数值分析第一次大作业

张晋

学号：15091060

最后更新于：May 27, 2017

目录 | 0

题目 1

设有 501×501 的实对称矩阵 $\boldsymbol{A}$,

$$\boldsymbol{A}=\begin{bmatrix} a_1 & b & c & & \\ b & \ddots & \ddots & \ddots & \\ c & \ddots & \ddots & \ddots & c \\ & \ddots & \ddots & \ddots & b \\ & & c & b & a_{501} \end{bmatrix}$$

其中:

$a_i=(1.64-0.024i)sin(0.2i)-0.64e^{\frac{0.1}{i}}(i=1,2,\cdots,501)$
$b=0.16$
$c=-0.064$
矩阵 $\boldsymbol{A}$ 的特征值为 $\lambda_i(i=1,2,\cdots,501)$ 并且有

$$\lambda_1\leq\lambda_2\leq\lambda_3\leq\cdots\leq\lambda_{501},|\lambda_s|=\min_{1\leq i\leq501}|\lambda_i|$$

1. 求 λ_1,λ_{501} 和 λ_s 的值。
2. 求 $\boldsymbol{A}$ 的与数 $\mu_k=\lambda_1+k\dfrac{\lambda_{501}-\lambda_1}{40}$ 最接近的特征值 $\lambda_{i_k}(k=1,2,\cdots,39)$。
3. 求 $\boldsymbol{A}$ 的 (谱范数) 条件数 $cond(\boldsymbol{A})_2$ 和行列式 $det\boldsymbol{A}$

算法设计方案 2

2.1 基本算法

2.1.1 幂法

幂法是一种求实矩阵 A 的按模最大的特征值 λ_1 及其对应的特征向量 $\boldsymbol{x}_1$ 的方法。特别适合于大型稀疏矩阵。

设 $\boldsymbol{A}=(a_{ij})_{n\times n}\in\mathbb{R}^{n\times n}$ 有一个完全特征向量组, 其特征值为 $\lambda_1,\lambda_2,\cdots,\lambda_n$，对应的特征向量为 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n$。

并设 A 的主特征值是实根，且满足

$$|\lambda_1|>|\lambda_2|\geq\cdots\geq|\lambda_n|$$

现在讨论求 λ_1 及 $\boldsymbol{x}_1$ 的基本方法.

$$\begin{aligned}
&\forall\boldsymbol{v}_0=a_1\boldsymbol{x}_1+a_2\boldsymbol{x}_2+\cdots+a_n\boldsymbol{x}_n,(a_1\neq 0)\\
&\ \boldsymbol{v}_1=\boldsymbol{A}\boldsymbol{v}_0=a_1\lambda_1\boldsymbol{x}_1+a_2\lambda_2\boldsymbol{x}_2+\cdots+a_n\lambda_n\boldsymbol{x}_n\\
&\ \cdots\\
&\ \boldsymbol{v}_k=\boldsymbol{A}\boldsymbol{v}_{k-1}=\lambda_1^k\left[a_1\boldsymbol{x}_1+a_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\boldsymbol{x}_2+\cdots+a_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\boldsymbol{x}_n\right]
\end{aligned}$$

当 k 很大时, 有

$$\boldsymbol{v}_k\approx\lambda_1^k a_1\boldsymbol{x}_1$$

$$\boldsymbol{v}_{k+1}\approx\lambda_1\boldsymbol{v}_k$$

$$\boldsymbol{A}\boldsymbol{v}_k \approx \lambda_1 \boldsymbol{v}_k$$

$$\lim_{k\to\infty} \frac{\boldsymbol{v}_k}{\lambda_1^k} = a_1 \boldsymbol{x}_1$$

即 $\boldsymbol{v}_k$ 是 λ_1 的近似的特征向量. 而主特征值

$$\lambda_1 \approx \frac{(\boldsymbol{v}_{k+1})_j}{(\boldsymbol{v}_k)_j}$$

那么若我们要求某个 n 阶方阵 A 的特征值和特征向量, 可以先任取一个初始向量 $\boldsymbol{x}_0$, 构造序列 $\{\boldsymbol{x}_{k=1}^{\infty}\}$, 其中 $\boldsymbol{x}_k := \boldsymbol{A}\boldsymbol{x}_{k-1}$ 那么当 k 充分大时，$\boldsymbol{x}_k$ 可近似作为矩阵 $\boldsymbol{A}$ 的特征值。

实际运算时为避免 $\boldsymbol{x}_k$ 的模过大或过小，通常每迭代一次都对 $\boldsymbol{x}_k$ 进行归一化。

算法过程如下：

(1) 任取一个长度为 n 的非零列向量 $\boldsymbol{x}_i$;

(2) 用矩阵 $\boldsymbol{A}$ 乘以 $\boldsymbol{x}_i$，得到列向量 $\boldsymbol{x}_{i+1} = \boldsymbol{A}\boldsymbol{x}_i$;

(3) 标准化列向量 $\boldsymbol{x}_{i+1}$，也就是表示成一个因子 (绝对值最大的元素的倒数) 乘以 $\boldsymbol{x}_{i+1}$，使得 $\boldsymbol{x}_{i+1}$ 中最大的元素是 1;

(4) 将 $\boldsymbol{x}_{i+1}$(不包括上面的因子) 作为 (1) 中的 $\boldsymbol{x}_i$ 重复 (1) 中的操作。这个迭代过程一直进行，直到相邻两个列向量 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_{i+1}$ 的差异满足一定的条件，这个差异可以用多种方式进行衡量，如果使用无限范数来衡量即是

$$\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|_\infty \le tol$$

大的特征值就是当前 $\boldsymbol{x_{i+1}}$ 进行标准化得到的因子。幂法肯定是会收敛的, 如果初始列向量和特征向量很接近的话收敛会很快，否则会很慢。幂法只能算出最大的特征值，并且最大的特征值不能是特征方程的重根，其次，最大特征值必须是实数。

程序伪代码如下：

Algorithm 1 The Power Method

1: Choose a random vector $\boldsymbol{u}_0 \in \mathbb{R}^n$
2: **while** $|\beta_k - \beta_{k-1}| > \epsilon$ **do**
3: $\quad \boldsymbol{y}_{k-1} = \boldsymbol{u}_{k-1}/\|\boldsymbol{u}_{k-1}\|$
4: $\quad \boldsymbol{u}_k = \boldsymbol{A}\boldsymbol{y}_{k-1}$
5: $\quad \beta_k = \boldsymbol{y}_{k-1}^T \boldsymbol{u}_k$
6: **end while**

2.1.2 反幂法

反幂法可求非奇异实矩阵的按模最小特征值及特征向量由于矩阵 A 的特征值 λ 的倒数就是矩阵 A^{-1} 的特征值，所以，如果对 A^{-1} 进行同样的幂法过程，那么就可以得到矩阵 A 的最小的特征值, 这种方法就叫做反幂法。

程序伪代码如下：

Algorithm 2 The Inverse Power Method

1: Choose a random vector $\boldsymbol{u}_0 \in \mathbb{R}^n$
2: **while** $|\beta_k - \beta_{k-1}| > \epsilon$ **do**
3: $\quad \boldsymbol{y}_{k-1} = \boldsymbol{u}_{k-1}/\|\boldsymbol{u}_{k-1}\|$
4: $\quad \boldsymbol{u}_k = \boldsymbol{A}^-1\boldsymbol{y}_{k-1}$
5: $\quad \beta_k = \boldsymbol{y}_{k-1}^T\boldsymbol{u}_k$
6: **end while**

PS: 其中 $\boldsymbol{u}_k$ 通常由 *Doolittle* 分解法求得

2.1.3 移位幂法

如果得到了矩阵 A 的最大或者最小的特征值，那么使用移位幂法可以得到其他的特征值。它的原理是：假设 $ax = \lambda_1 x$，其中 λ_1 是通过幂法求得的矩阵的最大的特征值，那么新的移位矩阵 $[a-\lambda_1 * I]$ 的特征值便是 $\lambda_2-\lambda_1, \lambda_3-\lambda_1, \cdots, \lambda_n-\lambda_1$。移位矩阵的特征向量和原矩阵的特征向量是一样的。如果对移位矩阵也应用基本幂法，得到特征值 α_k，那么原矩阵的特征值 $\lambda_k = \alpha_k + \lambda_1$。不断重复上面的过程 $k-2$ 次，便可以得到原矩阵的所有的特征值

2.1.4 Doolittle 分解法

定义: 杜尔里特算法 (Doolittle algorith) 从下至上地对矩阵 A 做初等行变换，将对角线左下方的元素变成零，然后再证明这些行变换的效果等同于左乘一系列单位下三角矩阵，这一系列单位下三角矩阵的乘积的逆就是 L 矩阵，它也是一个单位下三角矩阵。本质上是高斯消元法的一种表达形式。

复杂度: 时间复杂度一般在 $\frac{2n^3}{3}$ 左右。空间复杂度为 n^2，只占用一个储存矩阵 $\boldsymbol{A}$ 的数组。

具体步骤如下：

对给定的 $N \times N$ 矩阵 $A = (a_{n,n})$

有 $A^{(0)} := A$

然后定义对于 $n = 1, \cdots, N-1$ 的情况如下：

在第 n 步，消去矩阵 $A^{(n-1)}$ 的第 n 列主对角线下的元素：

将 $A^{(n-1)}$ 的第 n 行乘以

$$l_{i,n} := -\frac{a_{i,n}^{(n-1)}}{a_{n,n}^{(n-1)}}$$

之后加到第 i 行上去。其中 $i = n+1, \cdots, N$

这相当于在 $\boldsymbol{A}^{(n-1)}$ 的左边乘上一个单位下三角矩阵：

$$L_n = \begin{pmatrix} 1 & & & & & 0 \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{n+1,n} & \ddots & & \\ & & \vdots & & \ddots & \\ 0 & & l_{N,n} & & & 1 \end{pmatrix}$$

于是设:

$$A^{(n)} := L_n A^{(n-1)}$$

经过 $N-1$ 轮操作后，所有在主对角线下的系数都为 0 了，于是我们得到了一个上三角矩阵：$A^{(n-1)}$，这时就有：

$$\begin{aligned} A &= L_1^{-1} L_1 A^{(0)} \\ &= L_1^{-1} A^{(1)} \\ &= L_1^{-1} L_2^{-1} L_2 A^{(1)} \\ &= L_1^{-1} L_2^{-1} A^{(2)} \\ &= \dots \\ &= L_1^{-1} \dots L_{N-1}^{-1} A^{(N-1)} \end{aligned}$$

这时，矩阵 $A^{(n-1)}$ 就是 U，

$$L = L_1^{-1} \dots L_{N-1}^{-1}$$

下三角矩阵 L_k 的逆依然是下三角矩阵，而且下三角矩阵的乘积仍是下三角矩阵，所以 L 是下三角矩阵。于是我们得到分解：$A = LU$。

Doolittle 分解伪代码如下：

Algorithm 3 Doolittle Algorithm

for $k = 1;\ k \le n;\ k++$ **do**
  for $j = k;\ j < n;\ j++$ **do**
    $a_{kj} = a_{kj} - \sum_{t=1}^{k-1} a_{kt} a_{tj}$
  end for
  for $i = k+1;\ i \le n;\ i++$ **do**
    $a_{ik} = (a_{ik} - \sum_{t=1}^{k-1} a_{it} a_{tk}) / a_{kk}$
  end for
end for

2.2 问题求解

2.2.1 第一问

1. 求 λ_1, λ_{501} 和 λ_s 的值。

   (a) 通过幂法求得按模最大的特征值 λ_{m1}

   (b) 求出 $\lambda_{m1} = -10.7001$，可以判断 λ_{m1} 即为 λ_1

   (c) 使用原点平移法，将 A 数组的对角元减去 λ_1, 再使用幂法求出变换后矩阵的按模最大特征值 λ_{m2}, 此时 $\lambda_{501} = \lambda_{m2} + \lambda_1$

   (d) 使用反幂法即可求出 λ_s

2.2.2 第二问

2. 求 $\boldsymbol{A}$ 的与数 $\mu_k = \lambda_1 + k\dfrac{\lambda_{501} - \lambda_1}{40}$ 最接近的特征值 $\lambda_{i_k}(k = 1, 2, \cdots, 39)$。

   (a) 使用原点平移法，将 A 数组的对角元减去 μ_k, 再使用反幂法求出变换后矩阵的按模最大特征值 λ_m, 此时 $\lambda_{i_k} = 1/\lambda_m + \mu_k$

2.2.3 第三问

3. 求 $\boldsymbol{A}$ 的 (谱范数) 条件数 $cond(\boldsymbol{A})_2$ 和行列式 $det\boldsymbol{A}$

   (a) $cond(\boldsymbol{A})_2 = \left|\dfrac{\lambda_{m1}}{\lambda_s}\right| = \left|\dfrac{\lambda_{501}}{\lambda_s}\right|$

   (b) 使用 $Doolittle$ 算法将 A 数组分解成 $A = L \times U$, 于是有:

$$det\boldsymbol{A} = det\boldsymbol{L} \times det\boldsymbol{U} = \prod_{k=1}^{n} u_{kk}$$

源程序 3

3.1 C 语言版大作业

```c
#include<stdio.h>
#include<math.h>
#include<string.h>
#define N 510
double u[N],y[N],c[10][N];
const double eps=1e-12;
int n=501;

double A(int i,int j){
	int k=i-j+3;
	if(k<=0)
		return 0;
	return c[k][j];
}

double mmax(double U[]){
	double m=0;
	for(int i=1;i<=n;i++)
		if(fabs(m)<fabs(U[i]))
			m=U[i];
	return m;
}

int max(int x,int y){
	if(x>y)
```

```
        return x;
    return y;
}

int min(int x,int y){
    if(x<y)
        return x;
    return y;
}

void def(double s){
    for(int j=1;j<=n;j++){
        c[3][j]=(1.64-0.024*j)*sin(0.2*j)-0.64*exp(0.1/j)
            -s;
        c[2][j]=c[4][j]=0.16;
        c[1][j]=c[5][j]=-0.064;
    }

}


void put(double U[N]){
    for(int i=1;i<=n;i++)
        printf("%lf ,",U[i]);
    printf("\n");
}

void Doolittle(){
    for(int k=1;k<=n;k++){
        for(int j=k;j<=min(k+2,n);j++)
            for(int t=max(1,j-2);t<k;t++)
                c[k-j+3][j]-=A(k,t)*A(t,j);
        for(int i=k+1;i<=min(k+2,n);i++){
            for(int t=max(1,i-2);t<k;t++)
                c[i-k+3][k]-=A(i,t)*A(t,k);
            c[i-k+3][k]/=A(k,k);
        }
    }
}

void Slove(double X[],double b[]){
    for(int i=1;i<=n;i++){
        X[i]=b[i];
        for(int t=max(1,i-2);t<i;t++)
            X[i]-=A(i,t)*X[t];
    }
```

```
    for(int i=n;i;i--){
        for(int t=i+1;t<=min(i+2,n);t++)
            X[i]-=A(i,t)*X[t];
        X[i]/=A(i,i);
    }
}


void Normalization(){
    double s=0;
    for(int i=1;i<=n;i++)
        s+=u[i]*u[i];
    s=sqrt(s);
    for(int i=1;i<=n;i++)
        y[i]=u[i]/s;
}

void Multiplication(double U[N]){
    memset(u,0,sizeof(u));
    for(int i=1;i<=n;i++)
        for(int j=max(1,i-2);j<=min(n,i+2);j++)
            u[i]+=A(i,j)*U[j];
}

double PowerMethod(){
    double B=0,b=0,e=1;
    for(int i=1;i<=n;i++)
        u[i]=1;
    while(e>eps){
        Normalization();
        Multiplication(y);
        B=0;
        for(int i=1;i<=n;i++)
            B+=y[i]*u[i];
        e=fabs((B-b)/B);
        b=B;
    }
    return B;
}

double InversePowerMethod(){
    double B=0,b=0,e=1;
    for(int i=1;i<=n;i++)
        u[i]=1;
    Doolittle();
    while(e>eps){
```

```
            Normalization();
            Slove(u,y);
            B=0;
            for(int i=1;i<=n;i++)
                B+=y[i]*u[i];
            e=fabs((B-b)/B);
            b=B;
        }
        return B;
}

int main(){
        double lambda_1,lambda_S,lambda_501,uk,det=1;
//      freopen("input.in","r",stdin);
//      freopen("Works.out","w",stdout);
        memset(c,0,sizeof(c));
        memset(u,0,sizeof(u));
        def(0);
        lambda_1=PowerMethod();
        lambda_S=1/InversePowerMethod();
        for(int i=1;i<=n;i++)
            det*=A(i,i);
        def(lambda_1);
        lambda_501=PowerMethod()+lambda_1;
        printf("(1):\n");
        printf("Lambda_1=%.12e\n",lambda_1);
        printf("Lambda_S=%.12e\n",lambda_S);
        printf("Lambda_501=%.12e\n",lambda_501);
        printf("\n\n------------------------------------------------------------\n(2):\n")
            ;
        for(int k=1;k<=39;k++){
            uk=lambda_1+k*(lambda_501-lambda_1)/40;
            def(uk);
            printf("Lambda_i%d=%.12e\n",k,1/
                InversePowerMethod()+uk);
        }
        printf("\n\n------------------------------------------------------------\n(3):\n")
            ;
        printf("Cond(A)=%.12e\n",fabs(lambda_1/lambda_S));
        printf("Det(A)=%.12e\n",fabs(det));
        return 0;
}
```

3.2 MATLAB 版幂法

```
fid=fopen('A_out.txt','w');
for mm=1:501
n=501;
a=zeros(n);
u=zeros(n,1);
tol = 0.00000001 ;
u(mm)=1;
for i = 1:n
    for j = 1:n
        if(abs(i-j)==0)
            a(i,j)=(1.64-0.024.*i).*sin(0.2.*i)-0.64.*exp(0.1./i);
        elseif(abs(i-j)==1)
            a(i,j)=0.16;
        elseif(abs(i-j)==2)
            a(i,j)=-0.064;
        end
    end
end
e=1;
Bk=0;
while max ( abs ( e ) ) > tol
    b=abs(u);
    r=find (b==max(b))
    hr=u(r(1));
    y=u./abs(hr) ;
    u=a*y ;
    B=sign(hr).*u(r(1));
    e=(B-Bk)/B;
    Bk=B;
end
fprintf(fid,'Lambda=%f\n',B);
end
fclose(fid);
```

3.3 MATLAB 版反幂法

```
fid=fopen('B_out.txt','w');
n=501;
a=zeros(n);
u=zeros(n,1);
tol = 0.00000001 ;
for i = 1:n
     u(i)=1;
end
for i = 1:n
    u(1)=1;
    for j = 1:n
        if(abs(i-j)==0)
            a(i,j)=(1.64-0.024.*i).*sin(0.2.*i)-0.64.*exp
               (0.1./i);
        elseif(abs(i-j)==1)
            a(i,j)=0.16;
        elseif(abs(i-j)==2)
            a(i,j)=-0.064;
        end
    end
end
e=1;
Bk=0;
while max ( abs ( e ) ) > tol
    s=sqrt(u'*u);
    y=u./s
    u=inv(a)*y;
    B=y'*u;
    e=(B-Bk)/B;
    Bk=B;
end
fprintf(fid,'Lambda=%f\n',B);
fclose(fid);
```

计算结果 | 4

程序运行结果如下：

(1):

$\lambda_1 = -1.070011361502e + 001$

$\lambda_S = -5.557910794230e - 003$

$\lambda_{501} = 9.724634098777e + 000$

---

(2):

$\lambda_{i1} = -1.018293403315e + 001$

$\lambda_{i2} = -9.585707425068e + 000$

$\lambda_{i3} = -9.172672423928e + 000$

$\lambda_{i4} = -8.652284007898e + 000$

$\lambda_{i5} = -8.093483808675e + 000$

$\lambda_{i6} = -7.659405407692e + 000$

$\lambda_{i7} = -7.119684648691e + 000$

$\lambda_{i8} = -6.611764339397e + 000$

$\lambda_{i9} = -6.066103226595e + 000$

$\lambda_{i10} = -5.585101052628e + 000$

$\lambda_{i11} = -5.114083529812e + 000$

$\lambda_{i12} = -4.578872176865e + 000$

$\lambda_{i13} = -4.096470926260e + 000$

$\lambda_{i14} = -3.554211215751e + 000$

$\lambda_{i15} = -3.041090018133e + 000$

$\lambda_{i16} = -2.533970311130e + 000$

$\lambda_{i17} = -2.003230769563e + 000$

$\lambda_{i18} = -1.503557611227e + 000$

$\lambda_{i19} = -9.935586060075e - 001$

$\lambda_{i20} = -4.870426738850e - 001$

$\lambda_{i21} = 2.231736249575e - 002$

$\lambda_{i22} = 5.324174742069e - 001$

$\lambda_{i23} = 1.052898962693e + 000$

$\lambda_{i24} = 1.589445881881e + 000$

$\lambda_{i25} = 2.060330460274e + 000$

$\lambda_{i26} = 2.558075597073e + 000$
$\lambda_{i27} = 3.080240509307e + 000$
$\lambda_{i28} = 3.613620867692e + 000$
$\lambda_{i29} = 4.091378510451e + 000$
$\lambda_{i30} = 4.603035378279e + 000$
$\lambda_{i31} = 5.132924283898e + 000$
$\lambda_{i32} = 5.594906348083e + 000$
$\lambda_{i33} = 6.080933857027e + 000$
$\lambda_{i34} = 6.680354092112e + 000$
$\lambda_{i35} = 7.293877448127e + 000$
$\lambda_{i36} = 7.717111714236e + 000$
$\lambda_{i37} = 8.225220014050e + 000$
$\lambda_{i38} = 8.648666065193e + 000$
$\lambda_{i39} = 9.254200344575e + 000$

---

(3):
Cond(A)=1.925204273902e+003
Det(A)=2.772786141752e+118

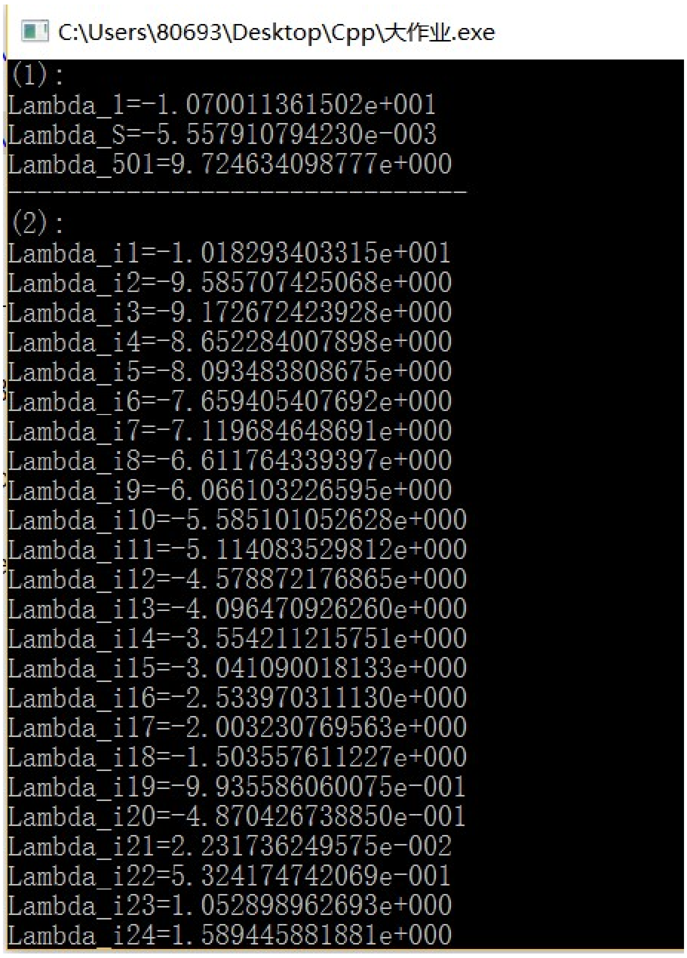
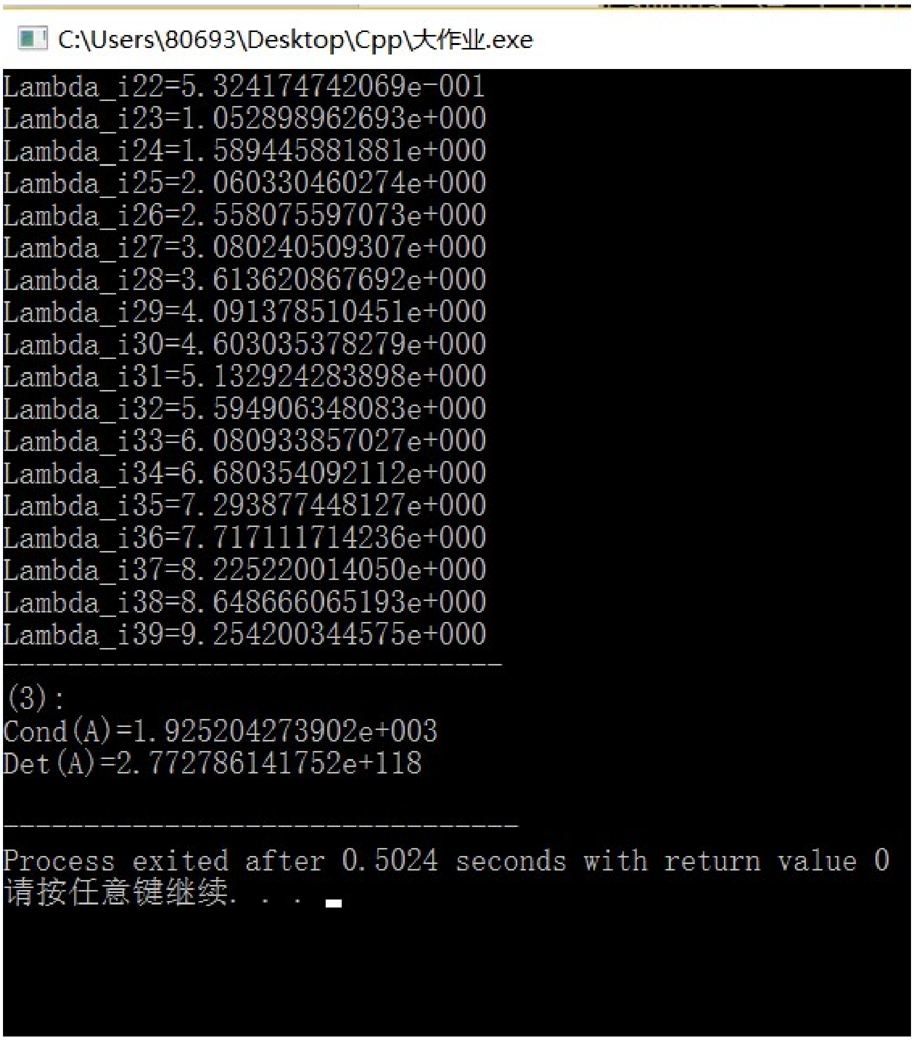


图 4.1: Result1


C:\Users\80693\Desktop\Cpp\大作业.exe
Lambda_i22=5.324174742069e-001
Lambda_i23=1.052898962693e+000
Lambda_i24=1.589445881881e+000
Lambda_i25=2.060330460274e+000
Lambda_i26=2.558075597073e+000
Lambda_i27=3.080240509307e+000
Lambda_i28=3.613620867692e+000
Lambda_i29=4.091378510451e+000
Lambda_i30=4.603035378279e+000
Lambda_i31=5.132924283898e+000
Lambda_i32=5.594906348083e+000
Lambda_i33=6.080933857027e+000
Lambda_i34=6.680354092112e+000
Lambda_i35=7.293877448127e+000
Lambda_i36=7.717111714236e+000
Lambda_i37=8.225220014050e+000
Lambda_i38=8.648666065193e+000
Lambda_i39=9.254200344575e+000
----
(3):
Cond(A)=1.925204273902e+003
Det(A)=2.772786141752e+118

Process exited after 0.5024 seconds with return value 0
请按任意键继续. . .

图 4.2: Result2

讨论 5

当我第一次运行幂法的程序时,所使用的初始向量为 u[1]=1,u[i]=0(i=2,3···500),所得到的结果为 $\lambda_1 = -2.080981085336e + 000$，而此时我已知正确答案是 $\lambda_1 = -1.070011361502e + 001$。

反复检查程序后，并没有发现程序出现问题，为了验证，我用 MATLAB 程序也写了一个幂法求特征值的程序，然而跑出来的 λ_1 的值依旧为-2.08。由于 MATLAB 程序简单明了，出现错误的几率很小。于是我又写了个程序，随机生成 $N \times N$ 的稀疏矩阵，并将其程序的计算结果和 Mathematica 计算出的结果比较，几乎全部吻合，这让我更困惑了。于是我用 Mathematica 将所有的特征值都求了出来，仔细分析，发现-2.08 恰好也是该矩阵的特征值，那么显然是算法出了问题，使得前面的特征值都被跳过了。

那么算法的问题出在哪里呢？我们再仔细分析算法，观察以下两个式子

$$u_0 = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n$$

$$u_k = \lambda_1^k \left[\alpha_1 x_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k x_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k x_n \right]$$

向量 $\boldsymbol{u}_0$ 被分解成特征向量 $\boldsymbol{x}_k(k = 1, 2, \cdots, n)$ 的线性组合,但当 $\alpha_1 = 0, \alpha \neq 0$ 时，

$$\lim_{k \to \infty} u_k \approx \lambda_1^k a_2 \boldsymbol{x}_1$$

那么，程序的问题应该是 $\alpha_1 = \alpha_2 = \cdots = 0$，使得-2.08 前的特征向量的 α_k 的值都为 0。

结论：虽然算法上说 $\boldsymbol{u}_0$ 的值是可以任取的，但当初始向量 $\boldsymbol{u}_0$ 中的 0 元素较多时，将使得比较多的 $\alpha_i = 0$，此时计算出的结果与真实值差距较大。

```
Out[3]= {-10.7001, -10.3071, -10.1829, -9.95398, -9.81077, 9.72463, -9.58571, -9.58111, 9.49789, -9.41064, 9.2542,
  -9.20855, -9.17267, 8.97234, -8.92808, 8.87088, -8.84711, -8.81385, 8.77593, -8.65228, 8.64867, -8.60425,
  -8.49806, -8.46408, 8.44421, -8.40503, 8.31592, 8.22522, -8.09348, -8.08683, 8.01986, -7.97524, 7.96447,
  -7.86238, -7.81299, 7.79762, -7.71977, 7.71711, -7.7007, -7.65941, 7.62462, 7.61439, -7.56694, 7.47416,
  7.40201, -7.37469, 7.34116, -7.32101, -7.30507, 7.29388, -7.20708, -7.14786, -7.11968, 7.0165, -6.97658,
  -6.97148, 6.85256, -6.82548, -6.82242, 6.7256, 6.70687, 6.68035, -6.63522, -6.63292, -6.62326, -6.61176,
  6.58162, 6.55172, 6.52802, 6.5245, -6.48791, 6.42925, 6.31761, -6.26814, 6.25621, 6.24728, -6.23416,
  -6.18267, -6.17113, 6.08093, -6.0661, -6.02703, 5.97662, -5.97409, -5.90781, -5.8673, -5.86394, 5.85511,
  -5.82342, 5.81492, -5.74535, -5.72564, -5.72018, -5.62687, 5.59491, -5.5851, -5.57542, 5.56789, -5.49291,
  5.47065, 5.46316, 5.40731, -5.39719, 5.36292, 5.35213, 5.33177, 5.22746, -5.19966, -5.1825, 5.18031, 5.16,
  5.14458, 5.13292, -5.11408, 5.08212, -4.98995, -4.95343, -4.91425, -4.89741, -4.87964, -4.8716, 4.85699,
  -4.83833, 4.81881, -4.79744, -4.75311, -4.71666, -4.71424, 4.69836, -4.60522, 4.60304, 4.58914, -4.57887,
  -4.549, 4.53459, 4.47929, -4.46146, 4.445, -4.43481, -4.42108, 4.39779, 4.3452, 4.29268, -4.27441,
  -4.25001, 4.17884, 4.17322, 4.15422, -4.15155, -4.12263, -4.09783, -4.09647, 4.09138, 4.05296, 4.04241,
  -4.01522, -4.00443, -4.00429, -3.98951, 3.92808, -3.84803, 3.83759, -3.81989, -3.80242, 3.79387, 3.79068,
  -3.76677, -3.74742, -3.7376, 3.73093, -3.71358, 3.66006, 3.63691, -3.63587, 3.61362, -3.55421, -3.54324,
  3.52143, 3.4875, -3.47937, 3.43719, 3.42921, -3.42898, 3.42826, 3.38868, -3.35558, -3.31369, -3.31337,
  -3.29541, -3.28007, 3.26886, -3.23215, -3.21979, -3.2094, -3.20751, 3.1098, 3.08024, 3.05464, -3.04109,
  3.02573, 3.00822, 2.98325, -2.97777, -2.9697, 2.9691, -2.91159, -2.90633, -2.90562, 2.88181, -2.84917,
  2.83965, 2.81604, 2.79428, 2.79215, -2.78431, -2.77759, -2.73186, -2.72551, 2.7202, 2.71892, -2.6919,
  -2.61172, 2.60805, 2.60346, -2.56236, 2.55808, -2.54933, -2.54553, -2.53397, -2.52643, 2.5218, -2.52175,
  -2.51917, 2.51847, 2.51098, -2.50572, -2.48663, -2.41565, 2.41442, -2.34713, -2.28253, 2.25005, 2.23614,
  2.21179, -2.20898, -2.19696, -2.17483, 2.15243, -2.13326, -2.12894, -2.10877, 2.09959, -2.09338, 2.08123,
  -2.08098, 2.06033, 2.05949, -2.04919, -2.03905, 2.01366, 2.01303, 2.00427, -2.00323, -1.98416, 1.9757,
  -1.92629, -1.921, 1.9156, -1.91418, -1.90934, -1.90259, 1.86914, -1.86494, -1.8458, 1.84217, -1.82768,
  1.80447, -1.75852, -1.74577, 1.73919, 1.73795, 1.72317, -1.72208, -1.71884, -1.67587, -1.66277, 1.65579,
  -1.63034, -1.63023, -1.62329, 1.61579, -1.61398, -1.60718, 1.58945, -1.5407, -1.53342, -1.52876, -1.50356,
  -1.49675, 1.49471, 1.48998, -1.48796, 1.46506, -1.4607, -1.452, -1.43051, 1.42594, -1.42248, 1.40798,
  -1.40787, -1.40242, -1.3789, -1.37407, 1.3696, -1.33621, -1.32278, 1.31354, 1.31162, 1.30124, 1.28625,
  -1.27929, -1.26897, -1.25277, -1.23771, -1.23095, -1.22487, 1.2221, -1.2055, 1.20122, -1.19471, 1.19409,
  -1.1766, -1.15955, -1.15286, 1.14525, -1.13638, -1.13327, -1.12144, 1.11934, -1.10547, 1.10079, -1.09927,
  -1.09842, -1.0752, -1.07444, 1.06991, 1.0529, -1.03739, -1.03714, -1.03362, -1.0278, -1.01102, -1.00643,
  0.997875, -0.993558, -0.981996, -0.969872, 0.966569, -0.95589, -0.947962, 0.947644, -0.933208, -0.932024,
  -0.904608, -0.900397, 0.897839, -0.896544, -0.873162, 0.866494, -0.860745, 0.851963, -0.836319, -0.82653,
  0.82166, 0.816792, -0.81383, 0.812931, -0.794815, 0.787377, 0.774831, -0.771913, -0.771677, -0.766859,
  -0.759525, -0.754323, 0.7526, 0.745858, -0.744284, -0.736796, -0.725538, -0.724097, -0.71243, 0.70753,
  -0.705252, 0.703951, 0.702504, -0.69528, 0.695055, -0.689882, -0.673687, -0.665718, -0.653484, -0.650019,
  0.647429, -0.631725, -0.626566, 0.619797, -0.609965, -0.608754, -0.604469, -0.579782, 0.562311, -0.559951,
  -0.55833, 0.54658, -0.539791, -0.536004, -0.533331, -0.533323, 0.532417, 0.527925, 0.518298, -0.517802,
  -0.517334, -0.508897, 0.501184, -0.487043, -0.484041, -0.475872, -0.473441, -0.469121, -0.463744, 0.46143
  -0.458461, -0.45285, 0.450507, -0.447866, 0.439304, 0.438248, -0.435231, -0.429503, -0.427952, -0.419276,
  -0.411983, -0.405262, -0.404887, -0.3991, 0.376447, 0.366319, -0.358633, -0.358342, -0.35431, 0.345805,
  -0.339889, -0.334532, -0.309127, -0.288772, 0.277525, 0.264516, 0.261417, -0.257543, 0.255618, 0.241192,
  0.238744, -0.232566, 0.231174, -0.226351, -0.22053, -0.213258, -0.204778, -0.195264, 0.19403, 0.193313,
  0.189445, -0.178814, 0.177581, -0.171638, 0.16761, -0.16031, -0.151996, 0.121248, -0.121009, 0.118085,
  0.115331, 0.104311, -0.100365, -0.092142, -0.0746513, -0.0713478, -0.0539957, -0.0477676, 0.0472612,
  -0.0343367, -0.0335993, 0.0266891, -0.0223811, 0.0223169, 0.0129293, 0.0103096, -0.00637492, -0.00555794}
```

图 5.1: Result of Eigenvalues