1.2 迭代法

直接法这在n比较小的时候还比较合适,但是实际问题往往要求解很大的n的矩阵,而且往往含有大量的0元素,在用直接法时就会耗费大量的时 间和存储单元,迭代法具有的特点是速度快.

1.2.1 解线性方程组的经典迭代法

这就需要将线性方程组进行改写

$$\mathrm{x} = G\mathrm{x} + \mathrm{b}$$

那么,当$x^{(0)}$给定后,可以利用迭代格式

$$\mathrm{x}^{(k)} = G\mathrm{x}^{(k-1)} + \mathrm{b}, \quad \forall k = 1, 2, \cdots, n.$$

得到序列$\{x^{(k)}\}_{k=0}^{n}$.

- 序列的收敛只与算子G有关,而与初值$\mathrm{x}^{(0)}$的选取无关!
- 收敛性：谱半径$\rho(G) < 1$

定理 1.2.1 (一半迭代法收敛性). 迭代矩阵普半径 < 1

Jacobi

为纪念普鲁士著名数学家雅可比

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

对Ax=b

$$A = \begin{bmatrix} 0 & & & & \\ a_{21} & 0 & & & \\ a_{31} & a_{32} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ & 0 & a_{23} & \cdots & a_{2n} \\ & & 0 & \cdots & a_{3n} \\ & & & \ddots & \vdots \\ & & & & 0 \end{bmatrix}$$

下三角阵　　对角阵　　上三角阵

$$= L + D + U$$

$\because a_{ii} \neq 0$(Jacobi假设)　　$\therefore D$可逆

进一步，有：　$(L+D+U)\,x = b$

$$Dx = -(L+U)x + b$$

$$x = -D^{-1}(L+U) + D^{-1}b \quad \rightarrow$$理论分析

Gauss-Sediel

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

矩阵G对角占优时两类迭代均收敛。但可以构造不同的例子显示：Gauss-Seidel法收敛时，Jacobi法可能不收敛；另一方面，Jacobi法收敛时， Gauss-Seidel法也可能不收敛!

定理 1.2.2 (对称正定矩阵收敛). content

松弛(Relaxation)迭代

在Gauss-Sediel基础上的一种加权修正,分两步

1.

$$\tilde{x}_i^{(k)} = \frac{1}{G_{ii}}\big(g_i - \sum_{j=1}^{i-1} G_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} G_{ij}x_j^{(k-1)}\big)$$

2.

$$\begin{aligned} x_i^{(k)} &= x_i^{(k-1)} + \omega(\tilde{x}_i^{(k)} - x_i^{(k-1)}) \\ &= (1-\omega)x_i^{(k-1)} + \omega\tilde{x}_i^{(k)} \end{aligned}$$

A对称正定时，松弛迭代收敛$0 < \omega < 2$. 收敛的快慢与松弛因子ω的选择有密切关系.$0 < \omega < 1$ under-relaxation methods, $\omega > 1$ over-relaxation methods. 如何选取最佳松弛因子,使矩阵谱半径达到最小,是一个尚未很好解决的问题.经验上可取$1.4 < \omega < 1.6$.

在上图所列的算例中,计算的精度要求误差达到小数点后第7位. 从计算的结果可以知道: Gauss-Sediel迭代法计算了34步,而SOR迭代法只用了14步. 请使用相同的参数设定，找出下述例题的SOR迭代解法中最优的ω的数值:

$$\begin{bmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}$$

定理 1.2.3 (SOR法收敛). content

1.2.2 求矩阵特征值的迭代法

计算矩阵一个或多个特征值的数值方法是科学计算中关心的重要问题。

定义 1.2.1 (矩阵特征值). 给定一个代表n维向量空间上的线性变化的$n \times n$矩阵A,希望找某个非零向量x和标量λ,满足

$$A\mathrm{x} = \lambda\mathrm{x}$$

则称λ为A的一个特征值，x为相应的（右）特征向量。

Example

- The linear system $A\mathbf{x} = \mathbf{b}$ given by

$$\begin{aligned} 4x_1 + 3x_2 &= 24 \\ 3x_1 + 4x_2 - x_3 &= 30 \\ -x_2 + 4x_3 &= -24 \end{aligned}$$

  has the solution $(3, 4, -5)^t$.
- Compare the iterations from the Gauss-Seidel method and the SOR method with $\omega = 1.25$ using $\mathbf{x}^{(0)} = (1, 1, 1)^t$ for both methods.

Solution (1/3)

For each $k = 1, 2, \ldots,$ the equations for the Gauss-Seidel method are

$$\begin{aligned} x_1^{(k)} &= -0.75x_2^{(k-1)} + 6 \\ x_2^{(k)} &= -0.75x_1^{(k)} + 0.25x_3^{(k-1)} + 7.5 \\ x_3^{(k)} &= 0.25x_2^{(k)} - 6 \end{aligned}$$

and the equations for the SOR method with $\omega = 1.25$ are

$$\begin{aligned} x_1^{(k)} &= -0.25x_1^{(k-1)} - 0.9375x_2^{(k-1)} + 7.5 \\ x_2^{(k)} &= -0.9375x_1^{(k)} - 0.25x_2^{(k-1)} + 0.3125x_3^{(k-1)} + 9.375 \\ x_3^{(k)} &= 0.3125x_2^{(k)} - 0.25x_3^{(k-1)} - 7.5 \end{aligned}$$

Gauss-Seidel Iterations

k	0	1	2	3	$\cdots$	7
$x_1^{(k)}$	1	5.250000	3.1406250	3.0878906		3.0134110
$x_2^{(k)}$	1	3.812500	3.8828125	3.9267578		3.9888241
$x_3^{(k)}$	1	−5.046875	−5.0292969	−5.0183105		−5.0027940

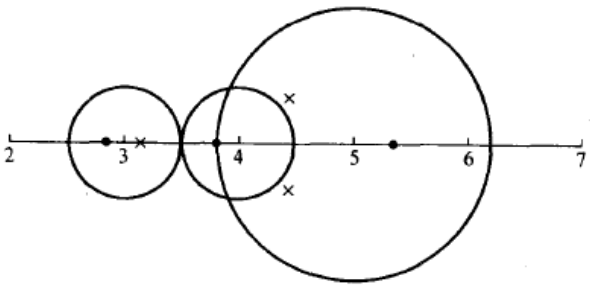
SOR Iterations ($\omega = 1.25$)

k	0	1	2	3	$\cdots$	7
$x_1^{(k)}$	1	6.312500	2.6223145	3.1333027		3.0000498
$x_2^{(k)}$	1	3.5195313	3.9585266	4.0102646		4.0002586
$x_3^{(k)}$	1	−6.6501465	−4.6004238	−5.0966863		−5.0003486

直接估算特征值

谱半径：特征值集合$\lambda(A)$中模最大，记为$\rho(A)$

定理 1.2.4 (Gershgorin圆盘定理). 设A是一个任意的矩阵.则A的特征值λ位于由$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|, \quad (i = 1, \cdots, n)$ 所定义的n个如下图所示的圆盘的并中：



例 1.2.1. 请估算下列矩阵特徵值所在的区间范围。

$$A_1 = \begin{bmatrix} 4.0 & -0.5 & 0.0 \\ 0.6 & 5.0 & -0.6 \\ 0.0 & 0.5 & 3.0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 4.0 & 0.5 & 0.0 \\ 0.6 & 5.0 & 0.6 \\ 0.0 & 0.5 & 3.0 \end{bmatrix}$$

敏感性分析

考虑特征值的扰动问题

$$(A + E)(\mathrm{x} + \delta\mathrm{x}) = (\lambda + \delta\lambda)(\mathrm{x} + \delta\mathrm{x})$$

忽略二阶项，再化简、左乘左特征向量y^H,可得：

$$\delta\lambda \approx \frac{\mathrm{y}^H E \mathrm{x}}{\mathrm{y}^H \mathrm{x}},$$

那么，运用范数不等式可得估计(其中θ是左、右特征向量夹角)

$$|\delta\lambda| \leq \frac{\|\mathrm{y}\|_2 \|\mathrm{x}\|_2}{|\mathrm{y}^H \mathrm{x}|} \|E\|_2 = \frac{1}{\cos\theta} \|E\|_2$$

实对称矩阵或复共轭矩阵的左右特征向量相同，良态！

求解特征值可粗略地分为：直接法和迭代法。

计算特征值的直接法可以从其定义出发，通过其与特征多项式零点的关系，将问题转换为方程求根问题： 特征值问题$A\mathrm{x} = \lambda\mathrm{x}$等价于求解方程

$$(A - \lambda \mathrm{I})\mathrm{x} = 0$$

x有非零解的充要条件是系数矩阵奇异，即特征多项式的零点

$$det(A - \lambda I) = 0$$

就是A的特征值。　　特征值问题↔方程求根问题

例 1.2.2. 算例：特征多项式零点

例 4.3　特征多项式　例 4.2 中的第 3 个矩阵的特征多项式为

$$\det\left(\begin{bmatrix}3 & 1\\ 1 & 3\end{bmatrix}-\lambda\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)=\det\left(\begin{bmatrix}3-\lambda & 1\\ 1 & 3-\lambda\end{bmatrix}\right)$$
$$=(3-\lambda)(3-\lambda)-1\times 1=\lambda^2-6\lambda+8=0,$$

利用求根公式，这个多项式的根为

$$\lambda=\frac{6\pm\sqrt{36-32}}{2}=\frac{6\pm 2}{2},$$

因此，特征值为 $\lambda_1=4,\lambda_2=2$.

迭代法是求解矩阵特征值的重要方法，迭代法通常用于稀疏矩阵，或能方便地执行矩阵向量乘法的隐式算子的情形。注意直接法中也需要用到迭代；如果其中用一个固定的迭代次数收敛几乎不失败，也称为直接法。幂法和反幂法是最简单的求解矩阵特征值的迭代法。

幂法

幂法只能求矩阵A绝对值最大的特征值

```
1 给定x_0;
2 for i = 0,1,··· until convergence do
3 |   计算y_{i+1} = Ax_i;
4 |   计算近似特征向量 x_{i+1} = y_{i+1}/‖y_{i+1}‖_2;
5 |   计算近似特征值 λ_{i+1} = x_{i+1}^T A x_{i+1};
6 end
```

反幂法

相反，反幂法将找到最小特征值，但可以通过“位移”的技巧计算任意数σ附近的那个特征值，其算法如下：

```
1 给定x_0;
2 for i = 0,1,··· until convergence do
3 |   计算y_{i+1} = (A − σI)^{-1} x_i;
4 |   计算近似特征向量: x_{i+1} = y_{i+1}/‖y_{i+1}‖_2;
5 |   计算近似特征值: λ_{i+1} = x_{i+1}^T A x_{i+1};
6 end
```

正交化是一类十分高效的计算技巧，将正交技巧用于矩阵特征值计算的迭代法中也有十分良好的效果。基于QR正交分解，人们提出了如下的QR算法：

```
给定A_0;
for i = 0,1,··· until convergence do
    分解A_i = Q_i R_i (QR分解);
    计算A_{i+1} = R_i Q_i;
end
```

利用该方法，我们将能同时得到矩阵A的所有特征值和特征向量， 最后 $\{A_i\}_{i=1}^{\infty}$ 将收敛到一个上三角矩阵，其对角元素即为所求所有特徵值。这是因为：

$$A_{i+1} = R_i Q_i = (Q_i^T Q_i) R_i Q_i = Q_i^T (Q_i R_i) Q_i = Q_i^T A_i Q_i$$

在这里，A_i等同于用正交迭代隐式计算矩阵$Z_i^T Z_i$，且数值稳定。 此外， 带位移的QR迭代可加快收敛（当选择的位移接近特征值时二次收敛）

瑞利Releigh商迭代

```
给定x_0满足‖x_0‖_2 = 1以及精度TOL;
ρ_0 = (x_0^T A x_0)/(x_0^T x_0);
for i = 0,1,··· until ‖A x_{i+1} − ρ_{i+1} x_{i+1}‖_2 < TOL do
    y_{i+1} = (A − ρ_i I)^{-1} x_i;
    x_{i+1} = y_{i+1}/‖y_{i+1}‖_2;
    ρ_{i+1} = (x_{i+1}^T A x_{i+1})/(x_{i+1}^T x_{i+1});
end
```

A对称情形可用。该算法等同于逆迭代算法中位移取为瑞利商$\rho(\mathrm{x}, A) := \frac{\mathrm{x}^T A \mathrm{x}}{\mathrm{x}^T \mathrm{x}}$。采用初值$\mathrm{x}_0 = [0, \cdots, 0, 1]^T$时与QR迭代得到的序列相同，是分析QR迭代的基础。Releigh商迭代法对单重特征值的计算是局部立方收敛的。

此外，三对角QR迭代是目前求对称三对角矩阵所有特征值最快($O(n)^2$)的方法($n = 25$以内);Matlab指令eig,LAPACK程序ssyev(稠密)和sstev(三对角阵)。 二分法和逆迭代(Bisection and inverse iteration)：二分法只求对称三对角阵特征值的一个子集（给定区间）；逆迭代可求相应的特征向量。最坏的情形（许多特征值很接近）不能保证精度；LAPACK程序ssyevx。分而治之(Divide-and-conquer)：求解$n > 25$对称三对角矩阵所有特征值和特征向量的最快方法，（平均$O(n)^{2.3}$，最坏$O(n)^3$)；LAPACK程序sstevd。 Jacobi方法是求特征值问题最古老的方法(1846年)；可反复利用Givens变换实现；通常比任何方法都慢($O(n^3)$)变换实现；但结果可以更精确。

1.2.3 变分迭代法

基 记秩为N的矩阵,可以由一组线性无关的向量$\{\psi_k\}_{k=1}^N$。

大规模线性方程组的求解是有困难的. 带来困难的原因不仅仅在于其规模之大，另一个重要的原因是线性方程组的不适定性(即矩阵奇异). 利用线性空间的理论，基于变分(即投影)理论的迭代解法是计算近似解的有效方法.

Let A be an $n \times n$ real matrix and $\mathcal{K}$ and $\mathcal{L}$ be two m -dimensional subspaces of R_n. A projection technique onto the subspace $\mathcal{K}$ and orthogonal to $\mathcal{L}$ is a process described as

$$\text{Find } \tilde{x} \in x_0 + \mathcal{K}, \text{ such that } b - A\tilde{x} \perp \mathcal{L}$$

等价于

$$\text{Find } \tilde{x} = x_0 + \delta, \delta \in \mathcal{K}, \text{ such that } (r_0 - A\delta, \omega) = 0, \forall \omega \in \mathcal{L}$$

最速下降法和极小残量法分别是由两类不同的投影方法导致的:

最速下降(Steep Descent)法

```
for j = 0,1,...,until convergence do
    r_j = b − Ax_j;
    α_j = (r_j, r_j)/(Ar_j, r_j);
    x_{j+1} = x_j + α_j r_j;
end
```

极小残量(Minimal Residual)法

```
for j = 0,1,...,until convergence do
    r_j = b − Ax_j;
    α_j = (Ar_j, r_j)/(Ar_j, Ar_j);
    x_{j+1} = x_j + α_j r_j;
end
```

1950年,美国国家标准局数值分析研究所 Hestenes，Stiefel 和 Lanczos，发明了Krylov子空间迭代法求解$Ax = b$。 构造迭代

$$Kx_{i+1} = Kx_i + (b - Ax_i)$$

其中，K(来源于作者俄国人Nikolai Krylov姓氏的首字母)是一个用投影法构造得到的接近于A的矩阵， 根据K所属空间$\mathcal{K}$的不同取法得到求解不同类型的迭代格式。

该迭代形式的算法的妙处在于， 它将复杂问题化简为阶段性的易于计算的子步骤。特别地在第m步，构造子空间

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, \cdots, A^{m-1}r_0\}.$$

只要分别采用$\mathrm{L}_m = \mathrm{K}_m$ 或 $\mathrm{L}_m = A\mathrm{K}_m$ 即可得到不同类型的变分迭代法. 让我们来了解一下如何构造子空间的数值方法.

首先,可以利用Arnoldi算法构造正交子空间K_m,如下述算法:

```
Choose a unit vector v_1;
for j = 1,2,...,m do
    Compute h_ij = (Av_j, v_i),   ∀i = 1,2,⋯,j;
    Compute u_j = Av_j − Σ_{i=1}^{j} h_ij v_i;
    h_{j+1,j} = ‖u_j‖_2;
    if h_{j+1,j} == 0 then
        Stop
    else
        v_{j+1} = u_j / h_{j+1,j}
    end
end
```

接着实施Arnoldi-Modified Gram-Schmidt正交化过程:

```
Choose a unit vector v_1;
for j = 1,2,...,m do
    Compute u_j = Av_j;
    for i = 1,...,j do
        h_{i,j} = (u_j, v_i);
        u_j = u_j − h_{i,j} v_i;
    end
    h_{j+1,j} = ‖u_j‖_2;
    if h_{j+1,j} == 0 then
        Stop
    else
        v_{j+1} = u_j / h_{j+1,j}
    end
end
```

Full Orthogonalization Method(FOM)结合了上述两个过程:

```
1 Compute r_0 = b − Ax_0, β = ‖r_0‖_2, and v_1 = r_0/β;
2 Define the m × m matrix H_m = {h_{i,j}}_{i,j=1,2,...,m}, set H_m = 0;
3 for j = 1, 2, ..., m do
4 |   Compute u_j = Av_j;
5 |   for i = 1, ..., j do
6 |   |   h_{i,j} = (u_j, v_i);
7 |   |   u_j = u_j − h_{i,j} v_i;
8 |   end
9 |   h_{j+1,j} = ‖u_j‖_2;
10|   if h_{j+1,j} == 0 then
11|   |   set m = j, and goto 16
12|   else
13|   |   Comput v_{j+1} = u_j/h_{j+1,j}
14|   end
15 end
16 Compute y_m = H_m^{-1}(βe_1) and x_m = x_0 + V_m y_m;
```

极小残差(Generalized Minimized Residual)法

简称GMRes. 通过上述算法的铺垫,可以归结为如下算法

```
1 Compute r_0 = b − Ax_0, β = ‖r_0‖_2, and v_1 = r_0/β;
2 Define the (m + 1) × m matrix H_m = {h_{i,j}}_{1≤i≤(m+1)}^{1≤j≤m}, set H_m = 0;
3 for j = 1, 2, ..., m do
4 |   Compute u_j = Av_j;
5 |   for i = 1, ..., j do
6 |   |   h_{i,j} = (u_j, v_i);
7 |   |   u_j = u_j − h_{i,j} v_i;
8 |   end
9 |   h_{j+1,j} = ‖u_j‖_2;
10|   if h_{j+1,j} == 0 then
11|   |   set m = j, and goto 16
12|   else
13|   |   Comput v_{j+1} = u_j/h_{j+1,j}
14|   end
15 end
16 Compute y_m to minimize ‖βe_1 − H_m y‖_2 and x_m = x_0 + V_m y_m;
```

共轭梯度法(Conjugate Gradient)及其变形

GMRes迭代算法对于非对称矩阵A尤其适用. 在A对称时的特殊情形,利用Lanczos三项递推关系可将FOM简化成如下简洁的算法:

```
Compute r_0 = b − Ax_0, p_0 = r_0;
for j = 0,1,...,until convergence do
    α_j = (r_j, r_j)/(Ap_j, p_j);
    x_{j+1} = x_j + α_j p_j;
    r_{j+1} = r_j − α_j A p_j;
    β_j = (r_{j+1}, r_{j+1})/(r_j, r_j);
    p_{j+1} = r_{j+1} + β_j p_j;
end
```

同理得到的迭代方法即为共轭梯度法(CG), 该迭代法对于对称矩阵A具有收敛性. 令 x_m 是执行第$m-$步Conjugate Gradient algorithm得到的近似解, 并且x^*是精确解，那么

$$||\mathrm{x}^* - \mathrm{x}_m||_A \leq \left[\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right]^m ||\mathrm{x}^* - \mathrm{x}_0||_A$$

其中κ是矩阵A最大与最小特征值的比值，即所谓条件数。可见κ越接近1，则共轭梯度法收敛越快！

CG算法的其他主要常见变形有

- ICCG：Incomplete Cholesky预处理的CG迭代
- BiCG：双正交共轭梯度法
- BiCGstab：稳定化的BiCG

1.2.4 预条件处理技术

预处理技术可以减少迭代法运行时的迭代次数(注意不一定能减少计算量). 预处理的优点在于能结合不同的迭代法优势,在很多情形可以减少总体的计算量. 以CG迭代法为例，该方法的主要思想是: 设M是non-singular矩阵，并且$M^{-1}A$的条件数相对教小，则求解

$$(M^{-1}A)\mathrm{x} = M^{-1}\mathrm{b}$$

相对容易， 或者

$$(AM^{-1})\mathrm{y} = \mathrm{b},$$

再求$M\mathrm{x} = \mathrm{y}$得到原方程的解。

显然地,$M\mathrm{x} = \mathrm{y}$相比于原问题更容易求解. 另一方面,CG算法采用的M也需要是对称正定的，这样不会破坏CG算法的收敛性. 举例来说，假设$A = L + D + L^T$,那么可以构造Jacobi预处理子

$$M := M_{Jacobi} = D^{-1}$$

或SOR预处理子

$$M := M_{SOR} = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega L^T), 0 < \omega < 2.$$

1.2.5 教材

关于特征值问题及其求解方法有不少专著

- Yousef Saad: Iterative methods for sparse systems(2nd edition)
- Yousef Saad: Numerical Methods for Large Eigenvalue Problems(2nd Edition)
- James Demmel, et. al. : Templates for the solution of algebraic eigenvalue problems
- R. S. Varga: Matrix Iterative Analysis(2nd Edition)
- H. Wilkinson: The Algebraic Eigenvalue Problem(Wilkinson Prize for Numerical Software)

1.2.6 软件包

petsc

偏微分方程数值解研究常用软件包,支持大规模并行计算.

slepc

特征值问题求解工具包

hypre

支持多重网格快速算法开发和大规模并行计算.

trillinos

偏微分方程数值解研究常用软件包,支持大规模并行计算.

1.2.7 Exercises

1. exe1
2. exe1
3. exe1

4. exe1

5. exe1

6. exe1

7. exe1

8. exe1

9. exe1

10. exe1

11. exe1

12. exe1

13. exe1

14. exe1

15. exe1 5