_m - \alpha A P_m$$

可知，任意 $z \in \text{span}\{r_0, ..., r_{m+1}\}$可分解为两部分

$$\underline{z_m \in \text{span}\{r_0, ..., r_m\} = \mathbb{K}_m} + \underline{c_{m+1} r_{m+1}} \rightarrow$$

$$z = z_m + c_{m+1}(r_m - \alpha A P_m) = \underline{z_m + c_{m+1} r_m}_{\in \mathbb{K}_m} - \alpha c_m \underline{A P_m}_{\in \mathbb{K}_m} \in \mathbb{K}_{m+1}$$

检验空间$\text{span}\{r_0, r_1, ..., r_{k-1}\}$ 和 $\mathbb{K}_k$中的元素可以相互表出即可。

$\mathbb{K}_m\left(A, v\right)$的生成很容易，且满足

$$\mathbb{K}_m \subset \mathbb{K}_{m+1}$$

由计算特征值的幂法知道 $A^j v$ 数值上很快就趋于主特征向量， 即krylov矩阵

$$K_m = \left[v \middle| Av \middle| \cdots \middle| A^{m-1} v\right]$$

是极其病态的。 因此，即使 $\{v, Av, ..., A^j v, ...\}$彼此线性无关，应用时也应当做正交投影. Arnoldi算法是生成 krylov子空间正交基的常用方法。

5.1 Arnoldi 方法

第五章已对K_n的完全正交化(FOM)进行了介绍. 对于大型稀疏矩阵, 一般情况下$m \ll n$, 此时Arnoldi算法同样可以给出子空间$\mathbb{K}_m$的一组正交基。

对K_m做非完全QR分解

$$K_m = QR = Q * \left[r_{11} e_1 \,\middle|\, R_1 \right]$$

其中$Q \in \boldsymbol{R}^{n\times m}$为正交矩阵， 其列向量是$\mathbb{K}_m$的一组正交基； $R \in \mathbb{R}^{m\times m}$是上三角矩阵, e_1表示单位矩阵的第一列， $R_1 \in \mathbb{R}^{m\times(m-1)}$是Hessenberg矩阵。 由$K_m$定义和上式可得

$$A * K_m = \left[K_m(:, 2:m) \,\middle|\, A^m * v \right] = A * Q * R = \left[Q * R_1 \,\middle|\, A^m * v \right] \tag{7}$$

我们分两种情况讨论上式

1) 若K_m 是个不变自空间， 即$v_m = A^m v \in \mathbb{K}_m = \mathrm{span}(Q)$, 则

$$v_m = A^m v = Q * t, \qquad t \in \mathbb{R}^m$$

进而

$$A * Q * R = \left[Q * R_1 \,\middle|\, v_m \right] = Q * \left[R_1 \,\middle|\, t \right] = Q * H_0$$
$$A * Q = Q * H_0 * R^{-1} = Q * H$$

其中H_0, $H \in \mathbb{R}^{m\times m}$是Hessenberg矩阵[6].

2) 若$v_m = A^m * v \notin (Q)$, 考虑(7)式的前m列,

$$A * Q * R = \left[Q * R_1 \,\middle|\, v_m \right] \quad \rightarrow \quad AQ * R_{(1:m+1,1:m)} = Q * R_1$$

注意到R是上三角， 删去最后一列， $R(:, 1:m)$的最后一行全为零， 则

$$Q * R_{(:,1:m)} = Q_{(:,1:m)} * R_{(1:m,1:m)}$$

进而

$$AQ_{(:,1:m)} * R_{(1:m,1:m)} = Q * R_1 \quad \rightarrow \quad AQ_{(:,1:m)} = Q * H$$

其中$H = R_1 * R^{-1}_{(1:m,1:m)} \in \mathbb{R}^{(m+1)\times m}$. 此时前一种情况略有不同， H不是方阵; 等式两端的Q也略有不同. 可改写上式为

$$AQ_{(:,1:m)} = Q * H = \left[Q_{(:,1:m)}, q_{m+1} \right] \begin{bmatrix} H_{(1:m,1:m)} \\ h_{m+1,m} * e_m^{\mathrm{T}} \end{bmatrix} = Q_{(:,1:m)} * H_{(:,1:m)} + \vec{w} * e_m^{\mathrm{T}}$$

其中$\vec{w} = h_{m+1,m} * q_{m+1}$, e_m是单位矩阵I_m的最后一列.

[6] 上三角的逆是上三角， 上三角乘以Hessenberg仍然是Hessenberg矩阵

Arnoldi的方法给出了$\boldsymbol{O}(n)$运算量的正交化过程，前提是A是稀疏矩阵且$m \ll n$, 将算法产生的单位列向量组装为矩阵

$$Q_{m+1} = \left[q_1, q_2, ..., q_m, q_{m+1}\right]_{n\times(m+1)}$$

$H \in \mathbb{R}^{(m+1)\times m}$为算法产生的上Hessenberg矩阵, H_-为删去最后一行的m阶方阵， 则成立

$$AQ_m = Q_{m+1}H = Q_m H_- + h_{m+1,m} q_{m+1} * e_m^T \tag{8}$$

$$Q_m^T A Q_m = H_- \tag{9}$$

若算法过程中产生了不变子空间, 即 $v_m \in span\{Q_m\}$, 则(8)中$h_{m+1,m} = 0$, 且

$$A * Q_m = Q_m * H_-$$

此时(9)成立. 若算法执行过程中未产生不变子空间，此时上式不成立。 由q_k的正交性，(8)两端同时左乘Q_m^{T}, (9)依然成立.

考虑待求解问题 $Ax = b$, 给定初值猜测x_0, 进而可得残量$r_0 = b - Ax_0$, 我们尝试在Krylov子空间

$$\mathbb{K}_m\left(A, r_0\right) = span\{v_0, Av_0, A^2 v_0, ..., A^{m-1} v_0\}, \quad v_0 = r_0 / \|r_0\|$$

中寻找近似解并提出要求

$$x_m = x_0 + z_m \underline{\underline{\in \mathbb{K}_m}}, \quad \& \quad r_m = b - Ax_m \perp \mathbb{K}_m$$

记Arnoldi算法产生的一组正交基Q_m, Hessenberg 矩阵 H_m, $z_m = Q_m * t$. 由正交性可得

$$x_m = x_0 + Q_m t \Rightarrow r_m = r_0 - AQ_m t \Rightarrow H_m t = Q_m^{\mathrm{T}} r_0 = \left[q_1^{\mathrm{T}} * r_0, 0, ..., 0\right]^{\mathrm{T}}$$

于是，x_m 可以按照如下方式得到

$$\begin{cases} x_m = x_0 + Q_m t \\ t = H_m^{-1}\left(\beta e_1\right) \end{cases}$$

其中 $\beta = \|r_0\|$, $e_1 = [1, 0, 0, ...0]^{\mathrm{T}}$

第五章给出的Arnoldi算法是基于basic Gram-Schmidt方法，当A的条件数不坏时，可以使用。 当A 是病态矩阵时，basic Gram-Schmidt方法数值不稳定，可采用修正的Gram-Schmidt方法 myArnoldi_MGS.m 以下是测试代码，读者可选择不同的子空间维度测试精度的不同，也可选择调整主对角元素的大小以 改变矩阵的条件数，观察精度的变化.

```
m = 22;     rng(0);     A = rand(m);
A = 5*speye(m) - ( tril(A,3) - tril(A,-2));
figure; spy(A)
r = ones(m,1);     % -- test problem
```

```
x0 = zeros(m,1);  r0 = r - A*x0;   e0 = norm(r0)
% % FOM when choose n = m
n = 10;     % choose n = 5, 10, 15, 20 and compare
[Q,H,k] = myArnoldi_MGS(A,r0,n);
bet = norm(r0);   e1 = 0*H(:,1);   e1(1) = bet;
% %      solve by Arnoldi
t = H \ e1;
x1 = x0 + Q(:,1:k)*t;
% % check new residual
r1 = r - A*x1;  e1 = norm(r1)
```

```
function [Q,H,k] = myArnoldi_MGS(A,v,n)
% % Arnoldi modified Gram-Schmidt method generate A,Q,H
% Full Orthogonalization Method: A*Q = Q*H
% for example: n = 5; A = rand(n); v = ones(n,1);
%      [Q,H] = myArnoldi(A,v);  norm(A*Q - Q*H), norm(Q'*Q-eye(n))
m = size(A,1);
Q = zeros(m,n);     H = zeros(n+1,n);
v = v./norm(v);     Q(:,1) = v;
for k = 1:n
    w = A*Q(:,k);
    for i = 1:k
        v = Q(:,i);
        H(i,k) = w'*v;
        w = w - H(i,k)*v;
    end
    H(k+1,k) = norm(w,2);
    Q(:,k+1) = w / H(k+1,k);
    if H(k+1,k) < 1e-12
        Q = Q(:,1:k); H = H(1:k,1:k);
        break
    end
end
```


标注 8 当矩阵A极其病态时（例如Hilbert矩阵），利用modified Gram-Schmidt 实现Arnoldi过程的误差依然很大；此时需要采用更稳定的数值方法，例如：double orthogonalization 和 Householder变换。基于Gram-Schmidt 的double orthogonalization 算法简单，容易理解；Householder变换的算法实现相对复杂，不直观. 进一步的讨论可参考[7]。一般情况下，double orthogonalization 就足够了，重复多次的正交化似无必要. Householder变换是一类应用非常广泛的工具，计算矩阵特征值的章节会再次提到。

[7]Saad, Iterative methods for large sparse linear systems. 2003, SIAM

5.2 Gmres算法简介

当m 很小时，Arnoldi算法是$\boldsymbol{O}(n)$的，但是随着m增加，系数会以m^2 递增，要当心！ 因此有一些改进的算法（主要就是为了避免m变的太大). 例如：固定$m=5$, 反复利用上述过程更新， restart FOM(Full Orthogonalization Method). 类似的可以导出著名的GMRES方法， 只是计算z_m 在 $\mathbb{K}_m$空间 中坐标t时，利用$(m+1)\times m$阶的Hessenberg矩阵的信息，使残量满足 $r_m \perp Q_{m+1}$. 于是子空间迭代解x_m 按照下式给出.

$$\begin{cases} x_m = x_0 + Q_m t \\ \min\limits_t \left\| H_{(m+1)\times m} t - (\beta e_1) \right\|_2 \end{cases}$$

实际应用时，迭代法都是和预处理结合使用的. Gmres结合非完全lu分解是普遍使用的算法 读者可测试下段代码，体会预处理的效率(若删去代码中L,U的部分，即是不使用预处理的germs方法， 此时收敛会慢很多)。

```
m = 211;     rng(0);
A = 5*speye(m) + sprandn(m,m,0.05);
figure; spy(A)
r = ones(m,1);     % -- test problem
x0 = zeros(m,1);  r0 = r - A*x0;    e0 = norm(r0)
% % test ilu + gmres for sparse matrix
[L,U] = ilu(A); b = r; m = 5; tol = 1e-9; maxI = 12;
[x1,kk,bet] = mygmres0(A,b,m,tol,maxI,L,U,x0);
r1 = r - A*x1;  e1 = norm(r1)
```

```
function [x0,k,bet] = mygmres0(A,b,m,tol,maxI,L,U,x0)
%  test code : ilu + gmres
for k = 1:maxI
     r0 = b - A*x0;  r0 =  U \ (L \ r0);    bet = norm(r0);
    if bet < tol % check residual,
        break
    end
    [V,H] = myArnoldi2(A,L,U,r0,m);    b0 = H(:,1)*0;  b0(1) = bet;
    y =  H \ b0; x0 = x0  + V(:,1:m)*y;
end
 fprintf('\n gmres(%d) stop at step = %d, and residual = %e\n',m,k,bet);
end
%  ==== combine with modified Gram Schmidt
function [Q,H,k] = myArnoldi2(A,L,U,v,n)
% % Arnoldi modified Gram-Schmidt method generate U\L\A,Q,H
m = size(A,1);
Q = zeros(m,n);     H = zeros(n+1,n);
v = v./norm(v);     Q(:,1) = v;
```

```
for k = 1:n
    w = A*Q(:,k);  w = U\(L\w);
    for i = 1:k
        v = Q(:,i);
        H(i,k) = w'*v;
        w = w - H(i,k)*v;
    end
    H(k+1,k) = norm(w,2);
    Q(:,k+1) = w / H(k+1,k);
    if H(k+1,k) < 1e-12
        Q = Q(:,1:k); H = H(1:k,1:k);
        break
    end
end
end
```

上机练习

1. 分别利用Jacobi, Gauss-Seidel, SOR迭代法 计算$Ax=b,\quad n=15$并画出向量x的图像

$$\begin{pmatrix} \frac{5}{2} & -1 & 0 & \cdots & 0 & -1 \\ -1 & \frac{5}{2} & -1 & 0 & \cdots & 0 \\ & \ddots & \ddots & \ddots & & \\ -1 & 0 & \cdots & 0 & -1 & \frac{5}{2} \end{pmatrix}_{n\times n} * \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{n\times 1}$$

2. 尝试自己写cg和germs算法程序计算上述例子，体会计算效率的不同。

6 实用迭代法小结(略)

1. 共轭梯度法适用于对称正定矩阵，但是矩阵条件数不好时收敛慢
2. 实用时多采用PCG，预处理共轭梯度法

   $$Ax = b \rightarrow M^{-1}Ax = M^{-1}b$$

   若需要一次求解多个方程，建议自己写程序 pcg3.m
3. 若矩阵不严格对称，但是接近对称，PCG依然有效
4. 若矩阵对称性差，子空间方法如GMRES效果更好(需要结合预处理)
5. 预处理时需要近似矩阵，长得像不一定真的是近似。例如A是二阶差分矩阵，B在A的基础上将中间部分行全设置为零， 如wetting transition 例子所涉及的rough surface。矩阵 inv(A)*B的条件数很差，不适合作为预处理。

~~我的理解(不一定对！！)~~共轭梯度法为下山找极小的过程，方向略有偏差依然可以下山。对称正定的要求主要是理论需要！

7 关于预处理和多重网格方法(略)

预处理是一个大的课题。对于一般问题，特别是当问题来源不明时，可以选择

$$pcg \Big/ gmres + ilu$$

该处理方式可以部分提高计算效率，也被广泛采用。

实际上，很多大型稀疏矩阵求解的问题来源于PDE数值格式，以Poisson方程为例

$$-\Delta u = f, \quad (x,y) \in [0,1]^2$$

如果是一维问题，则Poisson方程退化为一个 常微分方程两点边值问题

$$-u'' = f, \quad x \in (0,1)$$

对计算区域进行$(N+1)$等分，记$T_N = toeplitz([-2,1,zeros(1,N-2)])$为二阶差分矩阵， 即

$$T = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}$$

求解上述ODE两点边值问题转化为计算

$$T^h * u^h = b^h$$

该线性系统是常系数、三对角的，计算方法可以选择：Thomas（LU分解）、sine transform；简单高效。本无必要使用迭代法。

为了阐明一类重要的方法（Multigrid - Method），我们以一维问题介绍算法，进而推广到二维问题。

Multigrid method 的算法过程如下

AGM-(1) Iterate on $A_h u = b_h$ to reach u_h (say 3 Jacobi or Gauss-Seidel steps).

AGM-(2) Restrict the residual $r_h = b_h - A_h u_h$ to the coarse grid by $r_{2h} = R_h^{2h} r_h$.

AGM-(3) Solve $A_{2h} E_{2h} = r_{2h}$ (small size problem)

AGM-(4) Interpolate E_{2h} back to $E_h = I_{2h}^h E_{2h}$. Add E_h to u_h.

AGM-(5) Iterate 3 more times on $A_h u = b_h$ starting from the improved $u_h + E_h$.

上述算法中重要的是限制矩阵和插值矩阵，Gilbert Strang教材上给出一个基于线性插值的限制矩阵R，简单有效！

Interpolation $\boldsymbol{I v} = \boldsymbol{u}$

$\boldsymbol{u}$ on the fine ($\boldsymbol{h}$) grid from

$\boldsymbol{v}$ on the coarse (**2h**) grid

values are the $\boldsymbol{u}$'s.

$$\frac{1}{2}\begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} v_1/2 \\ v_1 \\ v_1/2 + v_2/2 \\ v_2 \\ v_2/2 + v_3/2 \\ v_3 \\ v_3/2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix}$$

插值矩阵和限制矩阵是成对的

Full weighting $\boldsymbol{R u} = \boldsymbol{v}$

Fine grid $\boldsymbol{u}$ to coarse grid $\boldsymbol{v}$

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Matlab code : GS_multigrid_1D.m

```
clear, syms z
u_ex =  sin(3*pi*z).*exp(z);  f_ex = diff(u_ex,2);
f_ex = matlabFunction(-f_ex);
u_ex = matlabFunction( u_ex);
% %  ==== gauss seidel method with multigrid precondition
```

```
n1 = 63;  h1 = 1/(n1+1);    x1 = ( h1 : h1 : 1-h1 ).';
A1 = toeplitz([2,-1,zeros(1,n1-2)]); A1 = sparse(A1)/h1^2;
b = f_ex(x1);
b(1) = b(1) - A1(1,2)*u_ex(0);
b(n1) = b(n1) - A1(n1,n1-1)*u_ex(1);
tic, u_dir = A1 \ b; toc,
absErr_dir = norm(u_ex(x1) - u_dir,inf),
figure(1);  fplot(u_ex,[0,1],'r-');  hold on,  plot(x1,u_dir,'ko')
% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % coef mat on coarse mesh
n2 = floor(n1/2); h2 = 2*h1;
A2 = toeplitz([2,-1,zeros(1,n2-2)]);  A2 = sparse(A2) / h2^2;
tic, dA2 = decomposition(A2); toc,
%  %%%%%%%%%%%%%%%%%%%%%%%%%
%  interp matrix
moj = [1;2;1];
IR = zeros(n1,n2);
for k = 1:n2
    IR(2*k-1:2*k+1,k) = moj;
end
IR = 1/2*sparse(IR(1:n1,1:n2));
% % restrict matrix
RI = 1/2*IR.';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% % % GS - mg
u0  = zeros(length(b),1);
maxI = 22;  Tol = 1e-9;  norm_b = norm(b);
L = tril(A1); U = A1 - L;  r0 = b;
for k = 1:maxI
    u_new = L \ (b - U*u0); % gauss seidel
%    correct by corase mesh
    if mod(k,3) == 0
    r0 = b - A1*u_new; % new residual
    z0 = RI*r0;  % restrict on the coarse mesh
    p0 = dA2 \ z0;  % solve new residual
    q0 = IR*p0;  % interp on the fine mesh
    u_new = u_new  + q0;  % update new soln
    end
  epn = norm(r0) / norm_b;  % check stop
    if epn < Tol
 fprintf('\n gs-mg stop at itr = %d, and res = %e\n',k,epn);
       break
```

```
    end
    u0 = u_new;
end
```

MG方法最大的特点和优势是随着网格加密，迭代次数不增加！纯粹一维问题的计算无需使用mg，直接法即可。

7.1 二维问题代码

下面给出二维问题两层网格的代码，算法过程基本一样。对于测试问题，当网格加密，迭代次数不增加。因为粗网格是精确求解的，加密后迭代次数反而略低。如果需要反复求解的问题，需要测试两种方法的效率：一是预先对系数矩阵进行decomposition，二是每步进行MG预处理迭代；特别是两层网格的计算，如果n很大（n = 1024），那么粗网格上的计算时效也要考虑。另外，迭代初值对收敛性有较大影响！

```
 % %  ==== gauss seidel method with multigrid precondition
n1 = 7999;  h1 = 1/(n1+1);    x1 = ( h1 : h1 : 1-h1 ).';
[xx,yy] = meshgrid(x1);
% --- test problem
Uex = sin(3*pi*xx).*exp(yy);
A1 = toeplitz([-2,1,zeros(1,n1-2)]); A1 = -sparse(A1)/h1^2;
bigA = kron(speye(n1),A1) + kron(A1,speye(n1));
b = bigA*Uex(:);
tic, dA = decomposition(bigA); toc,
tic, u_dir = dA \ b(:);  toc,
% u_dir = reshape(u_dir,n1,n1);
% figure(1); mesh(xx,yy,u_dir);  axis auto,
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% coef mat  on coarse mesh
n2 = floor(n1/2); h2 = h1*2;
A2 = toeplitz([-2,1,zeros(1,n2-2)]);  A2 = -sparse(A2) / h2^2;
middleA = kron(speye(n2),A2) +  kron(A2,speye(n2));
tic, dA2 = decomposition(middleA); toc,
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% coef mat  on coarse mesh
n3 = floor(n2/2); h3 = h2*2;
A3 = toeplitz([-2,1,zeros(1,n3-2)]);  A3 = -sparse(A3) / h3^2;
smallA = kron(speye(n3),A3) +  kron(A3,speye(n3));
tic, dA3 = decomposition(smallA); toc,
%  %%%%%%%%%%%%%%%%%%%%%%%%%
[IR,RI] = get_intp_rest_2d(n1,n2);
[IR2,RI2] = get_intp_rest_2d(n2,n3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
u_old  = zeros(length(b),1);
maxI = 15;  Tol = 1e-9;  norm_b = norm(b);
L = tril(bigA); U = bigA - L;  r0 = b;
%
mL = tril(middleA); mU = middleA - mL;
%
tic
for k = 1:maxI
    u_new = L \ (b - U*u_old); % gauss seidel
%
    if mod(k,3) == 0
%     r0 = b - bigA*u_new; % new residual
    r0 = U*(u_old - u_new);
    z0 = RI*r0;  % restrict on the coarse mesh
    p0 = dA2 \ z0;  % solve new residual'
%     p0 = GS_mg0(z0,mL,mU, RI2,IR2, dA3);
    q0 = IR*p0;  % interp on the fine mesh
    u_new = u_new  + q0;  % update new soln
    end
    epn = norm(r0) / norm_b;  % check stop
    if epn < Tol
 fprintf('\n gs-mg  stop at itr  = %d, and absErr = %e\n',k,epn);
        break
    end
    u_old = u_new;
end
toc,
u_mg = reshape(u_old,n1,n1);
figure(2); pcolor(xx,yy,u_mg); axis equal, shading interp,
norm(u_mg(:) - Uex(:), inf)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [IR,RI] = get_intp_rest_2d(n1,n2)
% linear interp matrix
IR = zeros(n1,n2);   moj = [1;2;1];
for k = 1:n2
    IR(2*k-1:2*k+1,k) = moj;
end
IR = 1/2*sparse( IR(1:n1,1:n2) );
% % restrict matrix
RI = 1/2*IR.';
```

```
% %%%%%% Restrict and Interp Mats in 2D
IR = kron(IR,IR);    RI = kron(RI,RI);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x0] = GS_mg0(b,L,U,RI,IR, dA2)
    x0 = 0*b;
    for j = 1:9
        x1 =  L \ (b - U*x0);
        if mod(j,3) == 0
%           r0 = b - A*x0;
         r0 = U*(x0 - x1);
         z0 = RI*r0;
         p0 = dA2 \ z0;
         zp0 = IR*p0;
         x1 = x1 + zp0;
        end
        x0 = x1;
    end
end
```

测试结果表明:

1. 二维Poisson问题, 4000^2网格数，矩阵分解直接法比多重网格优势！主要是稀疏矩阵的直接法太强大！！6000^2网格数，直接法预先分解和2/3层时效差不多！4层网格略优！超过8000^2的网格，二层网格几乎不能算；可考虑多层网格。
2. 3D问题，超过100等分的网格直接法基本放弃！
3. 3D问题，如网格数128^3，两层网格效率不如三层或者四层网格，矩阵分解64^3基本可行，但不那么明显高效了！
4. 3D问题，如网格数100^3, 两层网格+粗网格稀疏矩阵分解直接法，效率不弱！不必要三层以上网格
5. 3D问题，如网格数400^3很大，三层以下网格无法计算，要选择四层或以上；四层网格每步迭代耗时15秒（2020MacBook16‘）；6千万的矩阵，规模太大了，仅仅是可以算，太耗时了，等不起！！
6. 3D问题，测试显示：pcg + ilu 可行不优秀，pcg + mg 几乎不可行，GD + mg 效果可以，ilu + mg 优秀
7. 3D问题，测试结果：ilu + mg 对poisson、特别是双调和ch表现优异！！！而对cahn-hilliard求解：pcg+ilu, GD+mg, GS + mg收敛较慢，ilu + mg 比较好！

对角占优矩阵性质

记$A=(a_{ij})_{n\times n}$, 存在$\delta>0$ 满足

$$|a_{ii}|\geq\sum_{j\neq i}|a_{ij}|+\delta,\quad 1\leq i\leq n$$

称A为严格对角占优阵.

(1) A 可逆

(2) $\|A^{-1}\|=\sup\limits_{x\neq0}\dfrac{\|A^{-1}x\|}{\|x\|}=\sup\limits_{y\neq0}\dfrac{\|y\|}{\|Ay\|}$

(3) $\|A^{-1}\|_\infty\leq\dfrac{1}{\delta}$

(4) $\forall\lambda\in\mathrm{Sp}(A)$, 有$|\lambda|\geq\delta$.

(1) 反证法；假设A不可逆，则存在$x\in\mathbb{R}^n,\ x\neq0,s.t.,Ax=0$. 记$|x_{i_0}|=\|x\|_\infty$, 挑出第 i_0个方程，推出矛盾即可。

(2) 由于A 可逆，则集合

$$W=\left\{\frac{\|A^{-1}x\|}{\|x\|}\,\middle|\,x\in\mathbb{R}^n\right\}$$

与

$$W'=\left\{\frac{\|y\|}{\|Ay\|}\,\middle|\,y\in\mathbb{R}^n\right\}$$

相等[8]。这一结论给出了可逆矩阵的 从属范数的又一定义。

(3) 由上一结论[9]， 只要证明$\frac{1}{\delta}$是集合W'的上界即可， 其中向量范数取∞范数。即只要证明:

$$\frac{\|y\|_\infty}{\|Ay\|_\infty}\leq1/\delta\Leftrightarrow\delta\|y\|_\infty\leq\|Ay\|_\infty.$$

对$\forall y\in\mathbb{R}^n,y\neq0$, 记$|y_{i_0}|=\|y\|_\infty$, 则

$$\|Ay\|_\infty=\max_i\left|\sum_{j=1}^n a_{ij}y_j\right|\geq\left|\sum_{j=1}^n a_{i_0,j}y_j\right| \tag{10}$$

$$\geq|a_{i_0,i_0}y_{i_0}|-\sum_{j\neq i_0}^n|a_{i_0,j}y_j|\geq|a_{i_0,i_0}y_{i_0}|-\sum_{j\neq i_0}^n|a_{i_0,j}y_{i_0}| \tag{11}$$

$$=\left(|a_{i_0,i_0}|-\sum_{j\neq i_0}^n|a_{i_0,j}|\right)|y_{i_0}|\geq\delta\|y\|_\infty \tag{12}$$

也即$1/\delta$是 $\|y\|_\infty/\|Ay\|_\infty$的一个上界，上界大于等于上确界， $\|A^{-1}\|_\infty\leq1/\delta$

(4) 设

$$Ax=\lambda x,\ x\in\mathbb{R}^n,\ x\neq0$$

[8]检验W 与W'相互包含即可。

[9]Varah, A Lower Bound for the Smallest Singular Value of a Matrix, Linear Algebra and Its Applications 11,3-5 (1975)

记$|x_{i_0}| = \|x\|_\infty$, 考虑$Ax = \lambda x$ 的第i_0个方程 $\sum_{j=1}^{n} a_{i_0,j}x_j = \lambda x_{i_0}$ 进而

$$|a_{i_0,i_0}x_{i_0}| = \left|\lambda x_{i_0} - \sum_{j\neq i_0} a_{i_0,j}x_j\right| \tag{13}$$

$$\leq |\lambda|\cdot|x_{i_0}| + \sum_{j\neq i_0}|a_{i_0,j}x_j| \tag{14}$$

$$\leq |\lambda|\cdot|x_{i_0}| + \sum_{j\neq i_0}|a_{i_0,j}|\cdot|x_{i_0}| \tag{15}$$

消去$|x_{i_0}|$, 即得到

$$\delta \leq \left(|a_{i_0,i_0}| - \sum_{j\neq i_0}|a_{i_0,j}|\right) \leq |\lambda|$$

标注 9 结论(4)说明 A没有零特征值，同样说明A可逆。 另外，证明A可逆还有其它方法。例如，令 $A = D + F$, 则

$$A = D(D^{-1}F + I)$$

为两矩阵乘积. 分 析可知 D可逆，且$\rho(D^{-1}F) < 1$ 以及 $(D^{-1}F + I)$可逆，进而利用”可逆矩阵的乘积仍为可逆阵” 即得到结论.

定义 5 (可约/分矩阵) 设存在排列矩阵P, 使得 $PAP^\top$ 分块三角形矩阵，也即

$$PAP^\top = \begin{bmatrix} A_{11} & 0 \\ A_{12} & A_{22} \end{bmatrix}$$

且分块A_{22}是方阵，则称A为可约/分矩阵(reducible matrix); 否则称之为不可约矩阵. 等价定义: 记$\mathscr{W} = \{1, 2, ..., n\}$, 若存在指标集合 $\mathscr{S}, \mathscr{T}$ 满足

$$\mathscr{S} \cup \mathscr{T} = \mathscr{W}, \qquad \mathscr{S} \cap \mathscr{T} = \emptyset$$

且

$$a_{ij} = 0, \quad i \in \mathscr{S},\ j \in \mathscr{T},$$

则称A可约.

不可约的弱对角占优矩阵A ，

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

若至少一行是严格占优的，则 A可逆. 类似于严格对角占优矩阵的反证法证明。假设$Ax = 0$, $\|x\|_\infty = 1$, 记

$$\mathscr{S} = \{i : |x_i| = 1\},\ \mathscr{T} = \{k : |x_k| < 1\}$$

若$\mathcal{T}=\emptyset$, 也即x为常值向量，找到严格占优行推出矛盾 。 于是$\mathcal{T}$不是空集，也即是存在指标k使得$|x_k|<1$. 另一方面，由于A不可约，存在指标$i\in\mathcal{S}$ 使得[10] $a_{ik}\neq 0$. 考虑$Ax=0$的第i行，

$$|a_{ii}|\leq \sum_{j\in\mathcal{S},j\neq i}|a_{ij}||x_j|+\sum_{j\in\mathcal{T}}|a_{ij}||x_j|<\sum_{j\in\mathcal{S},j\neq i}|a_{ij}|+\sum_{j\in\mathcal{T}}|a_{ij}|$$

与弱对角占优矛盾！

标注 10 弱对角占优矩阵，若没有不可约的条件，则不能推出可逆， 例如

$$A=\begin{bmatrix}0 & 0\\ 1 & 2\end{bmatrix}$$

有一行严格占优，但明显不可逆。 另一方面，不可约又不是必要条件，例如

$$B=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$$

是可约的，弱占优且有一行严格占优，但是B可逆.

[10]如果不然，可通过排列将整个一块全为零，与不可约矛盾！