応用数理実験

固有値問題 (2) 実対称行列の三重対角化

桂田 祐史

2005 年 7 月 12 日

1 レポートの提出について

- 締め切りは　月　日
- 提出は紙 (桂田本人に手渡しするか、数学科資料室 6606 号室に提出) でも、E-mail (mk@math.meiji.ac.jp) でもどちらでも構わない。
- 質問は締め切り直前まで受け付ける。連絡は E-mail でどうぞ。
- これまでの課題は
  1. 連立 1 次方程式に対する直接法
  2. CG 法
  3. 固有値問題に対する冪乗法
  4. 実対称行列の三重対角化

  の 4 つ。プリントのコピーが欲しければ `http://www.math.meiji.ac.jp/~mk/lecture/suurikaisekitokuron/` にある。

2 今回の課題

次の二つの課題のうち、いずれか一つを選択する。プログラミング言語は何を採用しても良い。

2.1 実対称行列の三重対角化

(1) Householder 変換または Lanczos 法により実対称行列を三重対角行列に相似変換するプログラムを作成せよ (相似変換のための行列は求めなくとも構わない)。

(2) 実対称三重対角行列に対して (その特性を十分生かして)、冪(べき)乗法、逆反復法により、絶対値が最大の固有値と絶対値最小の固有値を求めるプログラムを作成し、(1) で作ったプログラムと結合せよ。

(3) 三重対角行列に対して、二分法により固有値を求めるプログラムを作成し、実験せよ。

三重対角化のプログラムのテスト用のデータとしては、例えば

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

を用いよ[1]。これを講義で説明した Householder 法[2]で三重対角化すると

$$P^{-1}AP = \begin{pmatrix} 1.0000000000 & -1.7320508076 & 0 & 0 \\ -1.7320508076 & 7.6666666667 & 1.2472191289 & 0 \\ 0 & 1.2472191289 & 0.9761904762 & -0.1237179148 \\ 0 & 0 & -0.1237179148 & 0.3571428571 \end{pmatrix}$$

となる。さらに固有値は

$$\begin{aligned} \lambda_1 &= 8.290859369381589606621222410409759227\cdots, \\ \lambda_2 &= 1, \\ \lambda_3 &= 0.426022047760461836484914938273277876 12\cdots, \\ \lambda_4 &= 0.283118582857948556893862651316962896 25\cdots, \end{aligned}$$

となる。

プログラムの正しさに自信が持てたら大きな行列で実験してみること。

2.2 QR 法の数値実験

MATLAB (あるいは Octave) を使って QR 法の実験を行なう。QR 分解をする関数は `qr()` である。

- 行列 A の構造 (実対称性や帯行列であることなど) は QR 変換で保存されるかどうか、数値計算の結果を観察して推測せよ (もちろん証明できれば良い)。
- 対角成分を固有値の近似値として採用するとして、収束の速さを調べよ。

[1] この行列も前回と同様、一松信著「数値解析」朝倉書店 (1982) から採った。

[2] Lanczos 法を用いた場合も、相似変換のための直交行列の第 1 列ベクトルを $\mathbf{e}_1 = (1, 0, \cdots, 0)^T$ に取ると、(符号の違いを無視すれば) 同じ結果になるはずである。

2.3 参考: ちょっと自分でやってみました

```
% ランダム Hessenberg 行列
function ret = rand_h(n)
  ret = triu(rand(n,n)) + diag(rand(n-1,1),-1);

% ランダム三重対角行列
function ret = rand_t(n)
  ret=diag(rand(n,1),0)+diag(rand(n-1,1),1)+diag(rand(n-1,1),-1);

% ランダム実対称三重対角行列
function ret = rand_st(n)
  u=rand(n-1,1);
  ret=diag(rand(n,1),0)+diag(u,1)+diag(u,-1);

% ランダム実対称行列
function ret = rand_s(n)
  a=rand(n,n);
  ret = (a+a')/2;

% QR 変換 (回数指定)
function QR = qr_iteration(a,n)
  QR = a;
  for i=1:n
    [q r]=qr(QR);
    QR=r*q;
  end

% 下三角部分のノルム
function n = norm_l(a)
  n = norm(a-triu(a));
```

3 参考 — 三重対角行列に対する Gauss の消去法

3.1 C 言語版

三重対角行列を LU 分解する関数 trilu() と、その結果を元に後退代入して連立 1 次方程式を解く関数 trisol() を納めた trid-lu.c, trid-lu.h (とそれらの "添字が 1 から始まる" バージョン trid-lu1.c, trid-lu1.h) を公開している (http://www.math.meiji.ac.jp/~mk/program/ から入手できる)。

関数 **trid()** の使用例

```
1  /*
2   * test-lu.c --- 三重対角行列係数の連立 1 次方程式を解く trilu(), trisol()
3   *                のテスト
4   *   コンパイル＆リンク: gcc -o test-lu test-lu.c trid-lu.c
5   *   trid-lu.c, trid-lu.h の入手は以下の URL で
6   *   http://www.math.meiji.ac.jp/~mk/program/linear/trid-lu.c
7   *   http://www.math.meiji.ac.jp/~mk/program/linear/trid-lu.h
```

```
 8   */
 9
10  #define NDIM 100
11
12  #include <stdio.h>
13  #include <math.h>
14  #include "trid-lu.h"
15
16  int main()
17  {
18    int i, n, nm1;
19    double al[NDIM],ad[NDIM],au[NDIM],b[NDIM],x[NDIM];
20    n = 10; nm1 = n - 1;
21    /* 係数行列 A */
22    ad[0] = 2.0; au[0] = -1.0;
23    for (i = 1; i < nm1; i++) {
24      al[i] = -1.0; ad[i] = 2.0; au[i] = - 1.0;
25    }
26    al[nm1] = - 1.0; ad[nm1] = 2.0;
27    /* 解 x */
28    for (i = 0; i < n; i++)
29      x[i] = i;
30    /* 右辺 */
31    b[0] = ad[0] * x[0] + au[0] * x[1];
32    for (i = 1; i < nm1; i++)
33      b[i] = al[i] * x[i-1] + ad[i] * x[i] + au[i] * x[i+1];
34    b[nm1] = al[nm1] * x[nm1-1] + ad[nm1] * x[nm1];
35    /* A を LU 分解する */
36    trilu(n, al, ad, au);
37    /* 後退代入して連立 1 次方程式 A x=b を解く */
38    trisol(n, al, ad, au, b);
39    /* 結果を表示する */
40    for (i = 0; i < n; i++)
41      printf("%f\n", b[i]);
42    return 0;
43  }
```

```
oyabun% make
gcc -O -W -Wall   -c -o test-trid-lu.o test-trid-lu.c
gcc -O -W -Wall -o test-trid-lu test-trid-lu.o trid-lu.o
oyabun% ./test-trid-lu
-0.000000
1.000000
2.000000
3.000000
4.000000
5.000000
6.000000
7.000000
8.000000
9.000000
oyabun%
```

A　MATLAB で Householder 法プログラムを書いて遊ぶ

ガラクタ箱だけど、後で使おう。

naive_householder.m

```
function H=naive_householder(A)
[N N]=size(A);
for k=1:N-2
 b=A(k+1:N,k);
 B=A(k+1:N,k+1:N);
 s=-sign(b(1))*norm(b);
 v=b;
 v(1)=v(1)-s;
 v=v/norm(v);
 Q=eye(N-k,N-k)-2*v*v';
 B=Q*B*Q;
 b=zeros(N-k,1);
 b(1)=s;
 A(k+1:N,k)=b;
 A(k,k+1:N)=b';
 A(k+1:N,k+1:N)=B;
end
H=A;
```

householder.m

```
function H=householder(A)
[N N]=size(A);
for k=1:N-2
 b=A(k+1:N,k);
 b1=b(1);
 B=A(k+1:N,k+1:N);
 s=-sign(b1)*norm(b);
 w=b;
 w(1)=w(1)-s;
 norm_w=2*(s*s-b1*s);
 p=B*w/(norm_w/2);
 alpha=w'*p/norm_w;
 q=p-alpha*w;
 B=B-w*q'-q*w';
 b=zeros(N-k,1);
 b(1)=s;
 A(k+1:N,k)=b;
 A(k,k+1:N)=b';
 A(k+1:N,k+1:N)=B;
end
H=A;
```

householder_exp2.m

```
%format long
A=[1 1 1 1;
   1 2 2 2;
   1 2 3 3;
   1 2 3 4]
householder(A)
```

householder_exp2.m の実行結果

```
>> householder_exp2

A =

     1     1     1     1
     1     2     2     2
     1     2     3     3
     1     2     3     4


ans =

    1.0000   -1.7321         0         0
   -1.7321    7.6667    1.2472         0
         0    1.2472    0.9762   -0.1237
         0         0   -0.1237    0.3571

>>
```

少し大きめの行列を作って、それを householder() で三重対角化してみた。ついでに固有値を計算して元の行列と変化がないことを確かめた (まあ相似変換が正しいことの状況証拠)。そのうち三重対角行列の固有値の計算をするプログラムを書こう。

householder_exp3.m

```
format short
N=10
A=zeros(N,N);
for i=1:N
   t=i*ones(N-i+1,1);
   A(i:N,i)=t;
   A(i,i:N)=t';
end
A
H=householder(A)
e1=sort(eig(A))
e2=sort(eig(H))
norm(e1-e2)
```

householder_exp3.m の実行結果

```
>> householder_exp3

N =

    10


A =

     1     1     1     1     1     1     1     1     1     1
     1     2     2     2     2     2     2     2     2     2
     1     2     3     3     3     3     3     3     3     3
     1     2     3     4     4     4     4     4     4     4
     1     2     3     4     5     5     5     5     5     5
     1     2     3     4     5     6     6     6     6     6
     1     2     3     4     5     6     7     7     7     7
     1     2     3     4     5     6     7     8     8     8
     1     2     3     4     5     6     7     8     9     9
     1     2     3     4     5     6     7     8     9    10


H =

  Columns 1 through 7

    1.0000   -3.0000         0         0         0         0         0
   -3.0000   40.6667  -11.9815         0         0         0         0
         0  -11.9815    7.8333    1.6583         0         0         0
         0         0    1.6583    2.2273    0.6166         0         0
         0         0         0    0.6166    1.1061    0.2998         0
         0         0         0         0    0.2998    0.6930    0.1625
         0         0         0         0         0    0.1625    0.4954
         0         0         0         0         0         0    0.0907
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 10

         0         0         0
         0         0         0
```

```
         0         0         0
         0         0         0
         0         0         0
         0         0         0
    0.0907         0         0
    0.3857    0.0484         0
    0.0484    0.3184   -0.0214
         0   -0.0214    0.2742


e1 =

    0.2557
    0.2738
    0.3080
    0.3662
    0.4652
    0.6431
    1.0000
    1.8730
    5.0489
   44.7661


e2 =

    0.2557
    0.2738
    0.3080
    0.3662
    0.4652
    0.6431
    1.0000
    1.8730
    5.0489
   44.7661


ans =

   1.4701e-14

>>
```

B MATLAB で QR 変換して遊んだときのガラクタ箱

(今度講義する羽目になったらちゃんと説明を書こう。)

```
% 『三重対角行列の QR 変換は三重対角ではないが、Hessenberg 形である。』
% つまり『Hessenberg の QR 変換は Hessenberg』ということかな？
% 三重対角行列のサンプルを作るには diag() が利用できる。
n=4; d=rand(n,1); u=rand(n-1,1); l=rand(n-1,1);
a=diag(d,0)+diag(u,1)+diag(l,-1)
for i=1:100
  [q r]=qr(a);
  a=r*q
```

```
end

% ちなみに関数にしておくと、
% function a = randtrid(n)
%   a=diag(rand(n,1),0)+diag(rand(n-1,1),1)+diag(rand(n-1,1),-1);

% 『実対称行列の QR 変換は実対称行列である。』
% 実対称行列のサンプルを作るには、三角部分の切り出し tril(), triu() の
% 利用も考えられるが、良く知られた命題
% 『任意の行列 A について、行列の「対称部分」 (A+A^T)/2 は対称行列になる。』
% を使うのが簡単だろう。
n=4;a=rand(n,n);a=(a+a')/2
for i=1:100
  [q r]=qr(a);
  a=r*q
end

% 『実対称三重対角行列の QR 変換は三重対角行列である。』
% (∵対称な Hessenberg 行列は三重対角であるから)
n=4; d=rand(n,1); u=rand(n-1,1); a=diag(d,0)+diag(u,1)+diag(u,-1)
for i=1:100
  [q r]=qr(a);
  a=r*q
end

% 最終的な結果と、eig() を使って得た近似固有値との比較
% a を変換してしまう前に近似固有値を計算して保存しておく必要がある。
n=4; d=rand(n,1); u=rand(n-1,1); a=diag(d,0)+diag(u,1)+diag(u,-1)
e=eig(a);
for i=1:100
  [q r]=qr(a);
  a=r*q
end
a
e

% 毎回、対角成分を eig() で計算した近似固有値と並べて見る
% 遅くなるので、あまりループは使いたくないが
%   d=zeros(n,1);
%   for j=1:n
%     d(j)=a(j,j)
%   end
% で対角成分を収めたベクトル d が得られる。
% e と d を並べて表示するには、
%   [e d]
% とするのが簡単。比較するためには、両方とも大きさの順に並べておくのがよい。
% それには sort() を使えばよい。
n=4;a=rand(n,n);a=(a+a')/2
e=sort(eig(a));
for i=1:100
  [q r]=qr(a);
  a=r*q
  d=zeros(n,1);
  for j=1:n
    d(j)=a(j,j);
  end
  d=sort(d);
  format long
```

```
  [e d]
  norm(d-e)
  format short
end

% 収束の様子を対角線の下側の成分が小さくなることで確認してみよう。
% 上三角部分（対角線込み）を取り出す triu() を用いて
n=4;a=rand(n,n);a=(a+a')/2
e=sort(eig(a));
for i=1:100
  [q r]=qr(a);
  a=r*q
  norm(a-triu(a))
end
% とするのが考えられる。
% 元の行列が対称の場合は、非対称部分の大きさを見るのも良いだろう。
% 対角部分の切り出しは、a .* eye(n,n) で出来るので、d-a のノルムを見てみる。
n=4;a=rand(n,n);a=(a+a')/2
e=sort(eig(a));
for i=1:100
  [q r]=qr(a);
  a=r*q
  d=a .* eye(n,n);
  norm(d-a)
end

% 結果が大きいとき、それを読むために、less のようなページャーが呼び出
% されて、読み終わった後、画面から消えてしまうが、それをファイルに残し
% たい場合は、(END) が出ているときに、s と打ってから LOG ファイルの名前
% を入力する。

%
function ret = rand_hessenberg(n)
  ret = triu(rand(n,n))w + diag(rand(n-1,1),-1);
%
function QR = qr_iteration(a,n)
  QR = a;
  for i=1:n
    [q r]=qr(QR);
    QR=r*q;
  end
%
a=rand_hessenberg(10)
a=qr_iteration(a,1)
% 後は繰り返し、というのもいいかも
-----------------------------------------------------------------
function ret = rand_h(n)
  ret = triu(rand(n,n)) + diag(rand(n-1,1),-1);

function ret = rand_t(n)
  ret=diag(rand(n,1),0)+diag(rand(n-1,1),1)+diag(rand(n-1,1),-1);

function ret = rand_st(n)
  u=rand(n-1,1);
  ret=diag(rand(n,1),0)+diag(u,1)+diag(u,-1);

function ret = rand_s(n)
  a=rand(n,n);
```

```
  ret = (a+a')/2;

function QR = qr_iteration(a,n)
  QR = a;
  for i=1:n
    [q r]=qr(QR);
    QR=r*q;
  end

function n = norm_l(a)
  n = norm(a-triu(a));
```