现代数值计算方法

第八章 矩阵特征值问题的计算

第八章 矩阵特征值问题的计算

许多工程实际问题的求解, 如振动问题、稳定性问题等, 最终都归结为求某些矩阵的特征值和特征向量的问题. 大家知道, n 阶方阵 A 的特征值与特征向量, 是满足如下两个方程的数 λ 和非零向量 ξ:

$$|\lambda I - A| = 0, \tag{8.1}$$

$$A\xi = \lambda\xi \quad \text{或} \quad (\lambda I - A)\xi = 0. \tag{8.2}$$

方程 (8.1) 称为矩阵 A 的特征方程, 它是 λ 的 n 次代数方程, 当 n 较大时难以准确求解. 因此, 从数值计算的观点来看, 用特征多项式来求矩阵特征值的方法并不可取.

在实际应用中, 求矩阵的特征值和特征向量通常采用迭代法. 其




基本思想是, 将特征值和特征向量作为一个无限序列的极限来求得. 舍入误差对这类方法的影响很小, 但通常计算量较大.

根据具体问题的需要, 有些实际问题只要计算绝对值最大的特征值, 也有些实际问题则只需计算绝对值最小的特征值, 当然, 更多的问题则要求计算全部特征值和特征向量. 本章介绍几种目前在计算机上比较常用的矩阵特征值问题的数值方法.

§8.1 幂法和反幂法

§8.1.1 幂法及其通用程序

幂法是通过求矩阵的特征向量来求出特征值的一种迭代法. 它主要用来求按模最大的特征值和相应的特征向量的. 其优点是算法简单, 容易计算机实现, 缺点是收敛速度慢, 其有效性依赖于矩阵特征值的分布情况.

适于使用幂法的常见情形是: A 的特征值可按模的大小排列为 $|\lambda_1| > |\lambda_2| \geq \lambda_3| \geq \cdots \geq |\lambda_n|$, 且其对应特征向量 $\xi_1,\ \xi_2, \cdots, \xi_n$ 线性无关. 此时, 任意非零向量 x_0 均可用 $\xi_1,\ \xi_2, \cdots, \xi_n$ 线性表示, 即

$$x_0 = \beta_1\xi_1 + \beta_2\xi_2 + \cdots + \beta_n\xi_n, \tag{8.3}$$

且 $\beta_1,\ \beta_2, \cdots, \beta_n$ 不全为零. 作向量序列 $x_k = A^k x_0$, 则

$$\begin{aligned} x_k &= A^k x_0 = \beta_1 A^k\xi_1 + \beta_2 A^k\xi_2 + \cdots + \beta_n A^k\xi_n \\ &= \beta_1\lambda_1^k\xi_1 + \beta_2\lambda_2^k\xi_2 + \cdots + \beta_n\lambda_n^k\xi_n \\ &= \lambda_1^k\left[\beta_1\xi_1 + \beta_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\xi_2 + \cdots + \beta_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\xi_n\right]. \end{aligned}$$

由此可见, 若 $\beta_1 \neq 0$, 则因 $k \to \infty$ 时,

$$\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0\ (i = 2, \cdots, n),$$

故当 k 充分大时, 必有

$$x_k \approx \lambda_1^k \beta_1 \xi_1,$$

即 x_k 可以近似看成 λ_1 对应的特征向量; 而 x_k 与 x_{k-1} 分量之比

$$\frac{(x_k)_i}{(x_{k-1})_i} \approx \frac{\lambda_1^k \beta_1 (\xi_1)_i}{\lambda_1^{k-1} \beta_1 (\xi_1)_i} = \lambda_1.$$

于是利用向量序列 $\{x_k\}$ 既可求出按模最大的特征值 λ_1, 又可求出对应的特征向量 ξ_1.

在实际计算中, 考虑到当 $|\lambda_1| > 1$ 时, $\lambda_1^k \to \infty$; 当 $|\lambda_1| < 1$ 时, $\lambda_1^k \to 0$, 因而计算 x_k 时可能会导致计算机 "上溢" 或 "下溢" 现象发生, 故采取每步将 x_k 归一化处理的办法, 即将 x_k 的各分量都除以绝对值最大的分量, 使 $\|x_k\|_\infty = 1$. 于是, 求 A 按模最大特征值 λ_1 和对应的特征向量 ξ_1 的算法, 可归纳为如下步骤:

算法 8.1 (幂法)

步 1　输入矩阵 A, 初始向量 y_0, 误差限 ε, 最大迭代次数 N;

步 2　置 $k := 1,\ \mu := 0,\ x_0 = y_0/\|y_0\|_\infty$;

步 3　计算 $y_k = Ax_{k-1}$;

步 4　计算

$$|[y_k]_r| = \max_{1\le i\le n} |[y_k]_i|,$$

其中 $[y_k]_i$ 表示向量 y_k 的第 i 个分量, 并置

$$m_k := [y_k]_r, \quad x_k := y_k/m_k;$$

步 5　若 $|m_k - \mu| < \varepsilon$, 停算, 输出 $m_k,\ x_k$; 否则, 转步 6;

步 6　若 $k < N$, 置 $k := k + 1,\ \mu := m_k$, 转步 3; 否则输出计算失败信息, 停算.

上述算法称为幂法. 可以证明下面的收敛性定理:

定理 8.1 设矩阵 A 的特征值可按模的大小排列为 $|\lambda_1| > |\lambda_2| \geq \lambda_3| \geq \cdots \geq |\lambda_n|$, 且其对应特征向量 $\xi_1,\ \xi_2, \cdots, \xi_n$ 线性无关. 序列 $\{x_k\}$ 由算法 8.1 产生, 则有

$$\lim_{k\to\infty} x_k = \xi_1^0 = \frac{\xi_1}{\max(\xi_1)}, \quad \lim_{k\to\infty} m_k = \lambda_1, \tag{8.4}$$

其中 ξ_1^0 表示将 ξ_1 单位化后得到的向量, $\max(\xi_1)$ 表示向量 ξ_1 绝对值最大的分量.

证 由算法 8.1 步 3 和步 4 知

$$x_k = \frac{y_k}{m_k} = \frac{Ax_{k-1}}{m_k} = \frac{A^2x_{k-2}}{m_km_{k-1}} = \cdots = \frac{A^kx_0}{m_km_{k-1}\cdots m_1}.$$

由于 x_k 的最大分量为 1, 即 $\max(x_k) = 1$, 故

$$m_km_{k-1}\cdots m_1 = \max(A^kx_0).$$

从而

$$x_k = \frac{A^k x_0}{\max(A^k x_0)} = \frac{\lambda_1^k \left[\beta_1 \xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \xi_i\right]}{\max\left\{\lambda_1^k \left[\beta_1 \xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \xi_i\right]\right\}}$$

$$= \frac{\beta_1 \xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \xi_i}{\max\left\{\beta_1 \xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \xi_i\right\}}$$

可见

$$\lim_{k\to\infty} x_k = \frac{\beta_1 \xi_1}{\max(\beta_1 \xi_1)} = \frac{\xi_1}{\max(\xi_1)} = \xi_1^0.$$

又

$$
\begin{aligned}
y_k &= Ax_{k-1} = \frac{A^k x_0}{\max(A^{k-1}x_0)} \\
&= \frac{\lambda_1^k \left[\beta_1\xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \xi_i\right]}{\lambda_1^{k-1} \max\left[\beta_1\xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^{k-1} \xi_i\right]},
\end{aligned}
$$

即有

$$
m_k = \max(y_k) = \lambda_1 \frac{\max\left[\beta_1\xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \xi_i\right]}{\max\left[\beta_1\xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^{k-1} \xi_i\right]},
$$

从而

$$
\lim_{k\to\infty} m_k = \lambda_1
$$

成立. □

下面给出幂法的 MATLAB 通用程序.

- 幂法 MATLAB 程序

```
%mapower.m
function [lambda,v,k]=mapower(A,x0,epsilon,max1)
%用途: 用幂法求矩阵的模最大特征值和对应的特征向量
%格式: [lambda,v,k]=mapower(A,x0,epsilon,max1)
%说明: A为n阶方阵,x0为初始向量,epsilon为上限,max1为循
%      环次数,lambda返回按模最大的特征值,v返回对应的
%      特征向量,k返回迭代次数
lambda=0; k=0; err=1;
while((k<max1)&(err>epsilon))
   y=A*x;   [m,j]=max(abs(y));
   m=y(j); x=y/m;
```

```
    err=abs(lambda-m);
    lambda=m;  k=k+1;
end
v=x;
```

例 8.1 利用幂法通用程序 mapower.m, 求 A 按模最大的特征值 λ_1 和对应的特征向量 ξ_1, 其中

$$A = \begin{pmatrix} -1 & 2 & 1 \\ 2 & -4 & 1 \\ 1 & 1 & -6 \end{pmatrix}.$$

解 在 MATLAB 命令窗口执行:

```
>> A=[-1,2,1;2,-4,1;1,1,-6];
>> x=[1;1;1];
```

Back
Close

```
>> [lambda,v,k]=mapower(A,x,1e-10,100)
lambda =
    -6.42106661402299
v =
    -0.04614548326375
    -0.37492113082845
     1.00000000000000
k =
    78
```

§8.1.2 幂法的加速技术

可以证明在定理 8.1 的条件下, 算法 8.1 是线性收敛的. 事实上,

设 k 充分大时, $A^k x_0$ 的最大分量是它的第 j 个分量, 则

$$\begin{aligned} m_k - \lambda_1 &= \max(y_k) - \lambda_1 = \frac{\max(A^k x_0)}{\max(A^{k-1} x_0)} - \lambda_1 \\ &= \frac{\left[\beta_1 \lambda_1^k \xi_1 + \beta_2 \lambda_2^k \xi_2 + \cdots + \beta_2 \lambda_n^k \xi_n\right]_j}{\left[\beta_1 \lambda_1^{k-1} \xi_1 + \beta_2 \lambda_2^{k-1} \xi_2 + \cdots + \beta_n \lambda_n^{k-1} \xi_n\right]_j} - \lambda_1 \\ &= \frac{\left[\beta_2 \lambda_2^{k-1} (\lambda_2 - \lambda_1) \xi_2 + \cdots + \beta_n \lambda_n^{k-1} (\lambda_n - \lambda_1) \xi_n\right]_j}{\left[\beta_1 \lambda_1^{k-1} \xi_1 + \beta_2 \lambda_2^{k-1} \xi_2 + \cdots + \beta_n \lambda_n^{k-1} \xi_n\right]_j} \end{aligned}$$

于是有

$$\begin{aligned} m_k - \lambda_1 &= \left(\frac{\lambda_2}{\lambda_1}\right)^{k-1} \frac{\left[\beta_2 (\lambda_2 - \lambda_1) \xi_2 + \sum_{i=3}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_2}\right)^{k-1} (\lambda_i - \lambda_1) \xi_i\right]_j}{\left[\beta_1 \xi_1 + \sum_{i=2}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^{k-1} \xi_i\right]_j} \\ &= \left(\frac{\lambda_2}{\lambda_1}\right)^{k-1} M_k, \quad M_k \to M, \end{aligned}$$

其中 M 为常数. 所以, 当 $k \to \infty$时,

$$\frac{|m_{k+1} - \lambda_1|}{|m_k - \lambda_1|} = \left|\frac{M_{k+1}(\lambda_2/\lambda_1)^k}{M_k(\lambda_2/\lambda_1)^{k-1}}\right| \to \left|\frac{\lambda_2}{\lambda_1}\right|,$$

即幂法的收敛速度与比值 $|\lambda_2/\lambda_1|$ 的大小有关, $|\lambda_2/\lambda_1|$ 越小, 收敛速度越快, 当此比值接近于 1 时, 收敛速度是非常缓慢的. 这一事实启发我们, 可以对矩阵作一原点位移, 令

$$B = A - \alpha I,$$

其中, α 为参数, 选择此参数可使矩阵 B 的上述比值更小, 以加快幂法的收敛速度. 设矩阵 A 的特征值为 $\lambda_1, \lambda_2, \cdots, \lambda_n$, 对应的特征向量为 $\xi_1, \xi_2, \cdots, \xi_n$, 则矩阵 B 的特征值为 $\lambda_1 - \alpha, \lambda_2 - \alpha, \cdots, \lambda_n - \alpha$, B 的特征向量和 A 的特征向量相同. 假设原点位移后, B 的特征值 $\lambda_1 - \alpha$

仍为绝对值最大的特征值, 选择 α 的目的是使

$$\max_{2\le i\le n}\frac{|\lambda_i-\alpha|}{|\lambda_1-\alpha|}<\left|\frac{\lambda_2}{\lambda_1}\right|. \tag{8.5}$$

适当地选择 α 可使幂法的收敛速度得到加速. 此时 $m_k\to\lambda_1-\alpha$, $m_k+\alpha\to\lambda_1$, 而 x_k 仍然收敛于 A 的特征向量 ξ_1^0. 这种加速收敛的方法称为原点位移法.

在实际计算中, 由于事先矩阵的特征值分布情况一般是不知道的, 参数 α 的选取存在困难, 故原点位移法是很难实现的. 但是我们将会看到, 在反幂法中原点位移参数 α 是非常容易选取的, 因此, 带原点位移的反幂法已成为改进特征值和特征向量精度的标准算法. 我们给出采用原点加速技术的幂法 MATLAB 通用程序如下:

- 原点位移幂法 MATLAB 程序

```
%mapowerp.m
```

```
function [lambda,v,k]=mapowerp(A,x,alpha,epsilon,max1)
%用途: 用原点位移幂法求矩阵的模最大特征值及特征向量
%格式: [lambda,v,k]=mapowerp(A,x,alpha, epsilon,max1)
%说明: A为n阶方阵,x为初始向量,epsilon为上限,max1为循
%      环次数,alpha为原点位移参数.lambda返回按模最大
%      的特征值,v返回对应的特征向量,k返回迭代次数
lambda=0; k=0; err=1;
n=length(x); A=A-alpha*eye(n);
while((k<max1)&(err>epsilon))
    y=A*x; [m,j]=max(abs(y));
    m=y(j); x=y/m;
    err=abs(lambda-m);
    lambda=m; k=k+1;
```

```
end
lambda=lambda-alpha;
v=x;
```

例 8.2 利用原点位移幂法通用程序 mapowerp.m, 求 A 按模最大的特征值 λ_1 和对应的特征向量 ξ_1, 其中

$$A = \begin{pmatrix} -1 & 2 & 1 \\ 2 & -4 & 1 \\ 1 & 1 & -6 \end{pmatrix}.$$

解 选取 $\alpha = -2$, 在 MATLAB 命令窗口执行:

```
>> A=[-1,2,1;2,-4,1;1,1,-6];
>> x=[1;1;1];
>> [lambda,v,k]=mapowerp(A,x,-2,1e-10,100)
```

Back
Close

```
lambda =
        -6.42106661417413
v =
   -0.04614548312203
   -0.37492113109948
    1.00000000000000
k =
   52
```

由上述结果可以看出, 在同样的精度控制下, 带原点位移加速的幂法只需要迭代 52 次, 而纯粹的幂法则需要迭代 78 次 (例 8.1).

§8.1.3 反幂法及其通用程序

设 A 可逆, 则对 A 的逆阵 A^{-1} 施以幂法称为反幂法. 由于 $A\xi_i =$

$\lambda_i\xi_i$ 时, 成立 $A^{-1}\xi_i = \lambda_i^{-1}\xi_i$. 因此, 若 $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_{n-1}| > |\lambda_n|$, 则 λ_n^{-1} 是 A^{-1} 按模最大的特征值, 此时按反幂法, 必有

$$m_k \to \lambda_n^{-1}, \quad x_k \to \xi_n^0,$$

且其收敛率为 $|\lambda_n/\lambda_{n-1}|$. 任取初始向量 x_0, 构造向量序列

$$x_{k+1} = A^{-1}x_k, \quad k = 0, 1, 2, \cdots \tag{8.6}$$

按幂法计算即可. 但用 (8.6) 计算, 首先要求 A^{-1}, 这比较麻烦而且是不经济的. 实际计算中, 通常用解方程组的办法, 即用

$$Ax_{k+1} = x_k, \quad k = 0, 1, 2, \cdots \tag{8.7}$$

求 x_{k+1}. 为防止计算机溢出, 实际计算时所用的公式为

$$\begin{cases} y_k = x_k/\|x_k\|_\infty, \\ Ax_{k+1} = y_k, \end{cases} \quad k = 0, 1, 2, \cdots \tag{8.8}$$

反幂法主要用于已知矩阵的近似特征值为 α 时, 求矩阵的特征向量并提高特征值的精度. 此时, 可以用原点位移法来加速迭代过程, 于是 (8.8) 相应为

$$\begin{cases} y_k = x_k/\|x_k\|_\infty, \\ (A-\alpha I)x_{k+1} = y_k, \end{cases} \qquad k = 0, 1, 2, \cdots \tag{8.9}$$

为节省计算量, 通常先用列主元 LU 分解将矩阵 $A-\alpha I$ 分解为下三角矩阵 L 和上三角矩阵 U, 这样在迭代过程中每一步就只要解两个三角方程组了.

反幂法的计算步骤如下:

算法 8.2 (反幂法)

步 1 输入矩阵 A, 初始向量 x_0, 近似值 α, 误差限 ε, 最大迭代次数 N;

步 2 置 $k := 1,\ \mu := 1;$

步 3 作列主元 LU 分解 $P(A - \alpha I) = LU;$

步 4 计算 $\left|[x_k]_r\right| = \max\limits_{1\le i\le n}\left|[x_k]_i\right|$, 并置 $m := [x_k]_r;$

步 5 计算 x 的新值: $y = x/m,\quad Lz = Py,\quad Ux = z;$

步 6 若 $\left|\dfrac{1}{m} - \dfrac{1}{\mu}\right| < \varepsilon$, 则置 $\lambda := \alpha + \dfrac{1}{m}$, 输出 $\lambda,\ x,$ 停算. 否则, 转步 7;

步 7 若 $k < N$, 置 $k := k + 1,\ \mu := m,$ 转步 4; 否则输出计算失败信息, 停算.

注 **8.1** (1) 在上述算法中, 如果 $\alpha = 0,$ 则求出 A 的按模最小的特征值.

(2) 通常首先用幂法求出 A 的按模最大的近似特征值作为算法 8.2 中的 α 值, 再使用该算法对 α 和相应的特征向量进行精确化.

我们给出反幂法的 MATLAB 通用程序如下:

```
%mainvp.m
function [lambda,v,k]=mainvp(A,x,alpha,epsilon,max1)
%用途: 用反幂法求矩阵的模最大特征值及特征向量
%格式: [lambda,v,k]=mainvp(A,x,alpha,epsilon,max1)
%说明: A为n阶方阵,x为初始向量,epsilon为上限,max1为循环
%      次数,alpha为模最大的近似特征值.lambda返回按模最
%      大的特征值,v返回对应的特征向量,k返回迭代次数
lambda=0; k=0; err=1;
mu=0.5; n=length(x);
A=A-alpha*eye(n);
[L,U,P]=lu(A);
while(k<max1)&(err>epsilon)
```

```
    [m,j]=max(abs(x));
    m=x(j);     y=x./m;
    z=L\(P*y);   x=U\z;
    err=abs(1/m-1/mu);
    k=k+1;   mu=m;
end
lambda=alpha+1/m;
v=y;
```

例 8.3 利用反幂法通用程序 mainvp.m, 求 A 近似于 -6.42 的特征值和对应的特征向量, 其中

$$A = \begin{pmatrix} -1 & 2 & 1 \\ 2 & -4 & 1 \\ 1 & 1 & -6 \end{pmatrix}.$$

解 注意到此处 $\alpha = -6.42$, 在 MATLAB 命令窗口执行:

```
>> A=[-1,2,1;2,-4,1;1,1,-6];
>> x=[1;1;1]; alpha=-6.42;
>> [lambda,v]=mainvp(A,x,alpha,1e-10,100)
lambda =
        -6.42106661430895
v =
   -0.04614548302620
   -0.37492113128274
    1.00000000000000
k =
    6
```

作业：P194: 8.1; P195: 8.3.