応用数理実験

固有値問題 (1) 固有値問題に対する冪乗法

桂田 祐史

2005 年 6 月 28 日

今回は固有値問題を取り扱うための冪乗法とその周辺の種々の手法を実験してもらう。
実験に用いる行列は小さなもので良い。自分で選ぶのが面倒なら、次の行列を使ってよい[1]。

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

- 選んだ行列のサイズが小さい場合は、固有多項式を計算して固有値に関する情報を調べよ[2]。サイズが小さくない場合は、Mathematica か (後で例をあげる)、MATLAB (Octave) の関数 `eig()` を用いて固有値を調べよ。
- 冪乗法により絶対値最大の固有値とそれに属する固有ベクトルを求めよ。収束の速さを調べよ。固有値を反復ベクトルの成分の比として計算した場合と、 Rayleigh 商で計算した場合とで、収束の速さに違いが出るか？
- 逆反復法により (連立 1 次方程式を解くのには LU 分解を使え)、絶対値最小の固有値とそれに属する固有ベクトルを求めよ。
- シフト法の実験をせよ。すなわち絶対値最大、最小の固有値以外の固有値を見い出せ。(例えば、上に掲げた行列 A に対しては 0.6 に近い固有値を探してみよ。)
- (余裕がある人向け) 絶対値最大の固有値に属する固有ベクトルを求めてから、その成分を含まない初期ベクトルを用いて冪乗法の反復を実行することにより、絶対値が二番目に大きな固有値の固有ベクトルを求めるアルゴリズムがある。適当な文献で調べるか、自分で考えて実験してみよ。
- (余裕がある人向け) 絶対値最大の固有値が二つ以上ある場合、冪乗法で何が起こるか観察せよ。

[1] この行列自体が一松信「数値解析」朝倉書店 (1982) から採ったものである。余談だが、この本は基本的な数値計算アルゴリズムに関して概観するには便利な本である。固有値は $\lambda_1 = 5.048917339522305313522214407023370089866\cdots$, $\lambda_2 = 0.643104132107790556105600489978699461920 1\cdots$, $\lambda_3 = 0.307978528369904130372185102997930881547 9\cdots$.

[2] 必要があれば Mathematica 等を利用せよ。その場合のヒント: 行列式を計算する `Det[]`, 指定された次数の単位行列を作る `IdentityMatrix[]`, 固有値を計算する `Eigenvalues[]`, 方程式を解く `Solve[]`, 方程式の解の近似値を求める `FindRoot[]`, 関数のグラフを描く `Plot[]` 等の手続きがある。

Mathmatica の使用例

```
oyabun% math
Mathematica 4.0 for Solaris
Copyright 1988-1999 Wolfram Research, Inc.
 -- Motif graphics initialized --

In[1]:= A={{1,1,1},{1,2,2},{1,2,3}}

Out[1]= {{1, 1, 1}, {1, 2, 2}, {1, 2, 3}}

In[2]:= Eigenvalues[A]

Out[2]= (省略 --- 興味があればやってみて下さい)

In[3]:= N[%,40]
                                                                  -46
Out[3]= {5.048917339522305313522214407023369723596 + 0. 10    I,

                                                                -42
>    0.307978528369904130372185102997930858027 + 0. 10    I,

                                                                -42
>    0.643104132107790556105600489978699416600 9 + 0. 10    I}

In[4]:= na=N[A,50]

Out[4]= {{1.0000000000000000000000000000000000000000000000000,
>     1.0000000000000000000000000000000000000000000000000,
>     1.0000000000000000000000000000000000000000000000000},
>    {1.0000000000000000000000000000000000000000000000000,
>     2.0000000000000000000000000000000000000000000000000,
>     2.0000000000000000000000000000000000000000000000000},
>    {1.0000000000000000000000000000000000000000000000000,
>     2.0000000000000000000000000000000000000000000000000,
>     3.0000000000000000000000000000000000000000000000000}}

In[5]:= Eigenvalues[na]

Out[5]= {5.048917339522305313522214407023369723596 3877860565,
>    0.643104132107790556105600489978699416600 87281326538,
>    0.307978528369904130372185102997930859802 73940067811}

In[6]:= Eigenvectors[na]

Out[6]= {{-0.327985277605681767796032025004549906408705321 12133,
>     -0.591009048506103525457945717999379808447138819 55014,
>     -0.736976229099578242338086307005170097961566501 57119},
>    {0.736976229099578242338086307005170097961566501 57119,
>     0.327985277605681767796032025004549906408705321 12133,
>     -0.591009048506103525457945717999379808447138819 55014},
>    {-0.591009048506103525457945717999379808447138819 55014,
>     0.736976229099578242338086307005170097961566501 57119,
>     -0.327985277605681767796032025004549906408705321 12133}}

In[7]:= NSolve[Det[x IdentityMatrix[3]-A]==0, x, WorkingPrecision->30]}

Out[7]= {{x -> 0.307978528369904130372185103 00},
         {x -> 0.643104132107790556105600489 98},
         {x -> 5.048917339522305313522214407 0}}
```

A Octave のサンプル・プログラム

グラフ化の説明もかねて Octave のサンプル・プログラムを以下に示す。なお、MATLAB でも同じプログラムが走ることを確認済み。

幂乗法

```
format long              % 表示桁数を多く取る
a=[1 1 1;1 2 2;1 2 3]
eigen=eig(a)             % eig() で固有値を求める（カンニング）
x=ones(3,1);             % 初期ベクトルを（1,1,1）とする
maxiter=10;
r=zeros(maxiter,1);      % 近似固有値を記録するベクトルの用意
for k=1:maxiter
  y=a*x;
  x=y/norm(y);
  r(k)=x'*(a*x);         % Rayleigh 商で近似固有値を計算し、r() に記録
end
error=r-max(eigen)               % 近似固有値の誤差を求める
plot(1:maxiter,abs(error))       % 反復回数と誤差をプロット
loglog(1:maxiter,abs(error))     % 両側対数目盛りでプロット
semilogy(1:maxiter,abs(error)) % 片側対数目盛りでプロット
```

B MATLAB メモ

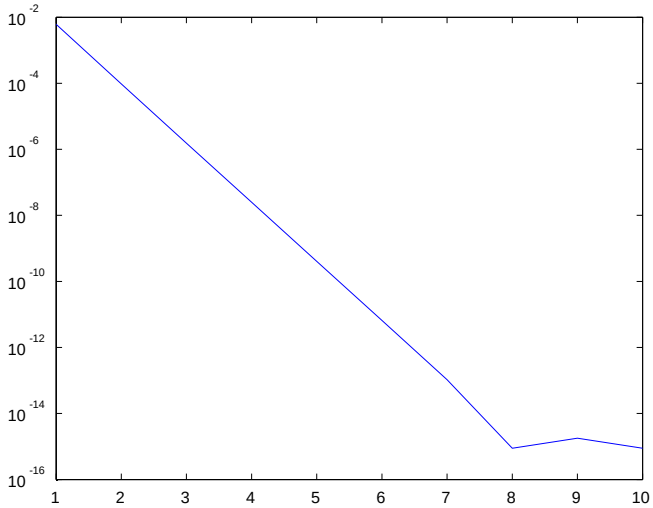
- 6701 号室の Windows XP マシンでは、M-file は Z:¥ か、あるいはその下、例えば Z:¥matlab に作ることを勧める。[File] メニューの [Set Path] サブメニューで [Add Folder] しておく。
- 2005 年度の情報処理教室のパソコンには、MATLAB や Octave がインストールされている。前者はライセンスの問題でつねに利用できるとは限らないが…

C　自分でやってみた

MATLAB を使って実験してみた。非常に気楽にプログラムを書いて出来てしまう。そろそろ課題を新しくして、この古い課題のプログラムをサンプルとして提示する形にするのが良いかもしれない。

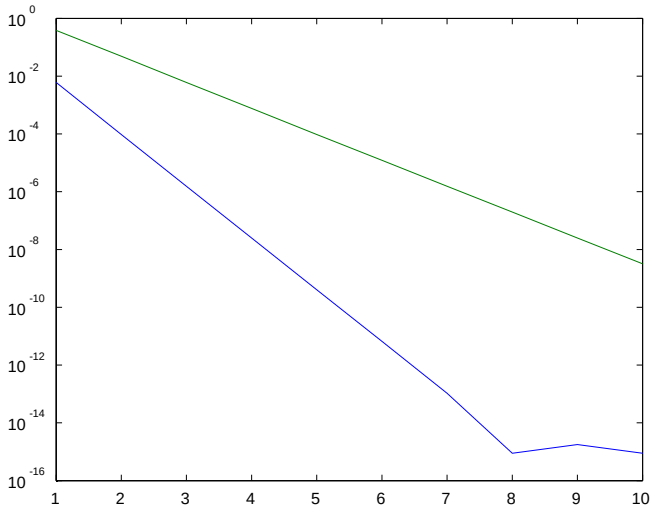
C.1　冪乗法

```
% eigen1.m --- power method
format long
a=[1 1 1;1 2 2;1 2 3]
eigen=eig(a)
x=ones(3,1);
maxiter=10;
r=zeros(maxiter,1);
for k=1:maxiter
    y=a*x;
    x=y/norm(y);
    r(k)=x'*(a*x);
end
r
error=r-max(eigen)
clf
semilogy(1:maxiter,abs(error))
```



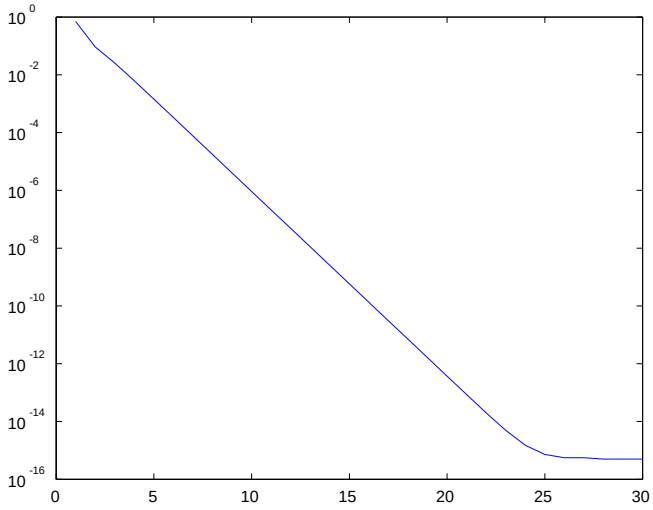
C.2 Rayleigh 商の効果

```
% naive_vs_rayleigh.m
format long
a=[1 1 1;1 2 2;1 2 3]
eigen=eig(a)
x=ones(3,1);
maxiter=10;
r=zeros(maxiter,1);
n=zeros(maxiter,1);
for k=1:maxiter
    y=a*x;
    x=y/norm(y);
    ax=a*x;
    r(k)=x'*ax;
    n(k)=ax(1)/x(1);
end
r
n
error_r=r-max(eigen)
error_n=n-max(eigen)
%subplot(1,2,1)
%semilogy(1:maxiter,abs(error_r))
%subplot(1,2,2)
%semilogy(1:maxiter,abs(error_n))
clf
semilogy(1:maxiter,abs(error_r),1:maxiter,abs(error_n))
```



C.3 逆反復法

```
% eigen2.m --- inverse iteration method
format long
a=[1 1 1;1 2 2;1 2 3]
eigen=eig(a)
x=ones(3,1);
maxiter=30;
r=zeros(maxiter,1);
[L u p]=lu(a);
shift=1;
for k=1:maxiter
    y=u\(L\(p*x));
    x=y/norm(y);
    r(k)=x'*(a*x);
end
r
error=r-min(eigen)
clf
semilogy(1:maxiter,abs(error))
```



C.4　シフト法

```
% eigen3.m --- shift method
format long
a=[1 1 1;1 2 2;1 2 3]
eigen=eig(a)
x=ones(3,1);
maxiter=10;
r=zeros(maxiter,1);
shift=0.6;
% we cannot use the factorization of a, can we?
[L u p]=lu(a-shift*eye(3,3));
for k=1:maxiter
    y=u\(L\(p*x));
    x=y/norm(y);
    r(k)=x'*(a*x);
end
r
eigen=sort(eigen);
error=r-eigen(2)
clf
semilogy(1:maxiter,abs(error))
```