第七章 求解线性方程组的Krylov子空间方法

殷东生
dyin@math.tsinghua.edu.cn

清华大学数学科学系

2018年秋季学期

经典的迭代法如 Jacobi、Gauss-Seidel 其收敛速度都很慢，而 SOR 的最优松弛因子的确定不是一件轻松的任务。对大型稀疏矩阵，更高效的方法是Krylov子空间方法，其理论基础是Cayley-Hamilton定理。

定义 (零化多项式–annihilation polynomial)

设 $A \in \mathbb{C}^{n\times n}$，若多项式 $g(z)$ 使得 $g(A) = 0$，则称$g(z)$为A的零化多项式。

定理 (Cayley-Hamilton)

设$A \in \mathbb{C}^{n\times n}$，则$A$的特征多项式 $\chi(z)$ 是A 的零化多项式。

设A 的特征多项式$\chi(z)$为：

$$\chi(z) = z^n + c_{n-1}z^{n-1} + \cdots + c_1 z + c_0, \quad c_0 = (-1)^n \det(A).$$

则由定理知若A非奇异，则$c_0 \neq 0$:

$$\chi(A) = A^n + c_{n-1}A^{n-1} + \cdot + c_1 A + c_0 I = 0,$$

$$\Longrightarrow \quad A^{-1} = -\frac{1}{c_0}A^{n-1} - \frac{c_{n-1}}{c_0}A^{n-2} + \cdots - \frac{c_1}{c_0}I$$

$$= q_{n-1}(A).$$

亦即A^{-1}可用矩阵A的$n-1$次多项式表示。
则使用迭代法时，任给初值$\boldsymbol{x}_0$，有

$$A\boldsymbol{x}^* - A\boldsymbol{x}_0 = \boldsymbol{b} - A\boldsymbol{x}_0 = \boldsymbol{r}_0$$

$$\boldsymbol{x}^* = \boldsymbol{x}_0 + A^{-1}\boldsymbol{r}_0 = \boldsymbol{x}_0 + q_{n-1}(A)\boldsymbol{r}_0.$$

理论上，可在空间$\mathcal{K} = \left\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \cdots, A^m\boldsymbol{r}_0, \cdots, A^{n-1}\boldsymbol{r}_0\right\}$中找到 方程组$A\boldsymbol{x} = \boldsymbol{b}$的准确解。但在科学与工程计算中的大型稀疏矩阵的阶多为10^6以上，在$\mathcal{K}$中搜索准确解的 计算代价太昂贵。

定义 (Krylov子空间)

由向量$\boldsymbol{r}_0$和矩阵A生成的子空间

$$\mathcal{K}_m = span(\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0),$$

称为m维*Krylov子空间*。

Krylov子空间方法就是用$\mathcal{K}_m$去逼近$\mathcal{K}$，即在Krylov子空间$\boldsymbol{x}_0 + \mathcal{K}_m$中求方程组的近似解$\boldsymbol{x}_m$。从函数逼近的角度看就是求矩阵多项式$q_{n-1}(A)$的最佳$m-1$次逼近多项式$q_{m-1}(A)$使得：

$$\begin{aligned}
\|\boldsymbol{x}^* - \boldsymbol{x}_m\| &= \|\boldsymbol{x}_0 + q_{n-1}(A)\boldsymbol{r}_0 - (\boldsymbol{x}_0 + q_{m-1}(A)\boldsymbol{r}_0)\| \\
&= \|q_{n-1}(A)\boldsymbol{r}_0 - q_{m-1}(A)\boldsymbol{r}_0\| \\
&= \min_{\boldsymbol{x}\in\boldsymbol{x}_0+\mathcal{K}_m} \|\boldsymbol{x}^* - \boldsymbol{x}\| \\
&= \min_{p\in\mathcal{P}_{m-1}(A)} \|q_{n-1}(A)\boldsymbol{r}_0 - p(A)\boldsymbol{r}_0\|
\end{aligned}$$

方法的关键是在那种范数下的最佳逼近？自然的选择是2-范数：

$$\|\boldsymbol{x}\|_2 = \sqrt{(\boldsymbol{x},\boldsymbol{x})}$$

亦即在Krylov子空间$\boldsymbol{x}_0 + \mathcal{K}_m$中求方程组的准确解$\boldsymbol{x}^*$的最佳平方逼近：

$$\|\boldsymbol{x}^* - \tilde{\boldsymbol{x}}_m\|_2 = \min_{x \in \mathcal{K}_m} \|\boldsymbol{x}^* - \boldsymbol{x}\|_2.$$

则由最佳平方逼近的性质知：

$$(\boldsymbol{x}^* - \tilde{\boldsymbol{x}}_m, \boldsymbol{y}) = 0, \quad \forall \boldsymbol{y} \in \mathcal{K}_m$$

设$V_m = (\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_m)$是$\mathcal{K}_m$的一组基，则

$$\tilde{\boldsymbol{x}}_m = V_m \boldsymbol{y}_m \Longrightarrow V_m^T V_m \boldsymbol{y}_m = V_m^T \boldsymbol{x}^*, \quad \text{法方程}.$$

但是$\boldsymbol{x}^*$未知，直接求自然的2-范数下的最佳平方逼近不可行。

对称正定阵

若$A \in \mathbb{R}^{n\times n}$正定对称，则

$$(Ax, x) = x^T Ax = (x, x)_A = \|x\|_{2,A}^2$$

定义了一种内积，同时也定义了相应的2-范数。
求此2-范数下在$\mathcal{K}_m$中的最佳平方逼近，则其近似解x_m满足：

$$\begin{aligned} & (x^* - x_m, y)_A = (A(x^* - x_m), y) \\ = \quad & (b - Ax_m, y) = (r_m, y) = 0, \quad \forall y \in \mathcal{K}_m. \end{aligned}$$

最后一个等式说明

$$r_m \perp \mathcal{K}_m. \text{ Petrov-Galerkin条件}$$

设$V_m = (v_1, \cdots, v_m)$是$\mathcal{K}_m$的一组基，则$x_m = V_m y_m$，此时法方程

$$V_m^T A V_m y_m = V_m^T b.$$

有唯一解。

一般矩阵

若$A \in \mathbb{R}^{n\times n}$非奇异，不对称正定，则可定义内积和2-范数：

$$(\boldsymbol{x},\boldsymbol{x})_A = (A\boldsymbol{x},A\boldsymbol{x}) = \boldsymbol{x}^T A^T A\boldsymbol{x} = \|\boldsymbol{x}\|_2^2.$$

此2-范数下在$\mathcal{K}_m$中的最佳平方逼近，则其近似解$\boldsymbol{x}_m$满足：

$$\begin{aligned} & (\boldsymbol{x}^* - \boldsymbol{x}_m, \boldsymbol{y})_A = (A(\boldsymbol{x}^* - \boldsymbol{x}_m), A\boldsymbol{y}) \\ = \quad & (\boldsymbol{b} - A\boldsymbol{x}_m, A\boldsymbol{y}) = (\boldsymbol{r}_m, A\boldsymbol{y}) = 0, \quad \forall \boldsymbol{y} \in \mathcal{K}_m. \end{aligned}$$

由上式可知：

$$\boldsymbol{r}_m \perp A\mathcal{K}_m.$$

设$V_m = (\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m)$是$\mathcal{K}_m$的一组基，则$\boldsymbol{x}_m = V_m \boldsymbol{y}_m$，此时法方程

$$V_m^T A^T A V_m \boldsymbol{y}_m = V_m^T A^T \boldsymbol{b}.$$

有唯一解。

病态基底

上面两种逼近方法的关键在于$\mathcal{K}_m$基底V_m的选取。因为

$$\mathcal{K}_m = span\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0\}$$

如果$\boldsymbol{r}_0, A\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0$线性无关，则可选取

$$V_m = (\boldsymbol{r}_0, A\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0)$$

但是，由求解特征值的幂法知，当i较大时，

$$A^i\boldsymbol{r}_0 \to \lambda^i\boldsymbol{u}.$$

这里λ是A的主特征值，$\boldsymbol{u}$是其对应的主特征向量。所以V_m的列收敛到A的主特征向量的倍数，则V_m的列逐渐成为$\mathcal{K}_m$的病态基底。为了克服这些困难，可对$(\boldsymbol{r}_0, A\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0)$进行Gram-Schmidt正交化来得到$\mathcal{K}_m$的一组正交基，这就是Arnoldi方法。

Arnoldi方法

Arnoldi方法是针对非对称矩阵的正交投影方法。在1951年时为了把稠密矩阵通过正交变换化为Hessenberg矩阵，然后求解特征值。
Arnoldi 过程是构造Krylov子空间的一组正交基的过程，
是Gram-Schmidt正交化的一种变形。设Krylov子空间为

$$\mathcal{K}_m(\boldsymbol{r}_0, A) = \text{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0\}$$

记

$$K_m = \left[\begin{array}{c|c|c|c} & & & \\ & & & \\ \boldsymbol{r}_0 & A\boldsymbol{r}_0 & \vdots & A^{m-1}\boldsymbol{r}_0 \\ & & & \\ & & & \end{array}\right] \in \mathbb{R}^{n\times m} \quad \text{Krylov 矩阵}$$

Arnoldi 迭代过程：选$\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$，然后对$A\boldsymbol{v}_j$进行正交化

$$\begin{aligned} \boldsymbol{w}_j &= A\boldsymbol{v}_j - (A\boldsymbol{v}_j, \boldsymbol{v}_1)\boldsymbol{v}_1 - (A\boldsymbol{v}_j, \boldsymbol{v}_2)\boldsymbol{v}_2 - \cdots - (A\boldsymbol{v}_j, \boldsymbol{v}_j)\boldsymbol{v}_j; \\ \boldsymbol{v}_{j+1} &= \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|_2}, \quad j = 1, 2, \cdots. \end{aligned}$$

定义

$$h_{ij} = (A\boldsymbol{v}_j, \boldsymbol{v}_i), \ i < j \ \text{和} \ h_{j+1,j} = \|\boldsymbol{w}_j\|_2$$

若$\boldsymbol{w}_j \neq 0$，则由 $\boldsymbol{w}_j \perp \text{span}\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_j\}$得

$$\begin{aligned} h_{j+1,j}^2 &= (\boldsymbol{w}_j, \boldsymbol{w}_j) = (A\boldsymbol{v}_j - \sum_{i=1}^{j} h_{ij}\boldsymbol{v}_i, \boldsymbol{w}_j) \\ &= (A\boldsymbol{v}_j, \|\boldsymbol{w}_j\|_2\boldsymbol{v}_{j+1}) = h_{j+1,j}(A\boldsymbol{v}_j, \boldsymbol{v}_{j+1}), \\ \Longrightarrow \|\boldsymbol{w}_j\|_2 &= h_{j+1,j} = (A\boldsymbol{v}_j, \boldsymbol{v}_{j+1}). \end{aligned}$$

算法 (Arnoldi迭代–CGS版本)

1. $\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$
2. $for\ j = 1, 2, \cdots, m\ do$
3. $\quad for\ i = 1, 2, \cdots, j\ do$
4. $\quad$ 计算$h_{i,j} = (A\boldsymbol{v}_j, \boldsymbol{v}_i)$
5. $\quad end\ for$
6. $\quad$ 计算 $\boldsymbol{w}_j = A\boldsymbol{v}_j - \sum_{i=1}^{j} h_{ij}\boldsymbol{v}_i$
7. $\quad h_{j+1,j} = \|\boldsymbol{w}_j\|_2$
8. $\quad if\ h_{j+1,j} = 0\ then\ stop$
9. $\quad \boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
10. $end\ for$

若$\boldsymbol{w}_j = 0$，算法会中断。若$\boldsymbol{w}_j \neq 0$，则迭代可得Arnoldi矩阵

$$V_m = \left[\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \vdots & \boldsymbol{v}_m \\ | & | & & | \\ | & | & & | \end{array}\right] \in \mathbb{R}^{n\times m} \quad \text{Arnoldi 矩阵}$$

和向量$\boldsymbol{v}_{m+1}$，定义上Hessenberg矩阵$\overline{H}_m \in \mathbb{R}^{(m+1)\times m}$，

$$\overline{H}_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1m} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2m} \\ & h_{32} & h_{33} & \cdots & h_{3m} \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \\ & & & & h_{m+1,m} \end{pmatrix}$$

注记

若算法在第m步中断，则$\boldsymbol{w}_m = \boldsymbol{0}$有定义，而$\boldsymbol{v}_{m+1}$无法定义，算法中断！

此时$\overline{H}_m$的最后一行为0，亦即 $h_{m+1,m} = 0$。

引理

假定*Arnoldi*算法在第m步之前不中断，则$\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m$是*Krylov*子空间$\mathcal{K}_m$的一组正交基，即

$$V_m^T V_m = I_{m\times m}.$$

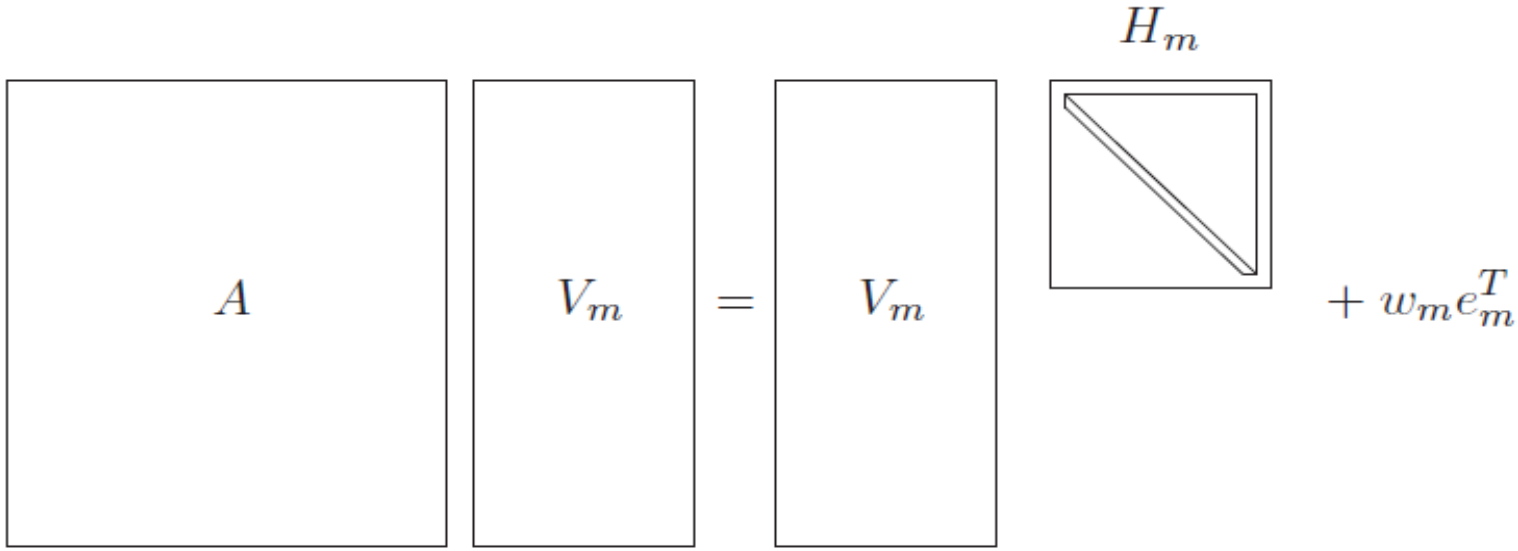
事实上，对$j = 1$有：

$$A\boldsymbol{v}_1 = (A\boldsymbol{v}_1, \boldsymbol{v}_1)\boldsymbol{v}_1 + \boldsymbol{w}_1 = \underbrace{(A\boldsymbol{v}_1, \boldsymbol{v}_1)}_{h_{11}}\boldsymbol{v}_1 + \underbrace{(A\boldsymbol{v}_1, \boldsymbol{v}_2)}_{h_{21}}\boldsymbol{v}_2$$

更一般的，

$$A\boldsymbol{v}_j = \sum_{i=1}^{j} h_{ij}\boldsymbol{v}_i + \boldsymbol{w}_j = \sum_{i=1}^{j+1} h_{ij}\boldsymbol{v}_i, \quad j = 1, \cdots, m.$$

Arnoldi过程可由下图表示：



其中$H_m \in \mathbb{R}^{m\times m}$是去掉$\overline{H}_m$的最后一行得到的，$\boldsymbol{e}_m$是第$m$个单位向量。
写成矩阵形式有

$$AV_m = \left[\begin{array}{c|c|c|c} & & & \\ A\boldsymbol{v}_1 & A\boldsymbol{v}_2 & \vdots & A\boldsymbol{v}_m \\ & & & \end{array}\right] = V_{m+1}\overline{H}_m = V_m H_m + \boldsymbol{w}_m \boldsymbol{e}_m^T$$

定理

设$\overline{H}_m, V_m, H_m$如前定义，则有

$$\begin{aligned} AV_m &= V_m H_m + \boldsymbol{w}_m \boldsymbol{e}_m^T = V_{m+1}\overline{H}_m, \\ V_m^T AV_m &= H_m. \end{aligned}$$

算法 (改进的Gram-Schmidt正交化方法(MGS))

1. $\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$
2. $for\ j = 1, 2, \cdots, m\ do$
3. 初始化$\boldsymbol{w}_j = A\boldsymbol{v}_j$
4. $for\ i = 1, 2, \cdots, j\ do$
5. 计算$h_{ij} = (\boldsymbol{w}_j, \boldsymbol{v}_i)$
6. 更新$\boldsymbol{w}_j = \boldsymbol{w}_j - h_{ij}\boldsymbol{v}_i$
7. $end\ for$
8. $h_{j+1,j} = \|\boldsymbol{w}_j\|_2$
9. $if\ h_{j+1,j} = 0\ then\ stop$
10. $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
11. $end\ for$

Arnoldi 方法的计算量

各种版本的Arnold算法的计算量和存储量如下：

	CGS	MGS	Householder
计算量	$2m^2n$	$2m^2n$	$4m^2n - \frac{4}{3}m^3$
存储量	$(m+1)n$	$(m+1)n$	$(m+1)n - \frac{1}{2}m^2$

注记

*Householder*正交化方法由于其算法的鲁棒性*(robustness)*较好而应用于商业软件和软件包中，尤其是在求解矩阵特征值问题时。但 在求解大型线性方程组时，*MGS*加上重新正交化技巧在大多数情形可以满足要求。

求解线性方程组的Arnoldi方法

用正交投影方法求解线性方程组

$$A\boldsymbol{x} = \boldsymbol{b}, \quad \text{初值 } \boldsymbol{x}_0, \ \boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0.$$

这里逼近子空间$\mathcal{K}_m$:

$$\mathcal{K}_m(A, \boldsymbol{r}_0) = \text{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0\},$$

上述方法在仿射子空间$\boldsymbol{x}_0 + \mathcal{K}_m$中求满足Galerkin条件的近似解$\boldsymbol{x}_m$:

$$\boldsymbol{r}_m \perp \mathcal{K}_m \Longleftrightarrow (\boldsymbol{b} - A\boldsymbol{x}_m) \perp \mathcal{K}_m.$$

在Arnoldi算法中若$\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$，令$\beta = \|\boldsymbol{r}_0\|_2$，则

$$V_m^T \boldsymbol{r}_0 = V_m^T(\beta \boldsymbol{v}_1) = \beta \boldsymbol{e}_1, \ \ \boldsymbol{e}_1 \in \mathbb{R}^m$$

因此，近似解$\boldsymbol{x}_m$如下给出：

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m \boldsymbol{y}_m.$$

这里$V_m = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m] \in \mathbb{R}^{n\times m}$的列向量是$\mathcal{K}_m$的一组基，则有

$$\boldsymbol{b} - A\boldsymbol{x}_m \perp \mathcal{K}_m \Longleftrightarrow \boldsymbol{b} - A(\boldsymbol{x}_0 + V_m \boldsymbol{y}_m) \perp \mathcal{K}_m$$

亦即

$$\begin{aligned} & (\boldsymbol{r}_0 - AV_m \boldsymbol{y}_m, \boldsymbol{v}), \quad \forall \boldsymbol{v} \in \mathcal{K}_m \\ \Longrightarrow \quad & V_m^T A V_m \boldsymbol{y}_m = V_m^T \boldsymbol{r}_0, \quad \text{法方程} \end{aligned}$$

所以

$$\boldsymbol{y}_m = (V_m^T A V_m)^{-1} V_m^T \boldsymbol{r}_0 = H_m^{-1}(\beta \boldsymbol{e}_1).$$

在Arnoldi过程中使用MGS，则可得完全正交化方法(full orthogonalization method–FOM)。

算法 (Full orthogonalization method (FOM))

1. 计算$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, \beta := \|\boldsymbol{r}_0\|_2$和$\boldsymbol{v}_1 := \boldsymbol{r}_0/\beta$
2. 定义矩阵$H_m = \{h_{i,j}\}_{i,j=1,\cdots,m} \in \mathbb{R}^{m\times m}$; 设$H_m = 0$
3. *For* $j = 1, 2, \cdots, m$, *Do*
4. 计算$\boldsymbol{w}_j := A\boldsymbol{v}_j$
5. *For* $i = 1, \cdots, j$, *Do*
6. $h_{ij} = (\boldsymbol{w}_j, \boldsymbol{v}_j)$
7. $\boldsymbol{w}_j := \boldsymbol{w}_j - h_{ij}\boldsymbol{v}_j$
8. *End Do*
9. 计算$h_{j+1,j} = \|\boldsymbol{w}_j\|_2$。若$h_{j+1,j} = 0$, 则设$m := j$转第12
10. 计算$\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
11. *End Do*
12. 计算$\boldsymbol{y}_m = H_m^{-1}(\beta\boldsymbol{e}_1)$和$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}_m$

FOM算法依赖于$\mathcal{K}_m$的维数m，在实际计算的时候需要动态的选取m。如果能够比较经济的得到残差$\boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m$（不需要计算出$\boldsymbol{x}_m$），则算法可在适当的时候停止。

问题

在*FOM*算法中，近似解$\boldsymbol{x}_m \in \boldsymbol{x}_0 + \mathcal{K}_m$的残差满足

$$\boldsymbol{b} - A\boldsymbol{x}_m = -h_{m+1,m}\boldsymbol{e}_m^T\boldsymbol{y}_m\boldsymbol{v}_{m+1}$$

因此

$$\|\boldsymbol{b} - A\boldsymbol{x}_m\|_2 = h_{m+1,m}|\boldsymbol{e}_m^T\boldsymbol{y}_m|.$$

问题

若*Arnoldi*过程在第j步中断，即$h_{j+1,j}=0$，则方法找到了准确解$\boldsymbol{x}_j=\boldsymbol{x}^*$。

证明： 第j步Arnoldi过程为

$$AV_j = V_jH_j + h_{j+1,j}\boldsymbol{v}_{j+1}\boldsymbol{e}_j^T$$

若中断，即$h_{j+1,j}=0$，则

$$AV_j = V_jH_j \Rightarrow H_j\text{的全部特征值均为}A\text{的特征值。}$$

设$(\mu,\boldsymbol{w})$是H_j的任一特征对，即

$$H_j\boldsymbol{w} = \mu\boldsymbol{w}$$

在$AV_j = V_jH_j$右乘$\boldsymbol{w}$得

$$AV_j\boldsymbol{w} = V_jH_j\boldsymbol{w} = V_j(\mu\boldsymbol{w}) = \mu V_j\boldsymbol{w}$$

因此

$$A(V_j\boldsymbol{w}) = \mu(V_j\boldsymbol{w}) \Rightarrow (\mu, V_j\boldsymbol{w})是A的一个特征对。$$

A非奇异，则其特征值λ都不为0，所以$\mu \neq 0$，则

$$\det H_j \neq 0 \Longrightarrow H_j^{-1}存在 \Longrightarrow H_j\boldsymbol{y}_j = \beta\boldsymbol{e}_1解存在唯一!$$

亦即

$$\boldsymbol{x}_j = \boldsymbol{x}_0 + V_j\boldsymbol{y}_j解存在唯一!$$

而

$$\|\boldsymbol{r}_j\|_2 = h_{j+1,j}|\boldsymbol{e}_j^T\boldsymbol{y}_j| = 0$$

所以

$$\boldsymbol{x}_j = \boldsymbol{x}^*.$$

Gram-Schmidt正交化随着m的增加，计算量至少增加$O(m^2n)$；存储量按照$O(mn)$增长。若$n \gg 1$，则m的取值不能太大，一个补救的方案就是重启的Arnoldi方法。

算法 (Restarted FOM–FOM(m))

1. 计算$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, \beta = \|\boldsymbol{r}_0\|_2$和$\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$
2. 从$\boldsymbol{v}_1$开始用*Arnoldi*算法生成*Arnoldi*基和矩阵H_m
3. 计算$\boldsymbol{y}_m = H_m^{-1}\beta\boldsymbol{e}_1$和$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}_m$，若满足迭代终止条件，退出
4. 令$\boldsymbol{x}_0 := \boldsymbol{x}_m$转*1*

m的最优选取？

注记

最优选择很困难，实际中凭经验给出，通常取$m = 10, 20, \cdots$，一般不超过*100*。

重启的Arnoldi方法可能会失败。

$$A = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \Rightarrow A^T A = I,\ |\lambda| = 1.$$

而

$$\boldsymbol{b} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \boldsymbol{e}_1,\ \ \boldsymbol{x}_0 = \mathbf{0} \in \mathbb{R}^n$$

由Arnoldi过程有

$$\boldsymbol{v}_1 = \boldsymbol{b} = \boldsymbol{e}_1,\ \ \boldsymbol{v}_i = \boldsymbol{e}_i,\ \ i = 2, 3, \cdots, m.$$

$$V_m = (\boldsymbol{e}_1, \boldsymbol{e_2}, \cdots, \boldsymbol{e}_m)$$

而

$$H_m = V_m^T A V_m = \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix} = A\text{的}m\text{阶顺序主子式。}$$

当$1 \le m \le n-1$时上述矩阵奇异，Arnoldi方法的近似解

$$\boldsymbol{x}_m,\ 1 \le m \le n-1 \text{ 不存在!}$$

只有$m = n$时

$$H_n = A,\ \boldsymbol{x}_n = \boldsymbol{x}^*$$

这意味着算法完全失败!

对称矩阵的Lanczos过程

若$A = A^T \in \mathbb{R}^{n\times n}$对称，对给定的$\boldsymbol{r}_0$，Arnoldi过程可得：

$$H_m = V_m^T A V_m = H_m^T \quad \text{也是对称的！}$$

而$H_m \in \mathbb{R}^{m\times m}$是上Henssenberg矩阵，所以$H_m$是三对角阵，

$$H_m := T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix},$$

这里

$$\begin{aligned} \alpha_j &= h_{jj} = (A\boldsymbol{v}_j, \boldsymbol{v}_j), \\ \beta_j &= h_{j,j-1} = (A\boldsymbol{v}_{j-1}, \boldsymbol{v}_j) = (A\boldsymbol{v}_j, \boldsymbol{v}_{j-1}) \end{aligned}$$

算法 (Lanczos迭代)

1. $\beta_1 = 0, \boldsymbol{v}_0 = \boldsymbol{0}$
2. $\boldsymbol{v}_1 = \boldsymbol{r}_0 / \|\boldsymbol{r}_0\|_2$
3. *for* $j = 1, 2, \cdots, m$ *do*
4. $\quad$ *Initialize* $\boldsymbol{w}_j = A\boldsymbol{v}_j - \beta_j \boldsymbol{v}_{j-1}$
5. $\quad$ *Compute* $\alpha_j = (\boldsymbol{w}_j, \boldsymbol{v}_j)$
6. $\quad$ *Update* $\boldsymbol{w}_j = \boldsymbol{w}_j - \alpha_j \boldsymbol{v}_j$
7. $\quad$ $\beta_{j+1} = \|\boldsymbol{w}_j\|_2$
8. $\quad$ *if* $\beta_j = 0$ *then stop*
9. $\quad$ $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j / \beta_{j+1}$
10. *end for*

习题

试用T_m的三对角结构证明

$$\boldsymbol{v}_{j+1} \in \text{span}\{\boldsymbol{v}_{j-1}, \boldsymbol{v}_j, A\boldsymbol{v}_j\}$$

特别的请证明如下的三项递推关系：

$$A\boldsymbol{v}_j = \beta_{j+1}\boldsymbol{v}_{j+1} + \alpha_j\boldsymbol{v}_j + \beta_j\boldsymbol{v}_{j-1}.$$

注意到 β_{j+1}可通过归一化$\boldsymbol{w}_j = A\boldsymbol{v}_j - \alpha\boldsymbol{v}_j - \beta_j\boldsymbol{v}_{j-1}$得到。

注记

由三项递推关系，基于*Lanczos*过程的算法每一步只需要保存三个向量。*Arnoldi*算法？

在Lanczos算法中有三项递推关系：

$$\beta_{j+1}\boldsymbol{v}_{j+1} = A\boldsymbol{v}_j - \alpha_j\boldsymbol{v}_j - \beta_j\boldsymbol{v}_{j-1}$$

而正交多项式也有类似的三项递推关系。实际上考虑映射

$$q \in \mathcal{P}_{m-1} \longrightarrow \boldsymbol{x} = q(A)\boldsymbol{v}_1 \in \mathcal{K}_m$$

则可以在$\mathcal{P}_{m-1}$上定义内积：

$$(p, q)_{\boldsymbol{v}_1} := (P(A)\boldsymbol{v}_1, q(A)\boldsymbol{v}_1).$$

若$\dim\mathcal{K}_m = m$，这上述定义的确是内积。而

$$\boldsymbol{v}_i = q_{i-1}(A)\boldsymbol{v}_1, \quad \forall \boldsymbol{v}_i \in \mathcal{K}_m.$$

则$\boldsymbol{v}_i$的正交性就转化为相应的多项式在内积$(\cdot,\cdot)_{\boldsymbol{v}_1}$下的正交性。

求解对称问题的直接Lanczos方法–D-Lanczos

设$A = A^T$(A不一定正定)。因为V_m的列向量是$\mathcal{K}_m$的一组正交基，用这组基作为投影方法的一组基，则由Galerkin 正交性：

求 $\boldsymbol{x}_m \in \boldsymbol{x}_0 + \mathcal{K}_m$

使得 $\boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m$ 满足 $\boldsymbol{r}_m \perp \mathcal{K}_m$

D-Lanczos方法是把Lanczos迭代和三对角Lanczos矩阵T_m的LU分解合成一个迭代格式的方法，即：正交化和求解$\boldsymbol{x}_m$同时进行。

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m \boldsymbol{y}_m, \quad V_m = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_m], \quad \boldsymbol{y}_m \in \mathbb{R}^m \text{ 待定}$$

所以

$$A\boldsymbol{x}_m = A\boldsymbol{x}_0 + AV_m\boldsymbol{y}_m \Longrightarrow \boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m = \boldsymbol{r}_0 - AV_m\boldsymbol{y}_m$$

由Galerkin正交性可得

$$\boldsymbol{r}_0 - AV_m\boldsymbol{y}_m \perp \mathcal{K}_m \;\; i.e. \;\; (\boldsymbol{r}_0 - AV_m\boldsymbol{y}_m, V_m) = 0, \;\; \forall \boldsymbol{y}_m \in \mathbb{R}^m$$

这和法方程等价

$$V_m^T(\boldsymbol{r}_0 - AV_m\boldsymbol{y}_m) = 0 \Longleftrightarrow \underbrace{V_m^T A V_m}_{T_m}\boldsymbol{y}_m = V_m^T\boldsymbol{r}_0 = \beta\boldsymbol{e}_1, \;\; \beta = \|\boldsymbol{r}_0\|_2.$$

其中$\boldsymbol{e}_1 = (1, 0, \cdots, 0)^T \in \mathbb{R}^m$，因为$\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$。因此$\boldsymbol{y}_m$可通过求解如下的三对角方程得到

$$T_m\boldsymbol{y}_m = \beta\boldsymbol{e}_1.$$

这是求解$A\boldsymbol{x} = \boldsymbol{b}$的一个投影方程。若Lanczos迭代没有中断，则上述方程可以利用$T_m = L_mU_m$求出唯一解：

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_mU_m^{-1}L_m^{-1}(\beta\boldsymbol{e}_1)$$

习题：假设$m = 1, \cdots, M$时，三对角的Lanczos矩阵

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix} \in \mathbb{R}^{m\times m}$$

的LU分解$T_m = L_m U_m$存在。则对$m = 2, \cdots, M$有

a) T_m的LU分解具有双对角形式

$$T_m = L_m U_m = \begin{pmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & \lambda_{m-1} & 1 & \\ & & & \lambda_m & 1 \end{pmatrix} \begin{pmatrix} \eta_1 & \omega_2 & & & \\ & \eta_2 & \omega_3 & & \\ & & \ddots & \ddots & \\ & & & \eta_{m-1} & \omega_m \\ & & & & \eta_m \end{pmatrix}$$

b) 验证$\lambda_m, \omega_m, \eta_m$的递推关系，令$\lambda_1 = 0$:

$$\omega_m = \beta_m, \quad \lambda_m = \frac{\beta_m}{\eta_{m-1}}, \quad \eta_m = \alpha_m - \lambda_m \omega_m$$

并证明L_m和U_m可以递归的从L_{m-1}和U_{m-1}得到

$$L_m = \left(\begin{array}{cc|c} & & 0 \\ & L_{m-1} & \vdots \\ & & 0 \\ \hline \mathbf{0}^T & \lambda_m & 1 \end{array}\right), \quad U_m = \left(\begin{array}{c|c} & \\ U_{m-1} & \omega_m \\ & \\ \hline \mathbf{0}^T & \eta_m \end{array}\right)$$

c) 证明若

$$L = \left(\begin{array}{c|c} L' & \mathbf{0} \\ \hline \boldsymbol{l}^T & 1 \end{array}\right), \quad U = \left(\begin{array}{c|c} U' & \boldsymbol{y} \\ \hline \mathbf{0}^T & \eta \end{array}\right), \quad \boldsymbol{l}, \boldsymbol{y} \in \mathbb{R}^{m-1}$$

则

$$L^{-1} = \left(\begin{array}{c|c} L'^{-1} & \mathbf{0} \\ \hline -\boldsymbol{l}^T L'^{-1} & 1 \end{array}\right), \quad U^{-1} = \left(\begin{array}{c|c} U'^{-1} & -\frac{1}{\eta} U'^{-1} \boldsymbol{y} \\ \hline \mathbf{0}^T & \frac{1}{\eta} \end{array}\right)$$

希望从$\boldsymbol{v}_i, L_m$和U_m的递归关系得到$\boldsymbol{x}_m$的一个简单的递归表达式。令

$$P_m = V_m U_m^{-1}, \quad \boldsymbol{z}_m = L_m^{-1}\beta\boldsymbol{e}_1 \Longrightarrow \boldsymbol{x}_m = \boldsymbol{x}_0 + P_m\boldsymbol{z}_m$$

从前面的习题知

$$\boldsymbol{z}_m = L_m^{-1}\beta\boldsymbol{e}_1 = \left(\begin{array}{c|c} L_{m-1}^{-1} & \boldsymbol{0} \\ \hline -\boldsymbol{l}_m^T & 1 \end{array}\right)\beta\boldsymbol{e}_1 = \begin{pmatrix} \boldsymbol{z}_{m-1} \\ \zeta_m \end{pmatrix}, \quad \zeta_m = \beta\boldsymbol{l}_m^T\boldsymbol{e}_1$$

由U_m和其逆矩阵的表达式有

$$\begin{aligned} P_m &= V_m U_m^{-1} = \left(\begin{array}{c|c} V_{m-1} & \boldsymbol{v}_m \end{array}\right)\left(\begin{array}{c|c} U_{m-1}^{-1} & \boldsymbol{y}_{m-1} \\ \hline \boldsymbol{0}^T & \frac{1}{\eta_m} \end{array}\right) \\ &= \left(\begin{array}{c|c} P_{m-1} & \boldsymbol{p}_m \end{array}\right), \text{ 其中 } \boldsymbol{p}_m = V_{m-1}\boldsymbol{y}_{m-1} + \frac{1}{\eta_m}\boldsymbol{v}_m \end{aligned}$$

这说明矩阵P_m可以从P_{m-1}加上一列得到。

因此

$$\begin{aligned} \boldsymbol{x}_m &= \boldsymbol{x}_0 + P_m \boldsymbol{z}_m = \boldsymbol{x}_0 + \left(\begin{array}{c|c} P_{m-1} & \boldsymbol{p}_m \end{array} \right) \begin{pmatrix} \boldsymbol{z}_{m-1} \\ \zeta_m \end{pmatrix} \\ &= \boldsymbol{x}_0 + P_{m-1}\boldsymbol{z}_{m-1} + \zeta_m \boldsymbol{p}_m \\ &= \boldsymbol{x}_{m-1} + \zeta_m \boldsymbol{p}_m \end{aligned}$$

关键是如何有效的得到搜索方向$\boldsymbol{p}_m$和ζ_m。
$\boldsymbol{p}_m$是矩阵$P_m = V_m U_m^{-1}$的第m列；可通过考虑$P_m U_m = V_m$的第m列，并注意到U_m是两对角矩阵得到：

$$V_{j,m} = \sum_{i=1}^{m} P_{j,i} U_{i,m} = P_{j,m-1}\omega_m + P_{j,m}\eta_m$$

由此可得

$$\boldsymbol{v}_m = \omega_m \boldsymbol{p}_{m-1} + \eta_m \boldsymbol{p}_m.$$

上面的关于搜索方向和$\boldsymbol{v}_i$的公式在$m=1$时亦成立（令$\boldsymbol{p}_0=\boldsymbol{0}$即可）！
类似的，z_m可由下面的方程得到：

$$L_m\boldsymbol{z}_m=\beta\boldsymbol{e}_1\Longrightarrow\boldsymbol{z}_m=\begin{pmatrix}\boldsymbol{z}_{m-1}\\ \zeta_m\end{pmatrix},$$

其中

$$\zeta_m=-\lambda_m\zeta_{m-1}$$

最后，每一步迭代，$\boldsymbol{x}_m$可由下式更新：

$$\boldsymbol{x}_m=\boldsymbol{x}_{m-1}+\zeta_m\boldsymbol{p}_m$$

D-Lanczos算法是前面的想法的具体实现：

1. α_m和β_m的定义以及Lanczos向量$\boldsymbol{v}_m$的递推关系
2. 三对角阵T_m的LU分解中的$\lambda_m, \omega_m, \zeta_m$的计算公式
3. 搜索方向的递推公式$\boldsymbol{v}_m = \omega_m \boldsymbol{p}_{m-1} + \eta_m \boldsymbol{p}_m$
4. $\zeta_m = -\lambda_m \zeta_{m-1}$
5. $\boldsymbol{x}_m$的递推公式：$\boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \zeta_m \boldsymbol{p}_m$

注记

*D-Lanczos*算法的实现是基于A的对称性的，并不要求A正定，上面的推导假定T_m的*LU*分解存在。若A对称正定，*LU*分解肯定存在，*D-Lanczos*算法就是*CG*法。但若A仅仅是对称，则*D-Lanczos*算法有可能中断。

算法 (D-Lanczos)

1. *Compute* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$; $\zeta_1 = \beta = \|\boldsymbol{r}_0\|_0$; $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$
2. $\lambda_1 = \beta_1 = 0, d_0 = 0$;
3. *for* $m = 1, 2, \cdots,$ *do*
4. *Compute* $\boldsymbol{w} = A\boldsymbol{v}_m - \beta_m \boldsymbol{v}_{m-1}$ *and* $\alpha_m = (\boldsymbol{w}, \boldsymbol{v}_m)$
5. *if* $m > 1$ *then compute* $\lambda_m = \frac{\beta_m}{\eta_{m-1}}$ *and* $\zeta_m = -\lambda_m \zeta_{m-1}$
6. $\eta_m = \alpha_m - \lambda_m \beta_m$
7. $\boldsymbol{p}_m = (\boldsymbol{v}_m - \beta_m \boldsymbol{p}_{m-1})/\eta_m$
8. $\boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \zeta_m \boldsymbol{p}_m$
9. *if* $\boldsymbol{x}_m$ *has converged, then stop*
10. $\boldsymbol{w} = \boldsymbol{w} - \alpha_m \boldsymbol{v}_m$; $\beta_{m+1} = \|\boldsymbol{w}\|_2$; $\boldsymbol{v}_{m+1} = \boldsymbol{w}/\beta_{m+1}$
11. *end for*

推论

设$\boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m, m = 0, 1, \cdots$ 是*D-Lanczos*算法的残量，

$\boldsymbol{p}_m$, $m = 0, 1, \cdots$ 是搜索方向，则

1. $\boldsymbol{r}_m = \sigma_m \boldsymbol{v}_{m+1}$，$\sigma_m \in \mathbb{R}$，则有$(\boldsymbol{r}_i, \boldsymbol{r}_j) = 0$，$\forall i \neq j$；
2. 搜索方向$\boldsymbol{p}_i$是A-共轭的：$(A\boldsymbol{p}_i, \boldsymbol{p}_j) = 0$，$\forall i \neq j$。

如果A是正定对称的，D-Lanczos算法加上正交和共轭条件就可以得到CG法。

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j \Longrightarrow \boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$$

因为$\boldsymbol{r}_j$相互正交，所以

$$(\boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j, \boldsymbol{r}_j) = 0 \Longrightarrow \alpha_j = \frac{(\boldsymbol{r}_j, \boldsymbol{r}_j)}{(A\boldsymbol{p}_j, \boldsymbol{r}_j)}$$

而Lanczos算法中，下一步的搜索方向

$$\boldsymbol{p}_m = (\boldsymbol{v}_m - \beta_m \boldsymbol{p}_{m-1})/\eta_m \Longrightarrow \boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j \boldsymbol{p}_j$$

而

$$(A\boldsymbol{p}_{j+1}, \boldsymbol{p}_j) = 0 \Longrightarrow \beta_j = -\frac{(\boldsymbol{r}_{j+1}, A\boldsymbol{p}_j)}{(A\boldsymbol{p}_j, \boldsymbol{p}_j)}$$

因为

$$\boldsymbol{r}_j = \boldsymbol{p}_j - \beta_{j-1}\boldsymbol{p}_{j-1}$$

所以

$$(A\boldsymbol{p}_j, \boldsymbol{r}_j) = (A\boldsymbol{p}_j, \boldsymbol{p}_j - \beta_{j-1}\boldsymbol{p}_{j-1}) = (A\boldsymbol{p}_j, \boldsymbol{p}_j)$$

则

$$\alpha_j = \frac{(\boldsymbol{r}_j, \boldsymbol{r}_j)}{(\boldsymbol{p}_j, A\boldsymbol{p}_j)}.$$

由上面的推导，还可以简化β_k的计算

$$\begin{aligned} \beta_j &= -\frac{(\boldsymbol{r}_{j+1}, A\boldsymbol{p}_j)}{(\boldsymbol{p}_j, A\boldsymbol{p}_j)} \\ &= -\frac{(\boldsymbol{r}_{j+1}, \alpha_j^{-1}(\boldsymbol{r}_j - \boldsymbol{r}_{j+1}))}{(\boldsymbol{p}_j, A\boldsymbol{p}_j)} \end{aligned}$$

因为

$$(\boldsymbol{r}_{j+1}, \boldsymbol{r}_j) = 0,$$

所以

$$\beta_j = \frac{\alpha_j^{-1}(\boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1})}{(\boldsymbol{p}_j, A\boldsymbol{p}_j)}.$$

又

$$\alpha_j = \frac{(\boldsymbol{r}_j, \boldsymbol{r}_j)}{(\boldsymbol{p}_j, A\boldsymbol{p}_j)},$$

故而

$$\beta_j = \frac{(\boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1})}{(\boldsymbol{r}_j, \boldsymbol{r}_j)}.$$

由此可以由D-Lanczos得到求解对称正定方程组的共轭梯度法。

算法 (共轭梯度法)

1. 任取$\boldsymbol{x}_0 \in \mathbb{R}^n$,
2. $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$;
3. 对$j = 0, 1, \cdots$

$$
\begin{aligned}
\alpha_j &= \frac{(\boldsymbol{r}_j, \boldsymbol{r}_j)}{(\boldsymbol{p}_j, A\boldsymbol{p}_j)} \\
\boldsymbol{x}_{j+1} &= \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j \\
\boldsymbol{r}_{j+1} &= \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j \\
\beta_j &= \frac{(\boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1})}{(\boldsymbol{r}_j, \boldsymbol{r}_j)} \\
\boldsymbol{p}_{j+1} &= \boldsymbol{r}_{j+1} + \beta_j \boldsymbol{p}_j
\end{aligned}
$$

在计算的过程中，当$\boldsymbol{r}^{(k)} = 0$时计算中止。

- 由于剩余向量相互正交，而$\mathbb{R}^n$中至多只有n个相互正交的非零向量，所以$\boldsymbol{r}_0, \boldsymbol{r}_1, \cdots, \boldsymbol{r}_n$中至少有一个为零。若$\boldsymbol{r}_k = 0$，则有$\boldsymbol{x}_k = \boldsymbol{x}^* = A^{-1}\boldsymbol{b}$为精确解。所以理论上用 CG法求解,至多$n$步便可得到精确解。实际上理论上CG法只需要$m$步即可收敛， 其中$m$是矩阵$A \in \mathbb{R}^{n\times n}$的不同的特征值的数目。 但由于舍入误差的存在，只能把它作为一个迭代法来处理。
- 另一方面在实际计算的时候在一定的精度要求下，CG只需要远远少于n的 迭代次就可以收敛到$\boldsymbol{x}^*$。如用5点格式在100×100的网格上求解Poisson方程得到阶数 为$n = 10000$的矩阵A，A有$m \approx 5000$个不同的特征值。得到和差分方程 的离散误差同阶的近似解只需要144次迭代。

而由CG法所得的多项式$p_m(x)$是下面的极小化问题的解

$$\min_{p\in\mathcal{P}_m} \|p(A)\boldsymbol{e}_0\|_{2,A}, \quad \boldsymbol{e}_0 = \boldsymbol{x}^* - \boldsymbol{x}_0.$$

因为A对称正定，所以

$$\begin{aligned} A &= R\Lambda R^{-1} \Rightarrow A^j = R\Lambda^j R^{-1} \\ &\Rightarrow p_k(A) = RP_k(\Lambda)R^{-1}, \end{aligned}$$

其中

$$P_k(\Lambda) = \begin{bmatrix} P_k(\lambda_1) & & & \\ & P_k(\lambda_2) & & \\ & & \ddots & \\ & & & P_k(\lambda_n) \end{bmatrix}$$

注记

如果$\lambda_1, \cdots, \lambda_n$是$P_k(x)$的根，则$P_k(\Lambda)$是零矩阵则 $\boldsymbol{e}_k = P_k(x)\boldsymbol{e}_0 = 0$。

若A有m个不同的特征值$\lambda_1, \cdots, \lambda_m$，则必有多项式 $P_m \in \mathcal{P}_m$以这些特征值为根，这说明CG法最多m步收敛。由CG法构造的 多项式为$P_m(x) = (1 - x/\lambda_1) \cdots (1 - x/\lambda_m)$。

定理

设$A \in \mathbb{R}^{n\times n}$对称正定，则

$$\frac{\|P(A)\boldsymbol{e}_0\|_A}{\|\boldsymbol{e}_0\|_A} \leq \max_{1\leq j\leq n} |P(\lambda_j)|.$$

证明： 因为A对称正定，所以A有一组相互正交规范特征向量$\boldsymbol{u}_j, j = 1, 2, \cdots, n$。

则

$$\boldsymbol{e}_0 = \sum_{j=1}^{n} a_j \boldsymbol{u}_j, \quad \forall \boldsymbol{e}_0 \in \mathbb{R}^n \Rightarrow \|\boldsymbol{e}_0\|_A^2 = \sum_{j=1}^{n} a_j^2 \lambda_j.$$

这是因为$\boldsymbol{e}_0 = R\boldsymbol{a}$，$R$是由特征向量生成的矩阵，$\boldsymbol{a}$是系数向量，而$R^T = R^{-1}$。

$$\begin{aligned} \|\boldsymbol{e}_0\|_A^2 &= (A\boldsymbol{e}_0, \boldsymbol{e}_0) = \boldsymbol{a}^T R^T A R \boldsymbol{e}_0 = \boldsymbol{a}^T \Lambda \boldsymbol{a}. \\ &\Rightarrow \quad P(A)\boldsymbol{a} = \sum_{j=1}^{n} a_j P(\lambda_j)\boldsymbol{u}_j \end{aligned}$$

则

$$\|P(A)\boldsymbol{e}_0\|_A^2 = \sum_{j=1}^{n} a_j^2 P(\lambda_j)^2 \lambda_j \leq \left[\max_{1\leq j\leq n} (P(\lambda_j))^2\right] \sum_{j=1}^{n} a_j^2 \lambda_j.$$

基于上面的定理，为了估计误差需要找到一个特殊的$\tilde{P}_k \in \mathcal{P}_k$得到一个随着$k$的增加而减小的误差的上界。因为从CG法得到的$P_k$是下面问题的解

$$\min_{P \in \mathcal{P}_k} \|P(A)\boldsymbol{e}_0\|_A,$$

所以

$$\|P_k(A)\boldsymbol{e}_0\|_A \leq \|\tilde{P}_k(A)\boldsymbol{e}_0\|_A,$$

这样就可以得到CG法收敛速度的估计：

$$\|\boldsymbol{e}_k\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|\boldsymbol{e}_0\|_A.$$

此处

$$\kappa = cond_2(A).$$

注记

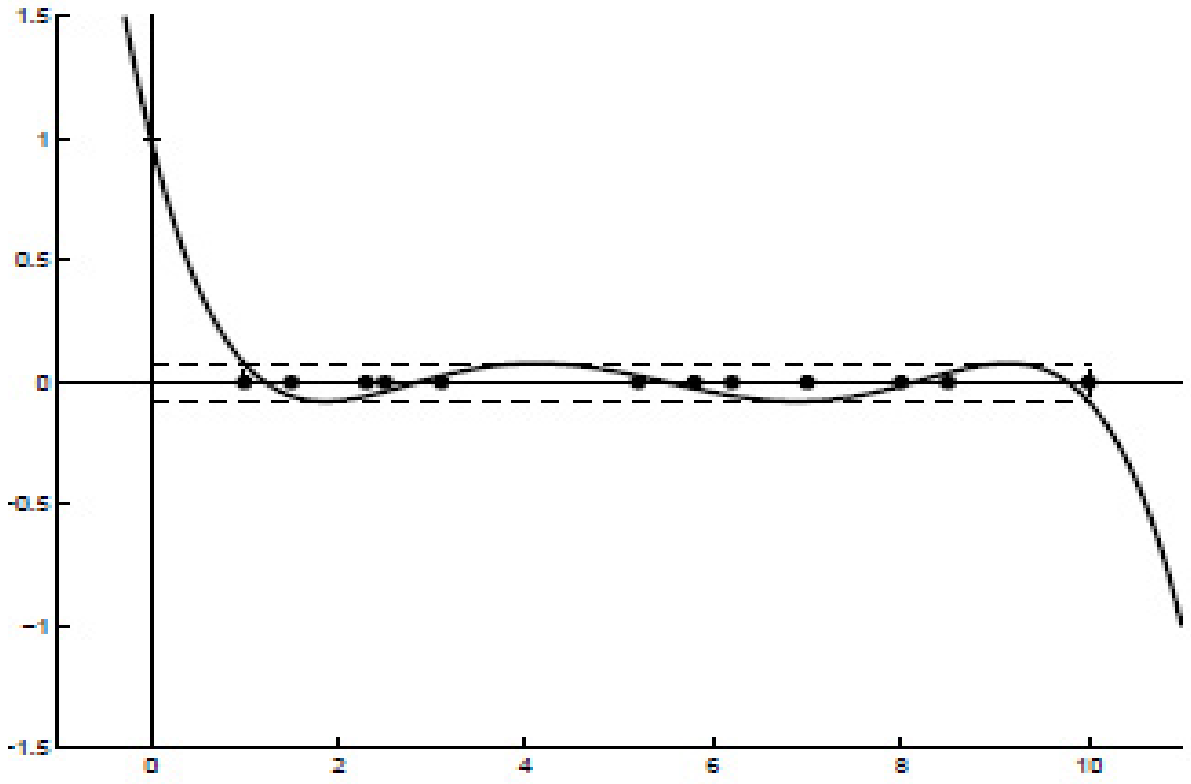
上面的估计只是给出了误差的上界，实际计算的时候误差可能更小。因为刚好$\boldsymbol{e}_0$和某些 特征向量正交或者实际的最优多项式$P_k(x)$在特征值λ_j处比选取的$\tilde{P}_k(x)$ 要小（尤其当特征值成串的集中在某些点上时）。

但是对于很多矩阵来说，上面误差的界是正确的。当κ很大时

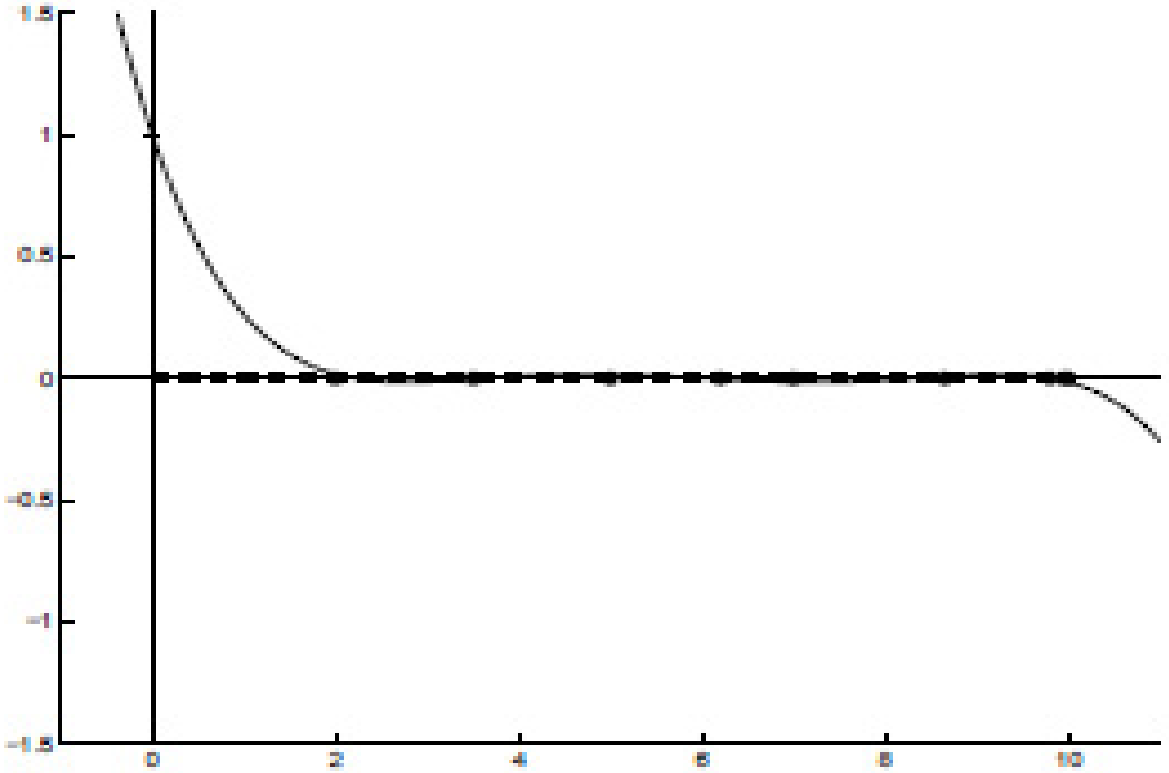
$$2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \approx 2\left(1-\frac{2}{\sqrt{\kappa}}\right)^k \approx 2e^{-2k/\sqrt{\kappa}}. \tag{0.1}$$

所以在实际计算的时候要得到所需的数值精度迭代次数应该为$k = O(\sqrt{\kappa})$。

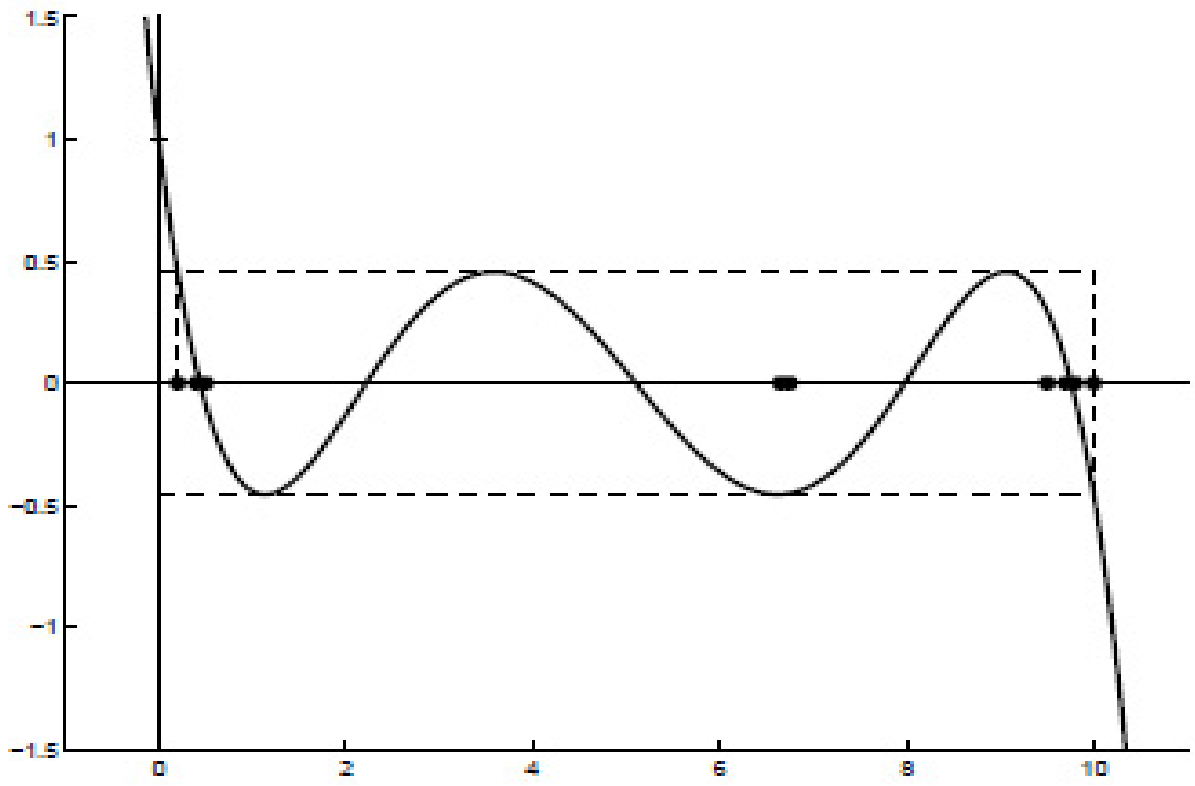
下图是$\tilde{P}_5(x)$的图像，$\|\boldsymbol{e}_5\|_A/\|\boldsymbol{e}_0\|_A \leq 0.0756, \kappa = 10$。



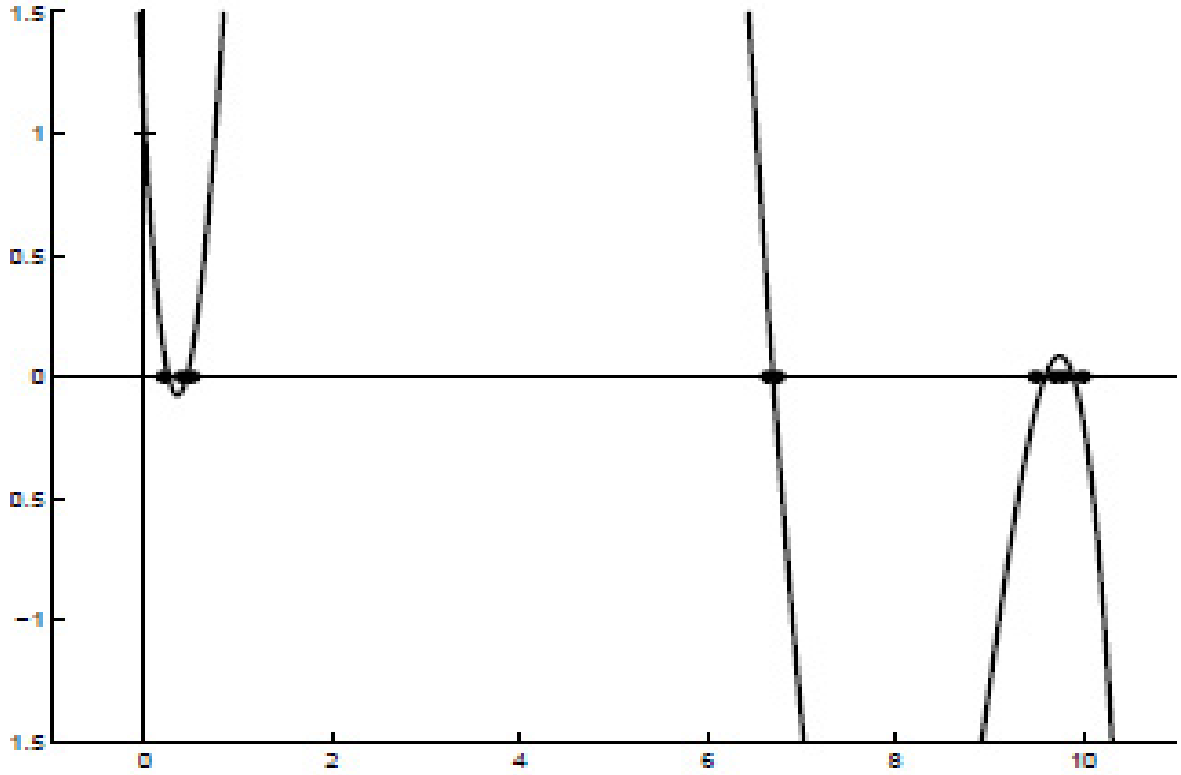
下图的特征值分布较好，$\|\boldsymbol{e}_5\|_A/\|\boldsymbol{e}_0\|_A \leq 0.0163, \kappa = 5$。



下图中$\lambda_1 = 0.2, \lambda_n = 10, \kappa = 50, \|\boldsymbol{e}_5\|_A / \|\boldsymbol{e}_0\|_A \leq 0.4553$， $\tilde{P}_5(x)$的图像。



实际上可以构造$P_5(x)$使得它在特征值附近取值很小，其它地方取值很大。

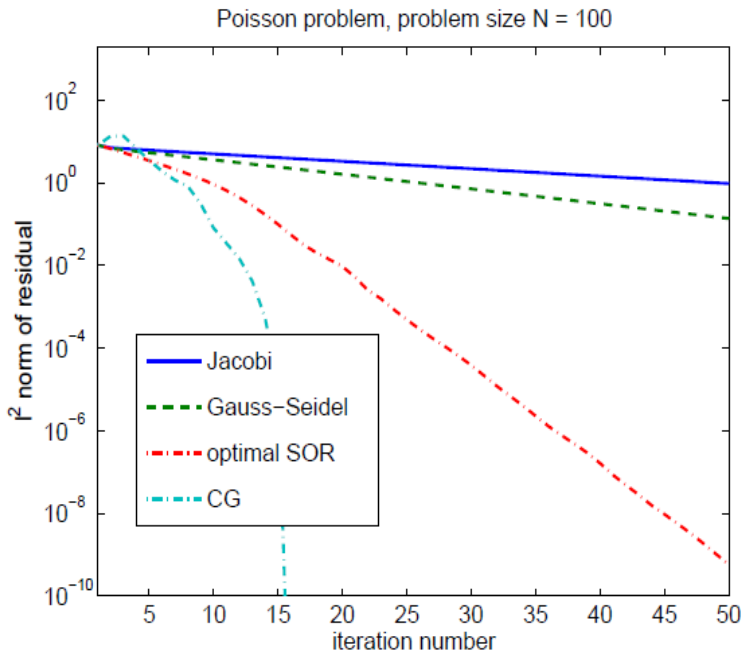


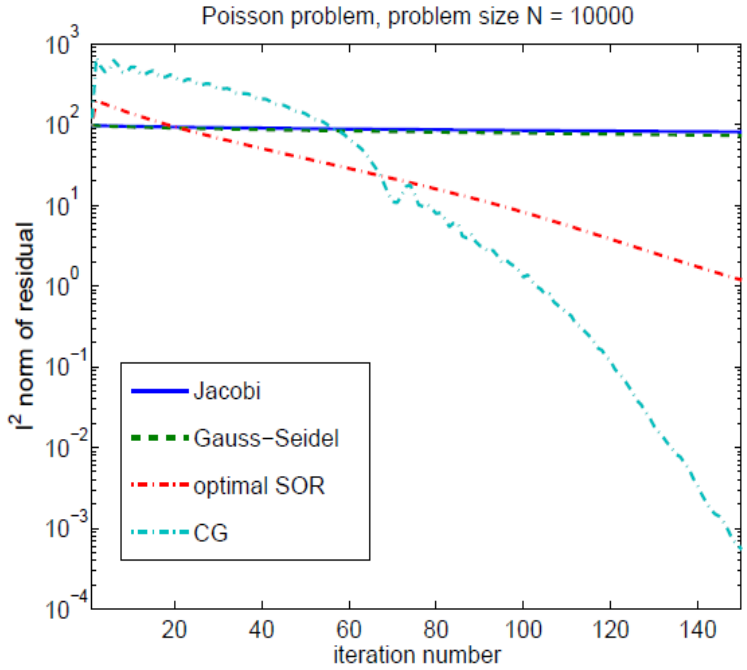
- 以二维Poisson方程的5点格式为例，设在每个方向上有n个节点。用自然排序得到的系数矩阵 的条件数$\kappa = O(1/h^2)$，这里$h = 1/(1+n)$。从(0.1)知CG法需要 $O(n)$次迭代才能收敛（这个结论对任何空间维数都成立）。
- 二维时有n^2个未知数，每步迭代时计算$A\boldsymbol{p}^{(k-1)}$需$O(n^2)$，因此CG法需 $O(n^3)$的计算量来得到给定精度的近似解。Gauss消去法的计算量为$O(n^4)$。最优 松弛因子的SOR需要$O(n^3 \ln n)$。
- 当然对于Poisson方程的数值求解问题，用FFT只需要$O(n^2 \ln n)$的计算量，比 CG法要快。但是对一般的问题，例如变系数的椭圆方程，CG法依然适用，但是SOR只有 在知道最优迭代因子的时候才能用，而FFT是不能直接用的。

四种迭代法收敛速度比较

注记

以上讨论对三维情形亦对，此时*CG*法的优势更为明显。





预处理CG法

CG法的收敛速度依赖于矩阵的条件数，当条件数很大时收敛速度很慢，此时需要做预处理。

$$A\boldsymbol{x} = \boldsymbol{b} \Leftrightarrow M_1^{-1}AM_2^{-1}\boldsymbol{y} = M_1^{-1}\boldsymbol{b},\ \boldsymbol{x} = M_2^{-1}\boldsymbol{y}.$$

但是对CG法一个可能的问题为$M^{-1}A$可能不是对称正定的即使M^{-1}和A是，此时CG法对 预处理后的系统不适用。 因此考虑和原方程等价

$$L^{-1}AL^{-T}(L^T\boldsymbol{x}) = L^{-1}\boldsymbol{b} \Leftrightarrow F\boldsymbol{y} = \boldsymbol{g}$$

其中

$$\boldsymbol{y} = L^T\boldsymbol{x}, \boldsymbol{g} = L^{-1}\boldsymbol{b}, F = L^{-1}AL^{-T}$$

仍然是对称正定阵，CG法适用。
但在实际计算的时候，一般不显式计算 F，因为计算量太大。

- 任取初值$\boldsymbol{y}^{(0)}$，$\tilde{\boldsymbol{r}}^{(0)} = \boldsymbol{g} - F\boldsymbol{y}^{(0)}$；
- 对$k = 1, 2, \cdots,$

$$
\begin{aligned}
\tilde{\alpha}_k &= \frac{(\tilde{\boldsymbol{r}}^{(k)}, \tilde{\boldsymbol{r}}^{(k)})}{(\tilde{\boldsymbol{p}}^{(k)}, F\tilde{\boldsymbol{p}}^{(k)})}, \\
\tilde{\boldsymbol{y}}^{(k+1)} &= \boldsymbol{y}^{(k)} + \tilde{\alpha}_k \tilde{\boldsymbol{p}}^{(k)}, \\
\tilde{\boldsymbol{r}}^{(k+1)} &= \tilde{\boldsymbol{r}}^{(k)} - \tilde{\alpha}_k F\tilde{\boldsymbol{p}}^{(k)}, \\
\tilde{\beta}_k &= \frac{(\tilde{\boldsymbol{r}}^{(k+1)}, \tilde{\boldsymbol{r}}^{(k+1)})}{(\tilde{\boldsymbol{r}}^{(k)}, \tilde{\boldsymbol{r}}^{(k)})}, \\
\tilde{\boldsymbol{p}}^{(k+1)} &= \tilde{\boldsymbol{r}}^{(k+1)} + \tilde{\beta}_k \tilde{\boldsymbol{p}}^{(k)}.
\end{aligned}
$$

注记

从形式上看需要计算L^{-1}及L，但是把上面的计算过程变回到原来的变量发现只需要计算 $M = LL^T$。

令$\boldsymbol{x}^{(k)} = L^{-T}\boldsymbol{y}^{(k)}$, 则

$$\begin{aligned}\tilde{\boldsymbol{r}}^{(k)} &= \boldsymbol{g} - F\boldsymbol{y}^{(k)} \\ &= L^{-1}(\boldsymbol{b} - AL^{-T}L^{T}\boldsymbol{x}^{(k)}) \\ &= L^{-1}\boldsymbol{r}^{(k)},\end{aligned}$$

再令

$$\begin{aligned}\boldsymbol{p}^{(k)} &= L^{-T}\tilde{\boldsymbol{p}}^{(k)}, \\ \boldsymbol{p}^{(0)} &= L^{-T}L^{-1}\boldsymbol{r}^{(0)}.\end{aligned}$$

记

$$\boldsymbol{z}^{(k)} = L^{-T}L^{-1}\boldsymbol{r}^{(k)} = M^{-1}\boldsymbol{r}^{(k)}.$$

算法 (预处理共轭梯度法)

1. 任取$\boldsymbol{x}^{(0)} \in \mathbb{R}^n$,
2. $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}, \boldsymbol{z}^{(0)} = M^{-1}\boldsymbol{r}^{(0)}, \boldsymbol{p}^{(0)} = \boldsymbol{z}^{(0)}$,
3. 对$k = 1, 2, \cdots$有

$$
\begin{aligned}
\alpha_k &= \frac{(\boldsymbol{z}^{(k)}, \boldsymbol{r}^{(k)})}{(\boldsymbol{p}^{(k)}, A\boldsymbol{p}^{(k)})}, \\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}, \\
\boldsymbol{r}^{(k+1)} &= \boldsymbol{r}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)}, \\
\text{求解} M\boldsymbol{z}^{(k+1)} &= \boldsymbol{r}^{(k+1)}, \\
\beta_k &= \frac{(\boldsymbol{z}^{(k+1)}, \boldsymbol{r}^{k+1})}{(\boldsymbol{z}^{(k)}, \boldsymbol{r}^{(k)})}, \\
\boldsymbol{p}^{(k+1)} &= \boldsymbol{z}^{(k+1)} + \beta_k \boldsymbol{p}^{(k)}.
\end{aligned}
$$

预处理共轭梯度法Matlab程序

```
function [x, error, niter, flag] = conjgrad(A, x, b, P, maxit, tol)
flag = 0; niter = 0; bnrm2 = norm( b );
if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end
r = b - A*x; error = norm( r ) / bnrm2;
if ( error < tol ) return, end
for niter = 1:maxit
    z  = P \ r; rho = (r'*z);
    if niter > 1
       beta = rho / rho1;   p = z + beta*p;
    else
       p = z;
    end
    q = A*p;                alpha = rho / (p'*q );
    x = x + alpha * p;     r = r - alpha*q;
    error = norm( r ) / bnrm2;
    if ( error <= tol ), break, end
    rho1 = rho;
end
if ( error > tol ) flag = 1; end
```

在实际计算的可考虑M的Cholesky分解$M = LL^T$。当$LL^T \approx A$时有

$$F \approx L^{-1}(LL^T)L^{-T} = I \Rightarrow cond(F) \approx 1.$$

一般可以考虑A的一种分裂

$$A = M - N, \ \ M = LL^T \quad (SPD),$$

N尽可能小。这称为A的不完全Cholesky分解。
M的最简单的选择是对角阵。

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

令$M = D$为Jacobi迭代的预处理矩阵，则

$$L = L^T = diag(\sqrt{a_{11}}, \sqrt{a_{22}}, \cdots, \sqrt{a_{nn}}).$$

另外的取法是M为SSOR迭代法的预处理矩阵

$$M_{SSOR} = SS^T,$$

其中

$$\begin{aligned} S &= [\omega(2-\omega)]^{-1/2}(D-\omega L)D^{-1/2}, \\ S^T &= [\omega(2-\omega)]^{-1/2}D^{-1/2}(D-\omega L^T). \end{aligned}$$

则$F = S^{-1}AS^{-T}$的条件数大约为$O(\sqrt{\kappa})$,这里κ是A的条件数。

注记

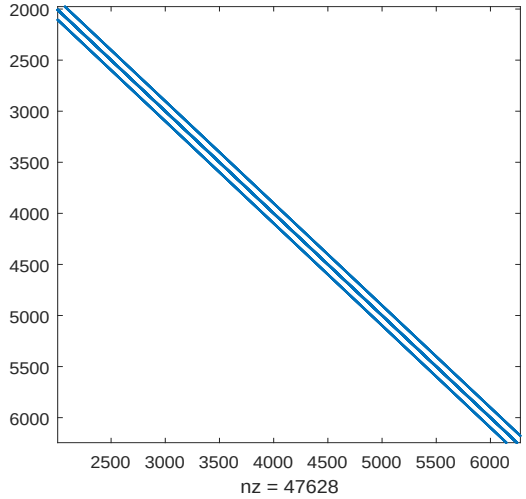
在预处理时，求解$Mz = r$要比$Ax = b$容易求解得多。

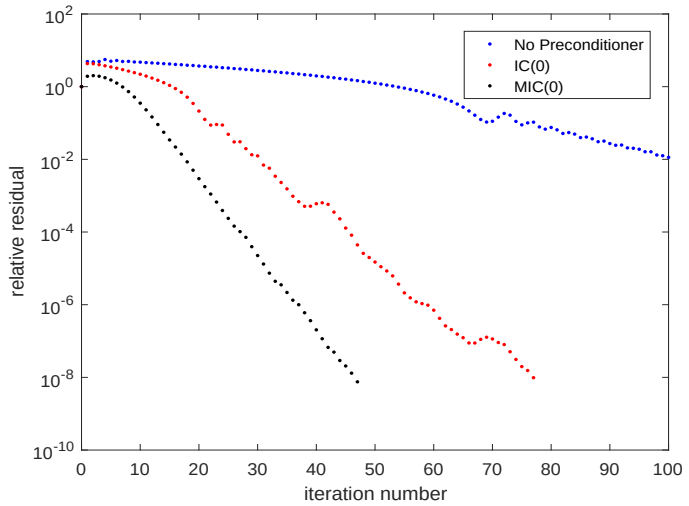
当 $M = I$ 时，$z^{(k)} = r^{(k)}$。 $M = A$ 时，CG 法一步收敛，但

$$Mz = r \Longleftrightarrow Ax = b$$

Matlab的pcg

```
>> A = delsq(numgrid('S',100));
b = ones(size(A,1),1);
[x0,fl0,rr0,it0,rv0] = pcg(A,b,1e-8,100);
>> L = ichol(A);
[x1,fl1,rr1,it1,rv1] = pcg(A,b,1e-8,100,L,L');
>> L = ichol(A,struct('michol','on'));
[x2,fl2,rr2,it2,rv2] = pcg(A,b,1e-8,100,L,L');
>> figure;
semilogy(0:it0,rv0/norm(b),'b.');
hold on;
semilogy(0:it1,rv1/norm(b),'r.');
semilogy(0:it2,rv2/norm(b),'k.');
legend('No Preconditioner','IC(0)','MIC(0)');
xlabel('iteration number');
ylabel('relative residual');
hold off;
```





Arnoldi/Lanczos过程与多项式逼近有内在联系：

Arnoldi/Lanczos逼近问题：

找一个首1的多项式$p_m \in \mathcal{P}_m$使得$\|p_m(A)\boldsymbol{r}_0\|_2$最小。 (*)

上面极小化问题的解由如下定理给出：

定理

若*Arnoldi/Lanczos*过程不中断，则逼近问题(*)有唯一解p_m，而p_m是H_m的特征多项式。

证明： 因为

$$\mathcal{K}_m = \mathcal{K}_m(A, \boldsymbol{r}_0) = \text{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0\}$$

而V_m的列向量组成$\mathcal{K}_m$的一组基；任何首1的多项式$p \in \mathcal{P}_m$，向量$p(A)\boldsymbol{r}_0$可写为

$$p(A)\boldsymbol{r}_0 = A^m\boldsymbol{r}_0 - V_m\boldsymbol{y}_m \in A^m\boldsymbol{r}_0 \oplus \mathcal{K}_m,\ \boldsymbol{y}_m \in \mathbb{R}^m \text{ 系数向量}$$

则问题(*)和下面的最小二乘问题等价：

$$\text{求 } \boldsymbol{y}_m \in \mathbb{R}^m, i.e., \ V_m\boldsymbol{y}_m \in \mathcal{K}_m \ \text{ 使得 } \|V_m\boldsymbol{y}_m - A^m\boldsymbol{r}_0\|_2 \text{ 最小}$$

算法不中断，所以$\text{rank}(V_m) = m$，则最小二乘问题有唯一解$\boldsymbol{y}_m$。则

$$V_m^T(\underbrace{A^m\boldsymbol{r}_0 - V_m\boldsymbol{y}_m}_{p_m(A)\boldsymbol{r}_0}) = 0 \Leftrightarrow p_m(A)\boldsymbol{r}_0 \perp \mathcal{K}_m$$

为求出p_m，考虑$H_m = V_m^T A V_m$，因为$V_m V_m^T = I_{m\times m}$，
而$\boldsymbol{r}_0, A\boldsymbol{r}_0, \cdots, A^{m-1}\boldsymbol{r}_0 \in \mathcal{K}_m$，则

$$\begin{aligned}
V_m^T A\boldsymbol{r}_0 &= V_m^T A V_m V_m^T\boldsymbol{r}_0 = H_m V_m^T\boldsymbol{r}_0 \\
V_m^T A^2\boldsymbol{r}_0 &= V_m^T A V_m V_m^T A\boldsymbol{r}_0 = V_m^T A V_m V_m^T A V_m V_m^T\boldsymbol{r}_0 = H_m^2 V_m^T\boldsymbol{r}_0 \\
&\cdots \\
V_m^T A^m\boldsymbol{r}_0 &= \cdots = H_m^m V_m^T\boldsymbol{r}_0
\end{aligned}$$

因此

$$V_m^T p_m(A)\boldsymbol{r}_0 = p(H_m)V_m^T\boldsymbol{r}_0,\ \ \forall p_m \in \mathcal{P}_m$$

考虑H_m的特征多项式χ_m，$\chi_m \in \mathcal{P}_m$是首1多项式，
且$\chi_m(H_m) = 0$(Cayley-Hamilton)。结合前面的推导有

$$0 = \chi_m(H_m)V_m^T\boldsymbol{r}_0 = V_m^T\chi_m(A)\boldsymbol{r}_0$$

因此$p_m = \chi_m$满足

$$V_m^T(A^m\boldsymbol{r}_0 - V_m\boldsymbol{y}_m) = 0$$

而满足上面正交性条件的解是唯一的。

注记

H_m是A的一种近似，满足$\chi_m(H_m)$是$\chi(A)$的近似；这个近似在(*)意义下最优。因此可以通过*Arnoldi*迭代用$\sigma(H_m)$去近似$\sigma(A)$。 而H_m的特征值也叫着*Ritz values*。

Arnoldi方法和Lanczos方法的比较

1. Lanczos过程的计算量和存储量：
   - 存储量：每步只存储$A\boldsymbol{v}_k, \boldsymbol{v}_{k-1}, \boldsymbol{v}_k, \boldsymbol{v}_{k+1}$。每步的存储量恒定。
   - 计算量：Lanczos算法每步计算$A\boldsymbol{v}_j, \alpha_j, \beta_j$和$\boldsymbol{v}_{j+1}$，需一个矩阵乘向量和$9n$ *flops*。每步的计算量恒定。求解三对角方程$T_m\boldsymbol{y}_m = \beta\boldsymbol{e}_1$计算量为$O(m)$。

   其总计算代价和步数m是线性关系。
2. Arnoldi过程的计算量和存储量：
   - 存储量：m步的Lanczos算法必须存储全部的$\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_m$，因$V_m$是稠密阵，$m$不能太大。
   - 计算量：执行m步的Arnoldi过程的计算代价为m个矩阵乘向量和$2nm^2$ *flops*。求方程组$H_m\boldsymbol{y}_m = \beta\boldsymbol{e}_1$的计算量为$O(m^2)$。总的计算代价和$m$不是线性关系，$m$较大时，计算代价很大，必须限制$m$。

习题：考察如下矩阵：

$$A = I + \alpha B,\ \ B \text{ 反对称阵 } B^T = -B.$$

a) 证明

$$\frac{(A\boldsymbol{x}, \boldsymbol{x})}{(\boldsymbol{x}, \boldsymbol{x})} = 1, \quad \forall \boldsymbol{x} \neq \boldsymbol{0}.$$

b) 验证用Arnoldi算法算出的A的Hessenberg矩阵具有如下的三对角形式：

$$H_m = \begin{pmatrix} 1 & -\eta_2 & & & \\ \eta_2 & 1 & -\eta_3 & & \\ & & \cdots & & \\ & & \eta_{m-1} & 1 & -\eta_m \\ & & & \eta_m & 1 \end{pmatrix}$$

c) 利用b)的结论解释用CG法求解$A\boldsymbol{x} = \boldsymbol{b}$，即使$A$不对称，得到的残量也是相互正交的(提示：考虑$A$的分解)：

$$A = \frac{A + A^T}{2} + \frac{A - A^T}{2}.$$

Full Orthogonalization method(FOM)

Arnoldi、D-Lanczos、CG等：

$$\text{求 } \boldsymbol{x}_m \in \boldsymbol{x}_0 + \mathcal{K}_m \text{ 使得 } \boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m \text{ 满足 } \boldsymbol{r}_m \perp \mathcal{K}_m.$$

而

$$\boldsymbol{r}_m = -A\boldsymbol{e}_m = -A(\boldsymbol{x}_m - \boldsymbol{x}^*) \Longrightarrow A\boldsymbol{e}_m = A(\boldsymbol{x}_m - \boldsymbol{x}^*) \perp \mathcal{K}_m.$$

特别的有

$$\boldsymbol{e}_m \perp A\mathcal{K}_m, \text{ 若}A\text{对称}, \quad \boldsymbol{e}_m \perp_A \mathcal{K}_m, \text{ 若}A\text{对称正定}。$$

若A对称正定，则有唯一解$\boldsymbol{x}_m$满足

$$\|\boldsymbol{e}_m\|_A = \|\boldsymbol{x}_m - \boldsymbol{x}^*\|_A = \min_{\boldsymbol{x} \in \boldsymbol{x}_0 + \mathcal{K}_m} \|\boldsymbol{x} - \boldsymbol{x}^*\|_A = \min_{\boldsymbol{e} \in \boldsymbol{e}_0 + \mathcal{K}_m} \|\boldsymbol{e}\|_A.$$

广义极小残量法（Generalized minimal residual method）

若$A \in \mathbb{R}^{n\times n}$非奇异，不对称正定，则可定义内积和2-范数：

$$(\boldsymbol{x},\boldsymbol{x})_A = (A\boldsymbol{x},A\boldsymbol{x}) = \boldsymbol{x}^T A^T A\boldsymbol{x} = \|\boldsymbol{x}\|_2^2.$$

此2-范数下在$\mathcal{K}_m$中的最佳平方逼近，即为广义极小残量法，
如GMRES、MINRES等。

$$\text{求 } \boldsymbol{x}_m \in \boldsymbol{x}_0 + \mathcal{K}_m \text{ 使得 } \boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m \text{ 满足 } \boldsymbol{r}_m \perp A\mathcal{K}_m.$$

上述问题等价于

$$A\boldsymbol{e}_m \perp A\mathcal{K}_m,$$

和如下的最小二乘问题等价：

$$\|\boldsymbol{r}_m\|_2 = \|\boldsymbol{b} - A\boldsymbol{x}_m\|_2 = \min_{\boldsymbol{x}\in\boldsymbol{x}_0+\mathcal{K}_m} \|\boldsymbol{b} - A\boldsymbol{x}\|_2 = \min_{\boldsymbol{r}\in\boldsymbol{r}_0+A\mathcal{K}_m} \|\boldsymbol{r}\|_2$$

用GMRES法求解近似解$\boldsymbol{x}_m$的流程：

1. 用Arnoldi方法得到$\mathcal{K}_m$的一组正交基$V_m = (\boldsymbol{v}_1 | \cdots, | \boldsymbol{v}_m)$；
2. 近似解$\boldsymbol{x}_m$可写为：

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m \boldsymbol{y}_m, \quad \boldsymbol{y}_m \in \mathbb{R}^m.$$

3. 利用Petrov-Galerkin正交性条件$\boldsymbol{r}_m \perp A\mathcal{K}_m$得到法方程

$$(AV_m)^T(A\boldsymbol{x}_m - \boldsymbol{b}) = 0 \Longleftrightarrow V_m^T A^T A V_m \boldsymbol{y}_m = (AV_m)^T \boldsymbol{r}_0$$

由此可以得到$\boldsymbol{x}_m$的表达式为

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m \boldsymbol{y}_m = \boldsymbol{x}_0 + V_m (V_m^T A^T A V_m)^{-1} (AV_m)^T \boldsymbol{r}_0$$

这里

$$V_m^T A^T A V_m \in \mathbb{R}^{m \times m}, \quad m \ll n.$$

习题：若$\boldsymbol{r}_0 \neq \boldsymbol{0}$，而$\overline{H}_m \in \mathbb{R}^{(m+1)\times m}$是由Arnoldi过程得到的上Hessenberg阵：

$$\overline{H}_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1m} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2m} \\ & h_{32} & h_{33} & \cdots & h_{3m} \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \\ & & & & h_{m+1,m} \end{pmatrix}$$

设$h_{j+1,j} \neq 0,\ j = 1, \cdots, m-1$，试证

1. $\mathcal{K}_m = \mathrm{span}\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m\}$且$\dim \mathcal{K}_m = m$；
2. $\overline{H}_m$是列满秩的：$\mathrm{rank}\, \overline{H}_m = m$；
3. 若$h_{m+1,m} = 0$，则$A^m \boldsymbol{r}_0 \in \mathcal{K}_m \Longrightarrow \mathcal{K}_m = \mathcal{K}_{m+1} = \cdots = \mathcal{K}_n$。

注记

*Arnoldi*过程当$h_{m+1,m}=0$而$h_{j+1,j}\neq 0,\ j=1,\cdots,m-1$时会中断，此时叫着幸运中断*(lucky breakdown)*，因为此时的$\boldsymbol{x}_m=\boldsymbol{x}^*$。

假定算法不中断

$$h_{j+1,j}\neq 0,\ \ j=1,\cdots,m-1.$$

令$\beta=\|\boldsymbol{r}_0\|_2$有$\beta\boldsymbol{v}_1=\boldsymbol{r}_0$。且

$$\beta V_{m+1}\boldsymbol{e}_1=\beta\boldsymbol{v}_1=\boldsymbol{r}_0,\ \ \boldsymbol{e}_1=(1,0,\cdots,0)^T\in\mathbb{R}^{m+1}.$$

近似解$\boldsymbol{x}_m=\boldsymbol{x}_0+V_m\boldsymbol{y}_m$的残量满足

$$\begin{aligned}\boldsymbol{b}-A\boldsymbol{x}_m &= b-A(\boldsymbol{x}_0+V_m\boldsymbol{y}_m)\\ &= \boldsymbol{r}_0-AV_m\boldsymbol{y}_m\\ &= \beta\boldsymbol{v}_1-V_{m+1}\overline{H}_m\boldsymbol{y}_m\\ &= V_{m+1}(\beta\boldsymbol{e}_1-\overline{H}_m\boldsymbol{y}_m).\end{aligned}$$

因为V_{m+1}的各列是相互正交的，所以

$$\|\boldsymbol{b}-A\boldsymbol{x}\|_2=\|V_{m+1}(\beta\boldsymbol{e}_1-\overline{H}_m\boldsymbol{y}_m)\|_2=\|\beta\boldsymbol{e}_1-\overline{H}_m\boldsymbol{y}_m\|_2.$$

因此Petrov-Galerkin正交条件和求解$\boldsymbol{y}_m$的最小二乘问题等价

$$\min_{\boldsymbol{x}\in\boldsymbol{x}_0+\mathcal{K}_m}\|\boldsymbol{b}-A\boldsymbol{x}\|_2=\min_{\boldsymbol{y}_m}\|\beta\boldsymbol{e}_1-\overline{H}_m\boldsymbol{y}_m\|_2$$

而Arnoldi算法不中断保证$\overline{H}_m$是列满秩的，因此上面的最小二乘问题有唯一解$\boldsymbol{y}_m\in\mathbb{R}^m$，可通过求解法方程得到：

$$\overline{H}_m^T\overline{H}_m\boldsymbol{y}_m=\overline{H}_m^T\beta\boldsymbol{e}_1$$

而$A\boldsymbol{x}=\boldsymbol{b}$的近似解$\boldsymbol{x}_m$可写为

$$\boldsymbol{x}_m=\boldsymbol{x}_0+V_m\boldsymbol{y}_m.$$

算法 (GMRES)

1. *Compute* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, \beta = \|\boldsymbol{r}_0\|_2$, *and* $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$
2. *for* $j = 1, 2, \cdots, m$ *do*
3. *Compute* $\boldsymbol{w}_j = A\boldsymbol{v}_j$
4. *for* $i = 1, \cdots, j$ *do*
5. $h_{ij} = (\boldsymbol{w}_j, \boldsymbol{v}_i)$
6. $\boldsymbol{w}_j = \boldsymbol{w}_j - h_{ij}\boldsymbol{v}_i$
7. *end for*
8. $h_{i+1,j} = \|\boldsymbol{w}_j\|_2$
9. *if* $h_{j+1,j} = 0$ *set* $m = i$ *goto 12* % *lucky breakdown*
10. $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
11. *end for*
12. *Define the* $(m+1) \times m$ *Hessenberg matrix* $\overline{H}_m = \{h_{ij}\}$
13. *Compute* y_m *as the minimizer of* $\|\beta\boldsymbol{e}_1 - \overline{H}_m\boldsymbol{y}_m\|_2$ *and set* $\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}_m$

为了求解最小二乘问题

$$\min \|\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y}\|_2$$

需要用Givens变换把Hessenberg矩阵化为上三角阵。此时的Givens变换为

$$J_i = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_i & s_i & & \\ & & & -s_i & c_i & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow \text{第} i \text{行} \\ \leftarrow \text{第} i+1 \text{行} \\ \\ \\ \\ \end{matrix}$$

其中$c_i^2 + s_i^2 = 1$。

若GMRES迭代了m次，则上述Givens矩阵

$$J_i \in \mathbb{R}^{(m+1)\times(m+1)}.$$

用一系列的J_i左乘$\overline{H}_m$，可以在每一步迭代消去$h_{i+1,i}$。
设$m=5$，则此时

$$\overline{H}_5 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad \overline{\boldsymbol{g}}_0 = \beta \boldsymbol{e}_1 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

首先$\overline{H}_5$左乘

$$J_1 = \begin{pmatrix} c_1 & s_1 & & & & \\ -s_1 & c_1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix}, \quad s_1 = \frac{h_{21}}{\sqrt{h_{11}^2+h_{21}^2}}, \quad c_1 = \frac{h_{11}}{\sqrt{h_{11}^2+h_{21}^2}}$$

可得矩阵和右端向量

$$\overline{H}_5^{(1)} = \begin{pmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} & h_{15}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} & h_{25}^{(1)} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad \overline{\boldsymbol{g}}_1 = \begin{pmatrix} c_1\beta \\ -s_1\beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

接着构造J_1消去h_{32}，只需取

$$s_2 = \frac{h_{32}}{\sqrt{(h_{22}^{(1)})^2 + h_{32}^2}}, \quad c_2 = \frac{h_{22}^{(1)}}{\sqrt{(h_{22}^{(1)})^2 + h_{32}^2}}$$

则经过m次Givens变换后，$\overline{H}_m$化为，

$$\overline{H}_5^{(1)} = \begin{pmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & 0 \end{pmatrix}, \quad \overline{\boldsymbol{g}}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{pmatrix}$$

一般来说，第i步的Givens变换J_i的c_i和s_i可如下定义：

$$s_i = \frac{h_{i+1,i}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}, \quad c_i = \frac{h_{ii}^{(i-1)}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}$$

定义Q_m是Givens矩阵的乘积

$$Q_m = J_m J_{m-1} \cdots J_1$$

且令

$$\begin{aligned} \overline{R}_m &= \overline{H}_m^{(m)} = Q_m \overline{H}_m \\ \overline{\boldsymbol{b}}_m &= Q_m(\beta \boldsymbol{e}_1) = (\gamma_1, \cdots, \gamma_{m+1})^T. \end{aligned}$$

因为$Q_m^T Q_m = I$，所以

$$\min \|\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y}\|_2 = \min \|\overline{\boldsymbol{g}}_m - \overline{R}_m \boldsymbol{y}\|_2$$

上述最小二乘问题可通过求解去掉$\overline{R}_m$的最后一行得到的上三角方程来求解得到y^*。而此时残差为

$$\|\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y}^*\|_2 = |\gamma_{m+1}|.$$

定理

设$m \leq n$，而$J_i, i = 1, \cdots, m$是把$\overline{H}_m$化成上三角阵$\overline{R}_m$的*Givens*变换，$\overline{\boldsymbol{g}} = (\gamma_1, \cdots, \gamma_{m+1})^T$。$R_m$和$\boldsymbol{g}_m$是分别去掉$\overline{R}_m$的最后一行和$\overline{\boldsymbol{g}}_m$的最后一个元素得到的，则有

1. $\operatorname{rank} AV_m = \operatorname{rank} R_m$，特别的若$r_{mm} = 0$，则$A$奇异；
2. 最小二乘问题$\min \|\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y}\|_2$的解$\boldsymbol{y}_m$为

$$\boldsymbol{y}_m = R_m^{-1} \boldsymbol{g}_m$$

3. 第m步的残量满足

$$\boldsymbol{b} - A\boldsymbol{x}_m = V_{m+1}(\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y}_m) = V_{m+1} Q_m^T (\gamma_{m+1} \boldsymbol{e}_{m+1})$$

所以有

$$\|\boldsymbol{b} - A\boldsymbol{x}_m\|_2 = |\gamma_{m+1}|$$

证明：(1)由Arnoldi过程有

$$\begin{aligned} AV_m &= V_{m+1}\overline{H}_m \\ &= V_{m+1}Q_m^T Q_m \overline{H}_m \\ &= V_{m+1}Q_m^T \overline{R}_m \end{aligned}$$

因为$V_{m+1}Q_m^T$是正交阵，而$\operatorname{rank} AV_m = \operatorname{rank} \overline{R}_m = \operatorname{rank} R_m$。若$r_{mm} = 0$，则$\operatorname{rank} R_m \le m-1 \Rightarrow \operatorname{rank} AV_m \le m-1$。但$V_m$是列满秩的，这说明$A$必奇异。

(2)任何$y \in \mathbb{R}^m$，有

$$\begin{aligned} \|\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y}\|_2^2 &= \|Q_m(\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y})\|_2^2 \\ &= \|\overline{\boldsymbol{g}}_m - \overline{R}_m \boldsymbol{y}\|_2 \\ &= |\gamma_{m+1}|^2 + \|\boldsymbol{g}_m - R_m \boldsymbol{y}\|_2^2 \end{aligned}$$

上式左边在$\|\boldsymbol{g}_m - R_m \boldsymbol{y}\|_2 = 0$，而$\det R_m \neq 0$，亦即$\boldsymbol{y} = R_m^{-1}\boldsymbol{g}_m$时达到最小值。

(3)对任何$\boldsymbol{x} = \boldsymbol{x}_0 + V_m \boldsymbol{y}$有

$$\begin{aligned} \boldsymbol{b} - A\boldsymbol{x} &= V_{m+1}(\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y}) \\ &= V_{m+1} Q_m^T Q_m (\beta \boldsymbol{e}_1 - \overline{H}_m \boldsymbol{y}) \\ &= V_{m+1} Q_m^T (\overline{\boldsymbol{g}}_m - \overline{R}_m \boldsymbol{y}) \end{aligned}$$

在(2)的证明中知$\|\overline{\boldsymbol{g}}_m - \overline{R}_m \boldsymbol{y}\|_2$当$R_m \boldsymbol{y} = \boldsymbol{g}_m$时达到极小值

$$\boldsymbol{b} - A\boldsymbol{x}_m = V_{m+1} Q_m^T (\gamma_{m+1} \boldsymbol{e}_{m+1}).$$

又$V_{m+1} Q_m^T$的列向量相互正交，所以

$$\|\boldsymbol{b} - A\boldsymbol{x}_m\|_2 = |\gamma_{m+1}|.$$

由$\bar{\boldsymbol{g}}_i = J_i \bar{\boldsymbol{g}}_{i-1}$知

$$\gamma_{j+1} = -s_j \gamma_j.$$

因此

$$\begin{aligned} \|\boldsymbol{r}_m\|_2 &= |\gamma_{m+1}| = s_m \|\boldsymbol{r}_{m-1}\|_2 = \cdots \\ &= |s_1 s_2 \cdots s_m| \|\boldsymbol{r}_0\|_2 = |s_1 s_2 \cdots s_m| \beta \end{aligned}$$

定理

若*Arnoldi*过程在第m步中断，则$\boldsymbol{x}_m = \boldsymbol{x}^*$

Arnoldi过程的计算量$O(2nm^2)$，其存储量为$O(mn)$，当m很大时，计算量和存储量都变得不可承受；因此 GMRES必须控制m的大小。一个解决方案就是重启的GMRES(m)：

算法 (重启的GMRES(m))

1. 选$\boldsymbol{x}_0 \in \mathbb{R}^n$，计算$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, \beta = \|\boldsymbol{r}_0\|_2$，给定$m$
2. 执行m步的*Arnoldi*过程，得到$V_{m+1}, \overline{H}_m$
3. 计算$\min_{\boldsymbol{y}\in\mathbb{R}^m} \|\beta\boldsymbol{e}_1 - \overline{H}_m\boldsymbol{y}\|_2$的解$\boldsymbol{y}_m$
4. 计算$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}_m$
5. 如果$\|\boldsymbol{r}_m\|_2 \leq \epsilon$，则停机，否则令$\boldsymbol{x}_m = \boldsymbol{x}_0$转步 *1*

注记

m的选取非常重要，对一般的A，取任意的$m \leq n-1$都不能保证*GMRES(m)*的收敛性。

例

用*GMRES(m)*方法求解

$$A=\begin{pmatrix}0 & & & 1\\ 1 & \ddots & 0 & \\ & \ddots & \ddots & \vdots\\ & & 1 & 0\end{pmatrix},\quad \boldsymbol{b}=\boldsymbol{e}_1,\boldsymbol{x}_0=\boldsymbol{0}$$

解：由Arnoldi过程知$\boldsymbol{v}_1=\boldsymbol{e}_1,\ \boldsymbol{v}_i=\boldsymbol{e}_i,i=2,\cdots,m,$

$$H_m=\begin{pmatrix}0 & & \\ 1 & \ddots & \\ & 1 & 0\end{pmatrix}\quad \overline{H}_m=\begin{pmatrix}0 & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0\\ & & & 1\end{pmatrix}$$

而最小二乘问题：

$$\min_{\boldsymbol{y}\in\mathbb{R}^m} \|\beta\boldsymbol{e}_1 - \overline{H}_m\boldsymbol{y}\|_2 \Longrightarrow \boldsymbol{y}_m = 0 \Longrightarrow \boldsymbol{x}_m = 0, \quad m \leq n-1$$

亦即

$$\begin{aligned} \boldsymbol{x}_0 &= \boldsymbol{x}_1 = \cdots = \boldsymbol{x}_{n-1} = 0 \\ \boldsymbol{r}_0 &= \boldsymbol{r}_1 = \cdots = \boldsymbol{r}_{n-1} = \boldsymbol{e}_1 \end{aligned}$$

Arnoldi方法$\boldsymbol{y}_m$无定义，即$H_m\boldsymbol{y}_m = \beta\boldsymbol{e}_1$中的

$$\|\boldsymbol{y}_m\|_2 \longrightarrow \infty$$

残量

$$\boldsymbol{r}_m = \boldsymbol{r}_0 - AV_m\boldsymbol{y}_m \longrightarrow \infty$$

GMRES(m)方法不收敛！

因为$\boldsymbol{x}_m \in \boldsymbol{x}_0 + \mathcal{K}_m(A, \boldsymbol{r}_0)$，所以存在$q_m \in \mathcal{P}_{m-1}$使得

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + q_m(A)\boldsymbol{r}_0$$

因此

$$\begin{aligned}
\|\boldsymbol{r}_m\|_2 &= \|\boldsymbol{b} - A\boldsymbol{x}_m\|_2 = \|\boldsymbol{r}_0 - Aq_m(A)\boldsymbol{r}_0\|_2 \\
&= \|(I - Aq_m(A))\boldsymbol{r}_0\|_2 = \|p_m(A)\boldsymbol{r}_0\|_2 \\
&= \min_{p\in\mathcal{P}_m, p(0)=1} \|p(A)\boldsymbol{r}_0\|_2
\end{aligned}$$

当$A = X\Lambda X^{-1}$时，$p(A) = Xp(\Lambda)X^{-1}$，因此

$$\begin{aligned}
\|\boldsymbol{r}_m\|_2 &= \min_{p\in\mathcal{P}_m, p(0)=1} \|Xp(\Lambda)X^{-1}\boldsymbol{r}_0\|_2 \\
&\le \kappa(X)\|\boldsymbol{r}_0\|_2 \min_{p\in\mathcal{P}_m, p(0)=1} \|p(\Lambda)\|_2 \\
&= \kappa(X)\|\boldsymbol{r}_0\|_2 \min_{p\in\mathcal{P}_m, p(0)=1} \max_{1\le i\le n} |p(\lambda_i)|
\end{aligned}$$

定理

设$A = X\Lambda X^{-1}, \lambda_i \in E(c,a,d)$，其中$E(c,a,d)$为中心点在$c$，焦距为$d$，长半轴为$a$的椭圆，且$0 \notin E(c,a,d)$，则

$$\|\boldsymbol{r}_m\|_2 \leq \kappa(X) \left| \frac{T_m(a/d)}{T_m(c/d)} \right| \|\boldsymbol{r}_0\|_2.$$

这里$T_m(z)$为m次的$Chebyshev$多项式。

对于$|z| > 1$,

$$T_m(z) = \frac{1}{2}\left((z+\sqrt{z^2-1})^m + (z-\sqrt{z^2-1})^m\right)$$

因此

$$\left| \frac{T_m(a/d)}{T_m(c/d)} \right| = \left(\frac{a+\sqrt{a^2-d^2}}{c+\sqrt{c^2-d^2}} \right)^m$$

Matlab函数-gmres

- x = gmres(A,b) attempts to solve the system of linear equations A*x = b for x. The n-by-n coefficient matrix A must be square and should be large and sparse. The column vector b must have length n. A can be a function handle, afun, such that afun(x) returns A*x. For this syntax, gmres does not restart; the maximum number of iterations is min(n,10).
- Parameterizing Functions explains how to provide additional parameters to the function afun, as well as the preconditioner function mfun described below, if necessary.
- gmres(A,b,restart) restarts the method every restart inner iterations. The maximum number of outer iterations is min(n/restart,10). The maximum number of total iterations is restart*min(n/restart,10). If restart is n or [], then gmres does not restart and the maximum number of total iterations is min(n,10).
- gmres(A,b,restart,tol) specifies the tolerance of the method. If tol is [], then gmres uses the default, 1e-6.

Matlab函数-gmres

- gmres(A,b,restart,tol,maxit) specifies the maximum number of outer iterations, i.e., the total number of iterations does not exceed restart*maxit. If maxit is [] then gmres uses the default, min(n/restart,10). If restart is n or [], then the maximum number of total iterations is maxit (instead of restart*maxit).
- gmres(A,b,restart,tol,maxit,M) and gmres(A,b,restart,tol,maxit,M1,M2) use preconditioner M or M = M1*M2 and effectively solve the system inv(M)*A*x = inv(M)*b for x. If M is [] then gmres applies no preconditioner. M can be a function handle mfun such that mfun(x) returns M\x.
- gmres(A,b,restart,tol,maxit,M1,M2,x0) specifies the first initial guess. If x0 is [], then gmres uses the default, an all-zero vector.

[x,flag] = gmres(A,b,...) also returns a convergence flag:

flag = 0 gmres converged to the desired tolerance tol within maxit outer iterations.

flag = 1 gmres iterated maxit times but did not converge.

flag = 2 Preconditioner M was ill-conditioned.

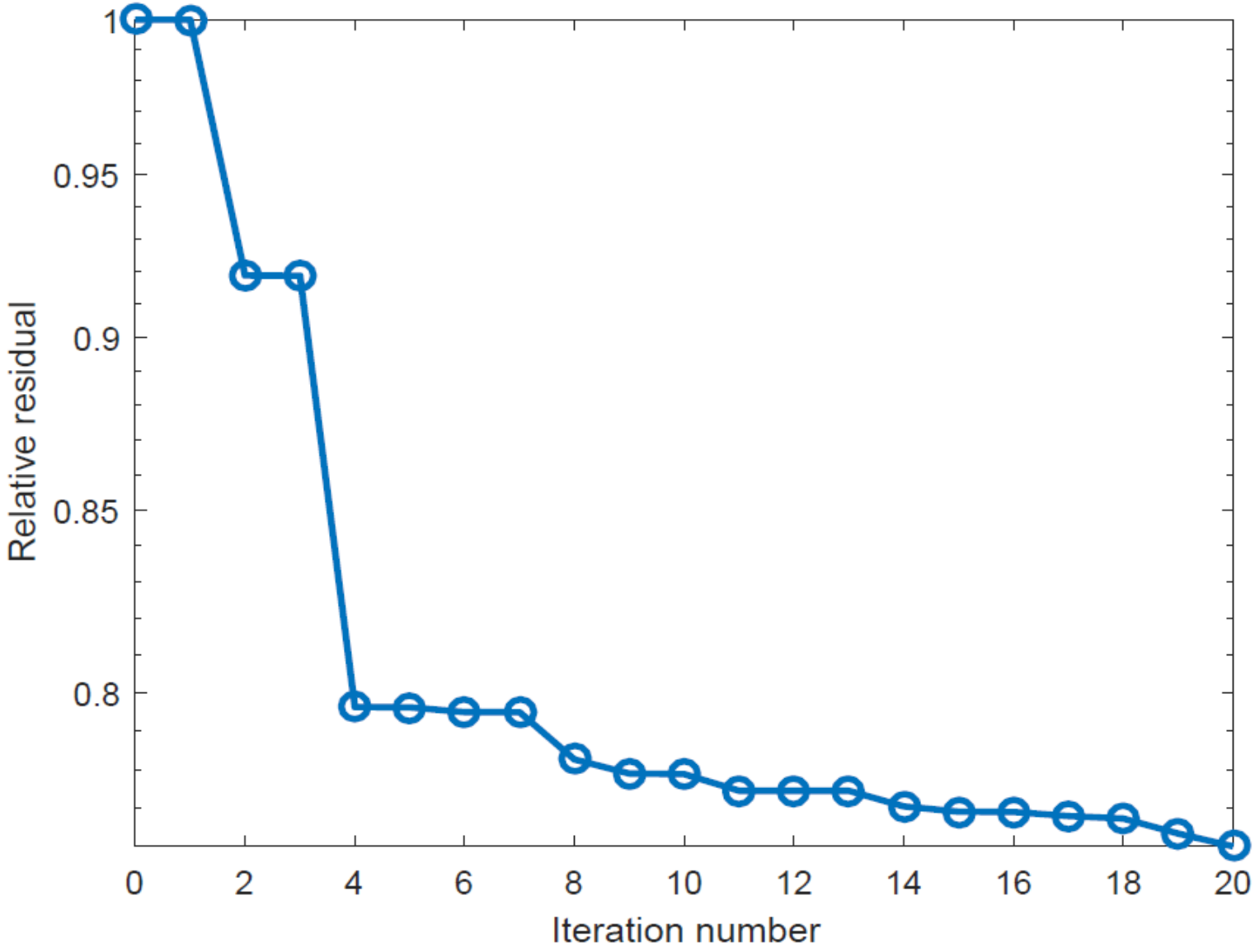
flag = 3 gmres stagnated. (Two consecutive iterates were the same.)

Whenever flag is not 0, the solution x returned is that with minimal norm residual computed over all the iterations. No messages are displayed if the flag output is specified.

- [x,flag,relres] = gmres(A,b,...) also returns the relative residual norm(b-A*x)/norm(b). If flag is 0, relres <= tol. The third output, relres, is the relative residual of the preconditioned system.
- [x,flag,relres,iter] = gmres(A,b,...) also returns both the outer and inner iteration numbers at which x was computed, where 0 <= iter(1) <= maxit and 0 <= iter(2) <= restart.
- [x,flag,relres,iter,resvec] = gmres(A,b,...) also returns a vector of the residual norms at each inner iteration. These are the residual norms for the preconditioned system.

gmres例子1

```
load west0479;
A = west0479;
%Set the tolerance and maximum number of iterations.
tol = 1e-12;
maxit = 20;
%Define b so that the true solution is a vector of all ones.
b = full(sum(A,2));
[x0,fl0,rr0,it0,rv0] = gmres(A,b,[],tol,maxit);
%fl0 is 1 because gmres does not converge to the requested tolerance 1e-12 within the
requested 20 iterations. The best approximate solution that gmres returns is the last one (as
indicated by it0(2) = 20). MATLAB stores the residual history in rv0.
%Plot the behavior of gmres.
semilogy(0:maxit,rv0/norm(b),'-o');
xlabel('Iteration number');
ylabel('Relative residual');
```


Relative residual
Iteration number
1
0.95
0.9
0.85
0.8
0
2
4
6
8
10
12
14
16
18
20

%*Use ilu to form the preconditioner, since A is nonsymmetric.*

[L,U] = ilu(A,struct('type','ilutp','droptol',1e-5));

Error using ilu There is a pivot equal to zero. Consider decreasing the drop tolerance or consider using the 'udiag' option

%*As indicated by the error message, try again with a reduced drop tolerance.*

[L,U] = ilu(A,struct('type','ilutp','droptol',1e-6));

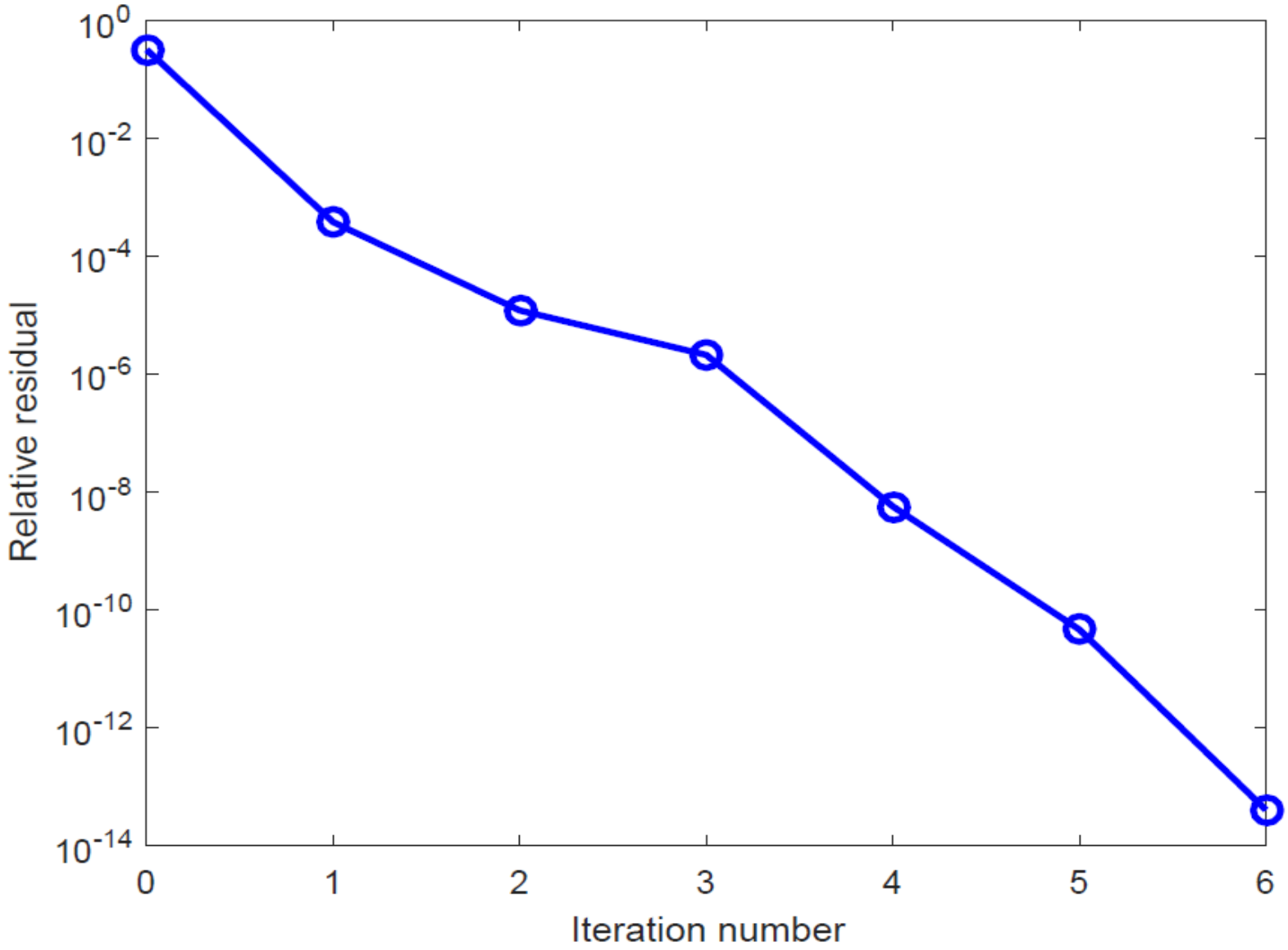
[x1,fl1,rr1,it1,rv1] = gmres(A,b,[],tol,maxit,L,U);

%*fl1 is 0 because gmres drives the relative residual to 9.5436e-14 (the value of rr1). The relative residual is less than the prescribed tolerance of 1e-12 at the sixth iteration (the value of it1(2)) when preconditioned by the incomplete LU factorization with a drop tolerance of 1e-6. The output, rv1(1), is norm(M\b), where M = L*U. The output, rv1(7), is norm(U\(L\(b-A*x1))).*

semilogy(0:it1(2),rv1/norm(b),'-o');

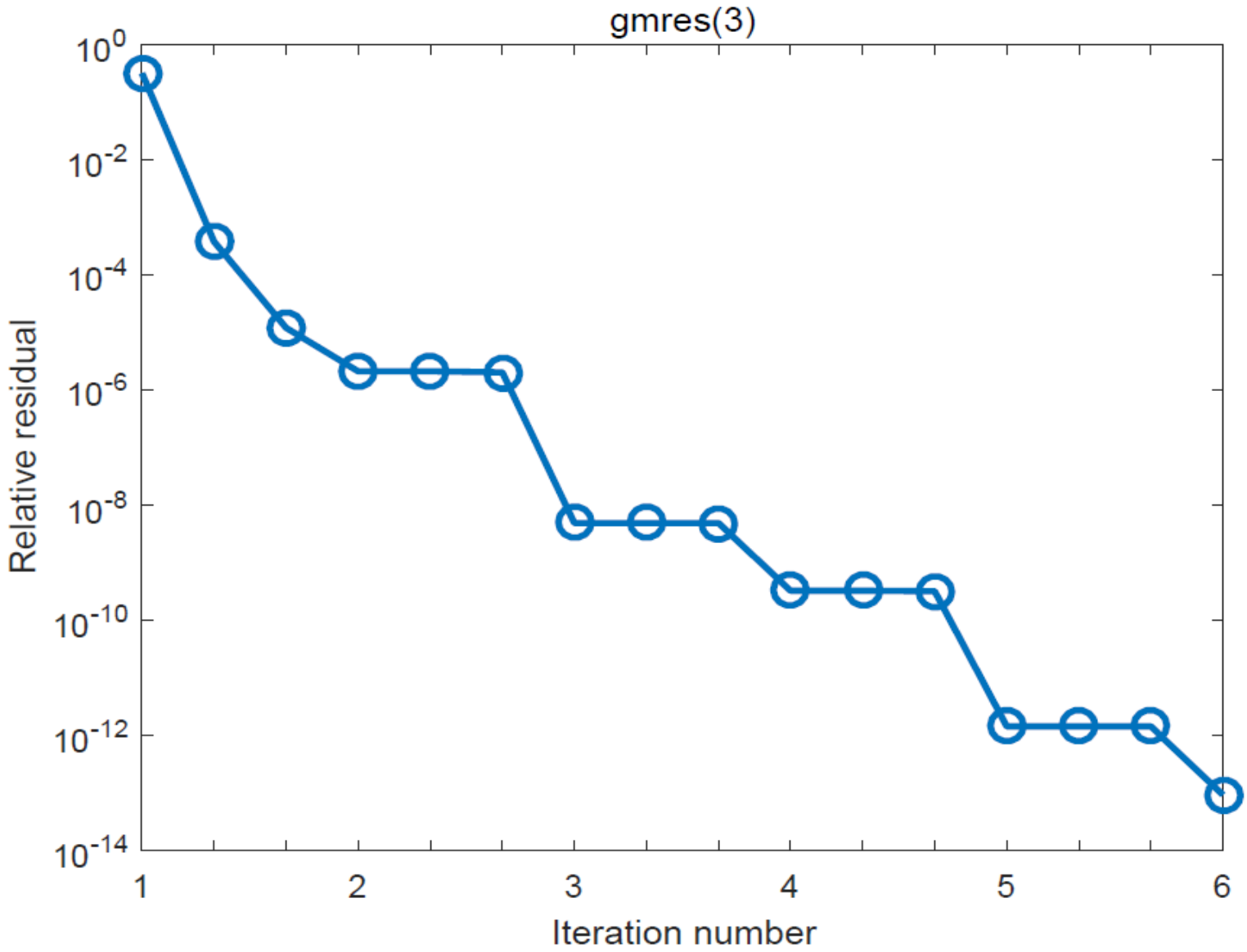
xlabel('Iteration number');

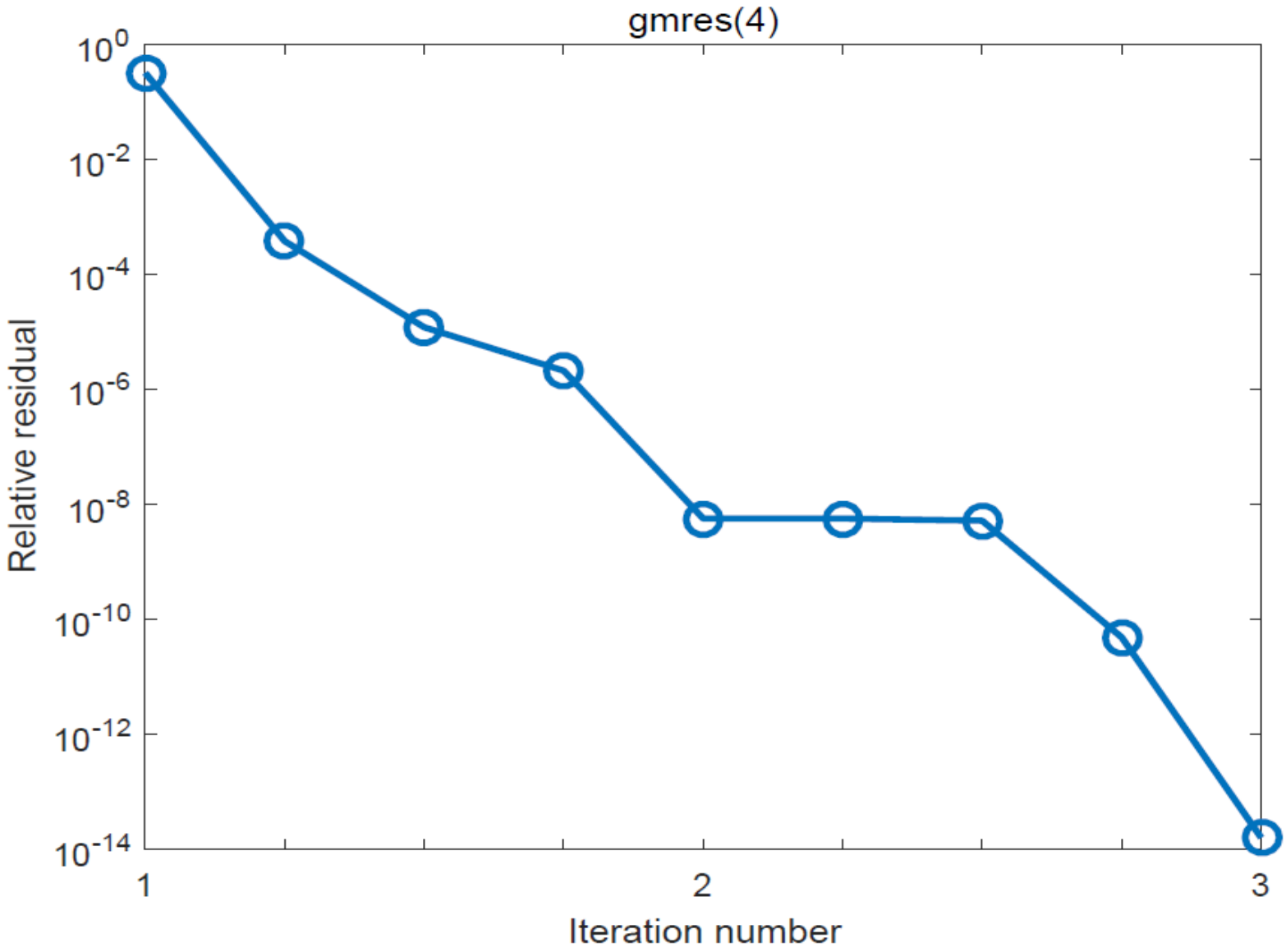
ylabel('Relative residual');

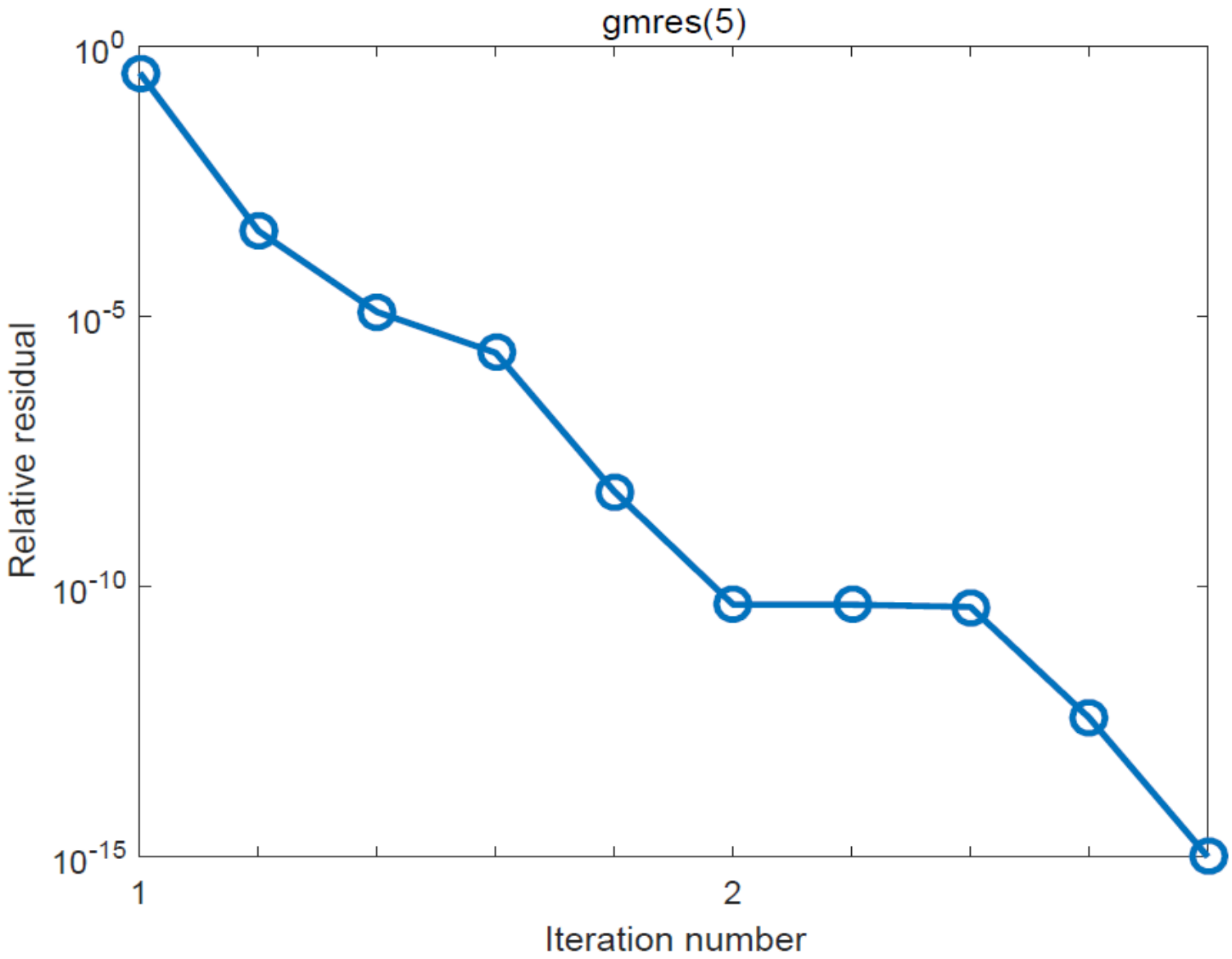

Relative residual
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
0
1
2
3
4
5
6
Iteration number

Using a Preconditioner with Restart

```
load west0479;
A = west0479;
%Define b so that the true solution is a vector of all ones.
b = full(sum(A,2));
%Construct an incomplete LU preconditioner as in the previous example.
[L,U] = ilu(A,struct('type','ilutp','droptol',1e-6));
%Execute gmres(3), gmres(4), and gmres(5)
tol = 1e-12;
maxit = 20;
re3 = 3;
[x3,fl3,rr3,it3,rv3] = gmres(A,b,re3,tol,maxit,L,U);
re4 = 4;
[x4,fl4,rr4,it4,rv4] = gmres(A,b,re4,tol,maxit,L,U);
re5 = 5;
[x5,fl5,rr5,it5,rv5] = gmres(A,b,re5,tol,maxit,L,U);
```


gmres(3)
Relative residual
Iteration number
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
1
2
3
4
5
6


gmres(4)
Relative residual
Iteration number
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
1
2
3


gmres(5)
10^{0}
10^{-5}
10^{-10}
10^{-15}
Relative residual
1
2
Iteration number

MINRES

若$A = A^T$，此时用Lanczos过程可把矩阵A化为三对角阵T_m。
在$\boldsymbol{x}_0 + \mathcal{K}_m(\boldsymbol{r}_0, A)$上求近似解

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m \boldsymbol{y}_m$$

使得

$$\|\boldsymbol{r}_m\|_2 = \|\boldsymbol{b} - A\boldsymbol{x}_m\|_2 = \min_{\boldsymbol{x} \in \boldsymbol{x}_0 + \mathcal{K}_m(\boldsymbol{r}_0, A)} \|\boldsymbol{b} - A\boldsymbol{x}\|_2$$

此时方法称为最小残量法（MINRES）。
随着m增大，$\|\boldsymbol{r}_m\|_2$一定单调下降，因为$\mathcal{K}_i(\boldsymbol{r}_0, A) \subset \mathcal{K}_j(\boldsymbol{r}_0, A)$，若$i < j$。
当$m = n$时，$\boldsymbol{x}_0 + \mathcal{K}_n(\boldsymbol{r}_0, A) = \mathbb{R}^n$，则

$$\|\boldsymbol{r}_n\|_2 = 0, \ \boldsymbol{x}_n = \boldsymbol{x}^*.$$

因此

$$0 = \|\boldsymbol{r}_n\|_2 \leq \|\boldsymbol{r}_{n-1}\|_2 \leq \cdots \leq \|\boldsymbol{r}_0\|_2.$$

由Lanczos过程知，

$$AV_m = V_m T_m + \beta_m v_{m+1} \boldsymbol{e}_m^T = V_{k+1} \overline{T}_m$$

其中T_m是对角阵

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix}$$

则有

$$\begin{aligned} \|\boldsymbol{r}_m\|_2 &= \|\boldsymbol{b} - A(\boldsymbol{x}_0 + V_m \boldsymbol{y}_m)\| = \|\boldsymbol{r}_0 - AV_m \boldsymbol{y}_m\|_2 \\ &= \min_{\boldsymbol{y} \in \mathbb{R}^m} \|\boldsymbol{r}_0 - AV_m \boldsymbol{y}\|_2 = \min_{\boldsymbol{y} \in \mathbb{R}^m} \|\boldsymbol{r}_0 - V_{m+1} \overline{T}_m \boldsymbol{y}\|_2 \\ &= \min_{\boldsymbol{y} \in \mathbb{R}^m} \|V_{m+1}(\beta \boldsymbol{e}_1 - \overline{T}_m \boldsymbol{y})\|_2 \\ &= \min_{\boldsymbol{y} \in \mathbb{R}^m} \|\beta \boldsymbol{e}_1 - \overline{T}_m \boldsymbol{y}\|_2 \end{aligned}$$

定理

若*Lanczos*过程第m步中断，则*MINRES*找到了准确解。

定理 (残差的计算)

对*MINRES*方法，令

$$\overline{T}_m = Q\begin{pmatrix} R \\ O \end{pmatrix},\ \bar{\boldsymbol{g}}_m = Q^T\beta\boldsymbol{e}_1 = \begin{pmatrix} * \\ \gamma_{m+1} \end{pmatrix}$$

则$\|\boldsymbol{r}_m\|_2 = |\gamma_{m+1}|$。

定理

设A的特征值为$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$，则*MINRES*方法的残量

$$\|\boldsymbol{r}_m\|_2 = \min_{p\in\mathcal{P}_m, p(0)=1} \|p(A)\boldsymbol{r}_0\|_2 \leq \|\boldsymbol{r}_0\|_2 \min_{p\in\mathcal{P}_m, p(0)=1} \max_{1\leq i\leq n} |p(\lambda_i)|$$

算法 (MINRES算法)

1. 选$\boldsymbol{r}_0 \in \mathbb{R}^n$，计算$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, \beta = \|\boldsymbol{r}_0\|_2, \boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$
2. 用*Lanczos*过程计算$\mathcal{K}_m(\boldsymbol{r}_0, A)$的标准正交基$V_m$和$\overline{T}_m$
3. 计算$\min \|\beta \boldsymbol{e}_1 - \overline{T}_m \boldsymbol{y}\|_2$的解$\boldsymbol{y}_m$
4. 若$\|\boldsymbol{r}_m\|_2 = |\gamma_{m+1}| \leq \epsilon$，则停止，否则转步2，保留已有的$V_m$和$\overline{T}_m$

注记

数值计算中，在有限精度下，*MINRES*的性态与理论上的方法差别很大。因为计算出$\{\boldsymbol{v}_i\}_{i=1}^m$会很快失去正交性，甚至线性相关，即过程不稳定。

结论：仍然收敛，但速度缓慢。

Matlab函数–minres

1. x = minres(A,b) attempts to find a minimum norm residual solution x to the system of linear equations A*x=b. The n-by-n coefficient matrix A must be symmetric but need not be positive definite. It should be large and sparse. The column vector b must have length n. You can specify A as a function handle, afun, such that afun(x) returns A*x.
2. minres(A,b,tol) specifies the tolerance of the method. If tol is [], then minres uses the default, 1e-6.
3. minres(A,b,tol,maxit) specifies the maximum number of iterations. If maxit is [], then minres uses the default, min(n,20).
4. minres(A,b,tol,maxit,M) and minres(A,b,tol,maxit,M1,M2) use symmetric positive definite preconditioner M or M = M1*M2 and effectively solve the system inv(sqrt(M))*A*inv(sqrt(M))*y = inv(sqrt(M))*b for y and then return x = inv(sqrt(M))*y. If M is [] then minres applies no preconditioner. M can be a function handle mfun, such that mfun(x) returns M\x.
5. minres(A,b,tol,maxit,M1,M2,x0) specifies the initial guess. If x0 is [], then minres uses the default, an all-zero vector.

Using minres with a Matrix Input

```
n = 100; on = ones(n,1);
A = spdiags([-2*on 4*on -2*on],-1:1,n,n);
b = sum(A,2);
tol = 1e-10;
maxit = 50;
M1 = spdiags(4*on,0,n,n);
x = minres(A,b,tol,maxit,M1);
minres converged at iteration 49 to a solution with relative residual 4.7e-014
```

Using minres instead of pcg

Use a symmetric indefinite matrix that fails with pcg.

```
A = diag([20:-1:1, -1:-1:-20]);
b = sum(A,2);  % The true solution is the vector of all ones.
x = pcg(A,b); % Errors out at the first iteration.
pcg stopped at iteration 1 without converging to the desired tolerance 1e-006
because a scalar quantity became too small or too large to continue
computing. The iterate returned (number 0) has relative residual 1
```

However, minres can handle the indefinite matrix A.

```
x = minres(A,b,1e-6,40);
minres converged at iteration 39 to a solution with relative residual 1.3e-007
```

习题：假设在GMRES方法中，Arnoldi过程开始时取$\boldsymbol{v}_1 = A\boldsymbol{v}_0/\|A\boldsymbol{v}_0\|_2$，其中$\boldsymbol{v}_0 = \boldsymbol{r}_0$。此时可用正交化方法生成一组正交基$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_{m-1}\}$。则此时近似解$\boldsymbol{x}_m$可用基底$\{\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_{m-1}\}$表出。即$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}_m$，其中$V_m$的列向量为$\boldsymbol{v}_i, 0 \le i \le m-1$。

1. 试证明求$\boldsymbol{y}_m$的最小二乘问题的系数矩阵不再是Hessenberg矩阵，而是上三角矩阵。
2. 证明此时残量$\boldsymbol{r}_k$和$\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_{k-1}$正交。
3. 导出不需要计算近似解$\boldsymbol{x}_m$就能计算残量$\boldsymbol{r}_m$的范数的公式，并写出完整的GMRES算法。

大型稀疏矩阵的特征值问题

Krylov子空间方法可以用来求解大型稀疏矩阵的特征值。
设$A \in \mathbb{R}^{n\times n}$，$\boldsymbol{x}_0 \in \mathbb{R}^n$，则相应的Krylov序列为$\boldsymbol{x}_k = A\boldsymbol{x}_{k-1}$，
对$k = 1, 2, \cdots, n$，相应的Krylov矩阵为

$$\begin{aligned} K_k &= [\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{k-1}] \\ &= [\boldsymbol{x}_0, A\boldsymbol{x}_0, \cdots, A^{(k-1)}\boldsymbol{x}_0], \end{aligned}$$

相应的Krylov子空间$\mathcal{K}_k = \mathrm{span}(K_k)$。特别当$k = n$时有

$$\begin{aligned} AK_n &= [A\boldsymbol{x}_0, \cdots, A\boldsymbol{x}_{n-2}, A\boldsymbol{x}_{n-1}] \\ &= [\boldsymbol{x}_1, \cdots \boldsymbol{x}_{n-1}, \boldsymbol{x}_n] \\ &= K_n[\boldsymbol{e}_2, \boldsymbol{e}_3, \cdots, \boldsymbol{e}_n, \boldsymbol{a}]. \end{aligned}$$

其中

$$\boldsymbol{a} = K_n^{-1}\boldsymbol{x}^{(n)}$$

假定K_n非奇异则

$$K_n^{-1} A K_n = C_n$$

所以A和C_n相似，而C_n为上Hessenberg矩阵。这样用矩阵–向量乘积得到了一种将矩阵相似化为Hessenberg矩阵的方法。
但是K_n的列连续地收敛到A的主特征向量使得K_n是$\mathcal{K}_n$的病态基底，但可以通过计算QR分解

$$Q_n R_n = K_n$$

进行校正，这样$n \times n$的矩阵Q_n的列构成了$\mathcal{K}_n$的正交基，所以

$$\begin{aligned} Q_n^T A Q_n &= (K_n R_n^{-1})^{-1} A K_n R_n^{-1} \\ &= R_n K_n^{-1} A K_n R_n^{-1} \\ &= R_n C_n R_n^{-1} \\ &= H \end{aligned}$$

由于上Hessenberg矩阵被上三角的矩阵作相似变换后保持上Hessenberg矩阵的形式，所以H为一个与A相似的上Hessenberg矩阵。为了每次计算出的是

$$Q_n = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_n]$$

的一个列，从

$$AQ_n = Q_n H$$

的第j列就可以得到$\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_n$的递推公式

$$A\boldsymbol{q}_k = h_{1k}\boldsymbol{q}_1 + \cdots + h_{kk}\boldsymbol{q}_k + h_{k+1,k}\boldsymbol{q}_{k+1}, \tag{0.2}$$

在(0.2)式中左乘q_j^T并利用正交性有$h_{jk} = \boldsymbol{q}_j^T A\boldsymbol{q}_k, j = 1, 2, \cdots, k$，则有

Arnoldi方法–Gram-Schmidt正交化：

1. 给定非零初始向量$x^{(1)}$满足$\|\boldsymbol{x}^{(1)}\|_2 = 1,$
2. 对$j = 1, 2, \cdots, m,$计算

$$\begin{aligned} h_{ij} &= (A\boldsymbol{x}_i, \boldsymbol{x}_j), \quad i = 1, 2, \cdots, j, \\ \boldsymbol{w}_j &= A\boldsymbol{x}_j - \sum_{i=1}^{j} h_{ij}\boldsymbol{x}_i, \\ h_{j+1,j} &= \|\boldsymbol{w}_j\|_1, \\ \boldsymbol{x}_{j+1} &= \boldsymbol{w}_j / h_{j+1,j} \end{aligned}$$

定理

向量$\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_k$构成子空间$\mathcal{K}_k = \text{span}(\boldsymbol{q}_1, A\boldsymbol{q}_1, \cdots, A\boldsymbol{q}_{k-1})$的一组正交基。

证明:从Arnoldi算法的构造过程知$\boldsymbol{q}_i, i = 1, \cdots, k$是相互正交的。而它们可以 张成$\mathcal{K}_k$是因为每个$\boldsymbol{q}_j$都可以表示成$P_{j-1}(A)\boldsymbol{q}_1$的形式，这里$P_{j-1}$是次数不超过$j-1$的多项式。 用归纳法来证明：
$j = 1$时$\boldsymbol{q}_1 = P_0(A)\boldsymbol{q}_1 = \boldsymbol{q}_1, P_0(t) = 1$，结论显然成立；假设对所有$\leq j$都成立，考虑$\boldsymbol{q}_{j+1}$有

$$h_{j+1,j+1}\boldsymbol{q}_{j+1} = A\boldsymbol{q}_j - \sum_{i=1}^{j} h_{ij}\boldsymbol{q}_i = AP_{j-1}(A)\boldsymbol{q}_1 - \sum_{i=1}^{j} h_{ij}P_{i-1}(A)\boldsymbol{q}_1.$$

这说明$\boldsymbol{q}_{j+1}$可以表示成$P_j(A)\boldsymbol{q}_1$，这里P_j是次数不超过j的多项式，定理证毕！

定理

令$Q_k = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_k] \in \mathbb{R}^{n\times k}$，$H_k \in \mathbb{R}^{k\times k}$是由*Arnoldi*算法定义的上*Hessenberg*矩阵，则有下面的关系成立：

$$AQ_k = Q_k H_k + h_{k+1,k}\boldsymbol{q}_{k+1}\boldsymbol{e}_k^T, \tag{0.3}$$

$$Q_k^T A Q_k = H_k, \tag{0.4}$$

证明:式(0.3)是基于下面的式子

$$A\boldsymbol{q}_j = \sum_{i=1}^{j+1} h_{ij}\boldsymbol{q}_i, \quad i = 1, 2, \cdots, k.$$

而上面的式子可以从Arnoldi算法直接得出。而(0.4)只需在(0.3)两边同时左乘Q_k^T然后利用Q_k的正交性即得！

求特征值问题

$$A\boldsymbol{u} = \lambda \boldsymbol{u}. \tag{0.5}$$

想在子空间$\mathcal{K}$中求$\tilde{\lambda} \in \mathbb{R}, \tilde{\boldsymbol{u}} \in \mathcal{K}$是$\lambda$和$\boldsymbol{u}$的近似值，亦即下面的Galerkin条件满足

$$A\tilde{\boldsymbol{u}} - \tilde{\lambda}\tilde{\boldsymbol{u}} \perp \mathcal{K} \Leftrightarrow (A\tilde{\boldsymbol{u}} - \tilde{\lambda}\tilde{\boldsymbol{u}}, \boldsymbol{v}) = 0, \ \forall \boldsymbol{v} \in \mathcal{K}.$$

假设$Q = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_k]$的列向量是$\mathcal{K}$的正交基，则可以把问题变到这组正交基上数值求解。令

$$\tilde{\boldsymbol{u}} = Q\boldsymbol{y} \Rightarrow (AQ\boldsymbol{y} - \tilde{\lambda}Q\boldsymbol{y}, \boldsymbol{q}_j) = 0, j = 1, \cdots, k.$$

所以$\boldsymbol{y}$和$\tilde{\lambda}$必须满足

$$B_k\boldsymbol{y} = \tilde{\lambda}\boldsymbol{y}, \ \text{其中 } B_k = Q^T AQ.$$

其实B_k是A在子空间$\mathcal{K}$上的投影A_k在基Q下的表示！

算法 (Rayleigh–Ritz过程)

1. 计算子空间$\mathcal{K}$的正交基$\{\boldsymbol{q}_i\}_{i=1}^k$，令$Q=[\boldsymbol{q}_1,\boldsymbol{q}_2,\cdots,\boldsymbol{q}_k]$;
2. 计算$B_k=Q^TAQ$;
3. 计算B_k的特征值，选出需要的$m\le k$个$\tilde{\lambda}_i, i=1,\cdots,m$;
4. 计算特征向量B_k的特征向量$\tilde{\lambda}_i$对应的特征向量$\boldsymbol{y}_i, 1\le i\le m$；然后计算$A$的特征向量$\tilde{\boldsymbol{u}}_i=Q\boldsymbol{y}_i, i=1,\cdots,m$。

上面的Rayleigh–Ritz过程中计算得到的$\tilde{\lambda}_i$称为Ritz值，而$\tilde{\boldsymbol{u}}_i$称为Ritz特征向量。

注记

实际上*Arnoldi*过程就是实现*Rayleigh–Ritz*过程的第一步基构造子空间的正交基。

Arnoldi/Lanczos逼近问题：

找一个首1的多项式$p_m \in \mathbb{P}_m$使得$\|p_m(A)\boldsymbol{r}_0\|_2$最小。 (*)

上面极小化问题的解由如下定理给出：

定理

若*Arnoldi/Lanczos*过程不中断，则逼近问题(*)有唯一解p_m，而p_m是H_m的特征多项式。

注记

H_m是A的一种近似，满足$\chi_m(H_m)$是$\chi(A)$的近似；这个近似在(*)意义下最优。因此可以通过*Arnoldi*迭代用$\sigma(H_m)$去近似$\sigma(A)$。而H_m的特征值也叫着*Ritz values*。

问题

若*Arnoldi*过程在第j步中断，即$h_{j+1,j} = 0$，则方法找到了准确的特征值和特征向量。

在Arnoldi算法中

$$Q_k = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_k] \in \mathbb{R}^{n\times k}$$

含有前k个Arnoldi向量$\boldsymbol{q}_j$，而

$$U_k = [\boldsymbol{q}_{k+1}, \cdots, \boldsymbol{q}_n] \in \mathbb{R}^{n\times(n-k)}$$

含有其余的Arnoldi向量，因此$Q_n = [Q_k, U_k]$，则

$$H = Q_n^T A Q_n = \begin{bmatrix} Q_k^T \\ U_k^T \end{bmatrix} A[Q_k, U_k] = \begin{bmatrix} Q_k^T A Q_k & Q_k^T A U_k \\ U_k^T A Q_k & U_k^T A U_k \end{bmatrix} = \begin{bmatrix} H_k & M \\ \tilde{H}_k & N \end{bmatrix}$$

其中H_k是上Hessenberg矩阵，H_k的特征值就是Ritz值，而$Q_k|y$是Ritz向量，此处$|y$是H_k的特征向量。当然需要其它的方法计算H_k的特征值和特征向量，如QR 迭代！如果$k \ll n$ 时经过几步迭代就能得到对A的末端特征值的很好的近似！

实际计算的时候必须能够很廉价的估计出Arnoldi过程的残量。

定理

设$\boldsymbol{y}_i$是H_k的特征值$\tilde{\lambda}_i$的特征向量，$\tilde{\boldsymbol{u}}_i = Q_k\boldsymbol{y}_i$是*Ritz*向量，则

$$(A - \tilde{\lambda}_i I)\tilde{\boldsymbol{u}}_i = h_{k+1,k}\boldsymbol{e}_k^T\boldsymbol{y}_i\boldsymbol{q}_{k+1}$$

因此

$$\|(A - \tilde{\lambda}_i I)\tilde{\boldsymbol{u}}_i\|_2 = h_{k+1,k}|\boldsymbol{e}_k^T\boldsymbol{y}_i|.$$

证明： 在(0.3)两端乘于$\boldsymbol{y}_i^T$：

$$AQ_k\boldsymbol{y}_i = Q_kH_k\boldsymbol{y}_i + h_{k+1,k}\boldsymbol{q}_{k+1}e_k^T\boldsymbol{y}_i = \tilde{\lambda}_iQ_k\boldsymbol{y}_i + h_{k+1,k}\boldsymbol{e}_k^T\boldsymbol{y}_i\boldsymbol{q}_{k+1}$$

因此

$$AQ_k\boldsymbol{y}_i - \tilde{\lambda}_iQ_k\boldsymbol{y}_i = h_{k+1,k}\boldsymbol{e}_k^T\boldsymbol{y}_i\boldsymbol{q}_{k+1}.$$

定理说明残量的模为特征向量$\boldsymbol{y}_i$的最后一个分量和$h_{k+1,k}$的乘积！

Arnoldi方法的计算量

Arnoldi方法的计算量和存储量随着迭代次数的增加而大幅增加。

- [存储量]

  需要存储k个长度为n的向量和一个$k \times k$的矩阵，总的存储量为$nk + k^2/2$。

- [计算量]

  在第k步迭代时需要计算$A\boldsymbol{q}_k$，计算量为$2 \times Nz$，这里Nz为A的 非零元素的个数，而和k个向量正交化的过程需要$4(k+1)n$的计算量。计算Ritz值和Ritz向量需要$O(k^3)$的计算量。

因此随着k的增大，方法的计算量也会快速增加，此时可以考虑重启的Arnoldi方法。

算法 (Iterative Arnoldi)

1. **Start**: 选取初始向量$\boldsymbol{v}_1$及维数m。
2. **Iterative:** 运行m步*Arnoldi*算法。
3. **Restart:** 计算最右端的特征值$\lambda_1^{(m)}$对应的特征向量$\boldsymbol{u}_1^{(m)}$。若满足要求，迭代终止，或者转向第二步。

注记

由于上面的原因，实际的*Arnoldi*过程先进行几步迭代，然后用得到的信息精心构造出一个新的初始向量，使得它的分量进一步接近所要求的特征向量。几次循环后通常可得满足精度要求 的末端特征值及相应的特征向量。

如果A是对称矩阵，则Arnoldi过程的计算量和存储量会显著降低。特别地此时矩阵 H_k是三对角的(通常写为T_k)，向量$\boldsymbol{q}_k$之间的关系为三项递归

定理

若对*Hermite*矩阵A用*Arnoldi*方法，则算法生成的系数$h_{ij} \in \mathbb{R}$且有

$$\begin{aligned} h_{ij} &= 0, \text{ for } 1 \le j < j-1, \\ h_{i,j+1} &= h_{j+1,j}, \ j = 1, 2, \cdots, m \end{aligned}$$

亦即由*Arnoldi*过程得到的H_m是实对称的三对角阵。

证明： 因为$H_m = V_m^H A V_m$是一个Hermite阵，而且也是一个上Hessenberg阵，所以H_m必然是一个三对角的 Hermite矩阵。另外$h_{j+1,j} = \|\omega_j\|_2 \in \mathbb{R}$,而$h_{jj} = (A\boldsymbol{v}_j, \boldsymbol{v}_j) \in \mathbb{R}$。

实际上在Arnoldi过程中取

$$\alpha_j \equiv h_{jj}, \quad \beta_j \equiv h_{j-1,j}.$$

可得

算法 (Lanczos迭代)

1. 取初始向量$\boldsymbol{q}_1$，$\|\boldsymbol{q}_1\|_2 = 1$，$\boldsymbol{u}_1 = A\boldsymbol{q}_1$
2. 对$k = 1, 2, \cdots, j$

$$\begin{aligned} \alpha_k &= (\boldsymbol{u}_k, \boldsymbol{q}_k) \\ \boldsymbol{w}_k &= \boldsymbol{u}_k - \alpha_k \boldsymbol{q}_k \\ \beta_k &= \|\boldsymbol{w}_k\|_2, \text{若} \beta_k = 0 \text{ 中断} \\ \boldsymbol{q}_{k+1} &= \boldsymbol{w}_k / \beta_k \\ \boldsymbol{u}_{k+1} &= A\boldsymbol{q}_{k+1} - \beta_k \boldsymbol{q}_k \end{aligned}$$

这里

$$T_n = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix}$$

亦即α_k和β_k分别是三对角对称阵T_k的对角元和次对角元。$\beta_k = 0$时说明Lanczos 迭代中断，但此时得到了一个不变子空间，由T_k得到的Ritz值和Ritz向量就是A的特征值和特征向量。

注记

在实际计算时，舍入误差可能使计算所得的*Lanczos*向量q_k的正交性有所损失，这是一个潜在的问题。 这个问题可以通过将向量重新正交化来克服，但要增加一定的计算成本。如果算法产生的特征值的近似程度很好，上述问题可不考虑， 但会产生重特征值。

注记

*Arnoldi*和*Lanczos*方法都能迅速的对谱集边缘的特征值作出近似。如果要求的使谱集内部的特征值，如接近σ的特征值，则可以对$(A-\sigma I)^{-1}$用这两种方法迭代，即相当于求解形如$(A-\sigma I)\boldsymbol{x}=\boldsymbol{y}$的方程组。此时矩阵的内部特征值恰好使这个位移矩阵的末端特征值；这种“反位移”技巧可以迅速的求得内部特征值。

参考书: Youcef Saad, Numerical Methods For Large Eigenvalue Problems, Manchester University Press, 1992.

Matlab求特征值命令eigs

- d = eigs(A) returns a vector of the six largest magnitude eigenvalues of matrix A.
- d = eigs(A,k) returns the k largest magnitude eigenvalues.
- d = eigs(A,k,sigma) returns k eigenvalues based on the value of sigma. For example, eigs(A,k,'sm') returns the k smallest magnitude eigenvalues.
- d = eigs(A,k,sigma,opts) additionally specifies options using a structure.
- d = eigs(A,B,___) solves the generalized eigenvalue problem A*V = B*V*D.
- d = eigs(Afun,n,) specifies a function handle, Afun, instead of a matrix, A.
- [V,D] = eigs(___) returns diagonal matrix D containing the eigenvalues on the main diagonal, and matrix V whose columns are the corresponding eigenvectors.
- [V,D,flag] = eigs(___) also returns a convergence flag. If flag is 0, then all the eigenvalues converged.

计算稀疏矩阵最大和最小特征值

计算稀疏矩阵的最大特征值

```
A = delsq(numgrid('C',15));
d = eigs(A)
d =
7.8666
7.7324
7.6531
7.5213
7.4480
7.3517
```

计算稀疏矩阵的最小特征值

```
A = delsq(numgrid('C',15));
d = eigs(A,5,'sm')
d =
0.5520
0.4787
0.3469
0.2676
0.1334
```

具体可以在Matlab命令窗口敲： help eigs 看帮助。

eigs命令中'sa'和'sm'的区别

首先生成一个对称正定的稀疏矩阵

```
A = delsq(numgrid('C', 150));
```

用'sa'计算最小的6个代数特征，算法基于矩阵A的Krylov子空间方法

```
tic
d = eigs(A, 6, 'sa')
toc
d =
0.0013
0.0025
0.0033
0.0045
0.0052
0.0063
Elapsed time is 2.948806 seconds.
```

用'sm'计算，用的是基于A^{-1}的Krylov子空间方法。

```
tic
dsm = eigs(A, 6, 'sm')
toc
dsm =
0.0063
0.0052
0.0045
0.0033
0.0025
0.0013
Elapsed time is 0.281773 seconds.
```