第11章　行列の固有値・固有ベクトル計算

> 現在実用になっている固有値の計算法はどれも相当に大げさなもので，下記のようないくつかの “部品”の適当な組み合わせからできている。ただ，どの部品にも，また組み合わせ方にも，いろいろと数値計算上注意を払わなければならない細かな点が多く，“数値計算の入門書”を読んで “原理”を理解して “小さな演習問題”をやったくらいでは，“本当に良いプログラム”を書くのは難しいのではないかと思われる。
>
> 伊里正夫・藤野和建「数値計算の常識」(共立出版)

正方行列の固有値・固有ベクトルを求める問題は，様々な用途に利用される。後述する常微分方程式の境界値問題も離散化する事でこの問題に帰着されるし，近年ではサーチエンジンにおいてもその計算が利用されている。しかし，連立一次方程式と異なり，5 次以上の行列の固有値は，有限回の代数的操作では真値を得ることは不可能である。従って，常に極限操作を有限回の計算で打ち切ることを念頭においてアルゴリズムを実行しなくてはならない。特に大規模な行列に対しては計算時間を短縮するために Reduction と呼ばれる操作を行うなど様々な工夫を行う必要がある。本章ではそのごく初歩を簡単に触れるに留める。

11.1　行列の固有値・固有ベクトル

定義 11.1.1 (正方行列の固有値・固有ベクトル)
$A \in M_n(\mathbb{C})$ に対し，

$$A\mathbf{v} = \lambda\mathbf{v} \quad (\mathbf{v} \neq 0) \tag{11.1}$$

を満足する $\lambda \in \mathbb{C}$ を行列 A の固有値 (eigenvalue)，$\mathbf{v} \in \mathbb{C}^n$ を行列 A の固有値 λ に対応する固有ベクトル (eigenvector) と呼ぶ。

本書では行列は全て実正方行列 $M_n(\mathbb{R})$ に限定する。また特定の行列 A の固有値であることを明示する必要がある場合は，$\lambda(A)$ と書くことにする。

定理 11.1.2
$A \in M_n(\mathbb{R})$ の固有値は，n 次実係数代数方程式 $|A - \lambda I| = 0$ の根である。この方程式を固有方程式，左辺を固有多項式と呼ぶ。従って，A の固有値は重複も含めて $\mathbb{C}$ 内に n 個存在する。

よって，5 次以上の正方行列の固有値を有限回の演算で求めることは一般には出来ないことになる。従って無限回反復を行うか，途中で打ち切って近似解を求めることで満足するほかない。

本章では A の n 個の固有値を，重複も含めて，絶対値の昇順に

$$|\lambda_1(A)| \geq |\lambda_2(A)| \geq \cdots \geq |\lambda_n(A)|$$

と書くことにする。

特に $A \in M_n(\mathbb{R})$ が対称行列，即ち $A^T = A$ であれば次の性質を持つ。

定理 11.1.3
実対称行列 A の固有値 $\lambda(A)$ は全て実数である。

従って，実対称行列においては複素数の固有値，固有ベクトルへの対策は必要ない。全て実数の範囲で求めることが出来るため，実対称行列に特化したアルゴリズム (Jacobi 法等) もある。逆に，複素数の固有値を求める際には，複素数演算を使用するか，特別な対策を講じる必要が出てくる。

問題 11.1.1
次の行列 $A, B \in M_2(\mathbb{R})$ の全ての固有値と固有ベクトルを求めよ。

$$A = \begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & -2 \\ 3 & 4 \end{bmatrix}$$

11.2　固有値・固有ベクトル計算の分類

固有値・固有ベクトルの計算の基盤となるものとして，相似変換がある。

定理 11.2.1 (相似変換)
$P \in M_n(\mathbb{R})$ が正則であれば，行列 A の固有方程式 $|A - \lambda I| = 0$ は

$$|P^{-1}AP - \lambda I| = |P^{-1}||A - \lambda I||P| = 0$$

より不変である。即ち，相似変換 $P^{-1}AP$ の固有値は A の固有値と同じである。但し固有ベクトルは $P^{-1}\mathbf{v}$ に変化する。

先に述べたように，5 次以上の行列の固有値を求めるには無限回の反復が必要となるが，それを「固有方程式を解く」という観点から行うのか，それとも「行列演算により対角行列もしくは (上) 三角行列へ収束させる」という観点から行うのかで，アルゴリズムを分類できる。また，固有値と固有ベクトルを同時に求める方法 (べき乗法・逆べき乗法，対称行列に対する Jacobi 法) なのか，固有値のみを求める方法 (QR 法，Bisection 法) なのかという観点でも分類が可能である。更に計算量を減らす目的で行列の Reduction(Householder 法, Lanczos 法) が行われることも多い。後者の場合，行列 A の固有値が判明すれば，

$$(A - \lambda I)\mathbf{v} = 0 \tag{11.2}$$

を解くことにより，固有値 λ に属する固有ベクトル $\mathbf{v}$ を求めることが出来る。但し，$\mathrm{rank}(A - \lambda I) < n$ なので，未知数を最低 1 つ，あらかじめ設定しておいてから解く必要が出てくる。

それに対して，Leverrier-Faddeev 法 (→ 演習問題 5) のように固有多項式の係数を明示的に導出し，代数方程式の解法 (第 13 章の演習問題参照) の解法を使用する方法は，数値解に混入する丸め誤差が大きくなるという理由で現在では使用されないのが普通である。逆に，代数方程式をコンパニオン行列の固有値問題として解く方法が推奨されつつある。

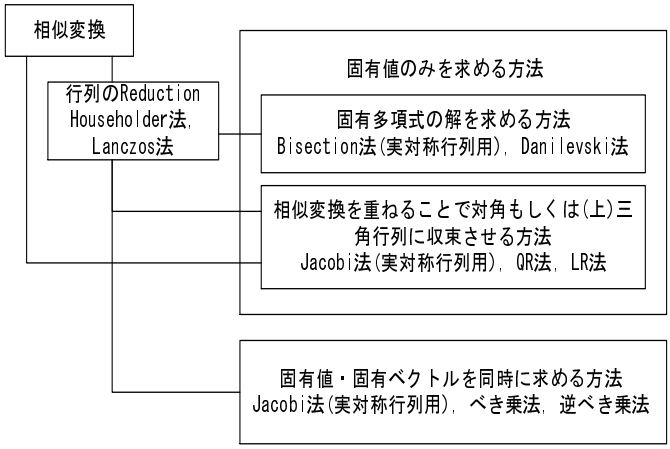

図 11.1: 固有値・固有ベクトル計算法の分類

11.3 べき乗法と逆べき乗法

べき乗法は最も単純な，絶対値最大固有値 $\lambda_1 = \lambda_1(A)$ とそれに属する固有ベクトル $\mathbf{v}_1$ を同時に求める方法である。もし全ての固有値が相異なる ($i < j$ の時，$\lambda_i \neq \lambda_j$, かつ，$|\lambda_i| > |\lambda_j|$) ならば，各固有値 $\lambda_i = \lambda_i(A)$ に属する固有ベクトル $\mathbf{v}_i$ は n 次元線型空間の基底となるため，任意のベクトル $\mathbf{x}_0$ は

$$\mathbf{x}_0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n$$

と表現できる。従って $\mathbf{x}_k := A^k\mathbf{x}_0$ とすれば

$$\mathbf{x}_k = (\lambda_1)^k \left\{ c_1\mathbf{v}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k \mathbf{v}_2 + \cdots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k \mathbf{v}_n \right\}$$

であるから，

$$\mathbf{x}_k = (\lambda_1)^k c_1\mathbf{v}_1 + O\left(\left(\frac{|\lambda_2|}{|\lambda_1|}\right)^k\right)$$

となり，固有ベクトル $\mathbf{v}_1$ へ収束する。さすれば固有値 λ_1 は Reiley 商

$$\lambda_1 \approx \frac{(A\mathbf{x}_{k+1}, \mathbf{x}_k)}{(\mathbf{x}_k, \mathbf{x}_k)}$$

を計算することで得られる。実際には overflow を防ぐため，反復一回ごとに $\|\mathbf{x}_k\| = 1$ となるように正規化する。

アルゴリズム 19 (べき乗法)

1. 初期ベクトル $\mathbf{x}_0$(ここで $\|\mathbf{x}_0\| = 1$) を決める。
2. for $k = 0, 1, 2, ...$
   (a) $\mathbf{y}_{k+1} := A\mathbf{x}_k$
   (b) $\gamma_{k+1} := (\mathbf{y}_{k+1}, \mathbf{x}_k)/(\mathbf{x}_k, \mathbf{x}_k)$
   (c) 収束判定
   (d) $\mathbf{x}_{k+1} := \mathbf{y}_{k+1}/\|\mathbf{y}_{k+1}\|$

このアルゴリズムに従うと，γ_k が $\lambda_1(A)$ へ，$\mathbf{x}_k$ はそれに属する固有ベクトルへと収束する。収束判定は固有値の近似値 γ_k，あるいは固有ベクトルの近似値 $\mathbf{x}_k$ を見て判断する。具体的には第 5 章の停止則の議論を参照されたい。

A が正則行列であるとき，A^{-1} に対してべき乗法を実行すると，その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ることになる。即ち，A の絶対値最小固有値とそれに属する固有ベクトルを得ることが出来る。

但し，前の章で示した通り，逆行列そのものを求めるのは計算量の観点から好ましいことではない。従って一度 A を LU 分解しておき，逆行列を乗する部分で後退代入のみ行うように工夫する。

アルゴリズム 20 (逆べき乗法)

1. 初期ベクトル $\mathbf{x}_0$(ここで $\|\mathbf{x}_0\| = 1$) を決める。
2. A を LU 分解し，$L_A U_A = A$ とする。
3. for $k = 0, 1, 2, ...$
   (a) $\mathbf{y}_{k+1} := U_A^{-1} L_A^{-1} \mathbf{x}_k$ とする。即ち $A\mathbf{y}_{k+1} = \mathbf{x}_k$ を $\mathbf{y}_{k+1}$ について解く。
   (b) $\delta_{k+1} := (\mathbf{y}_{k+1}, \mathbf{x}_k)/(\mathbf{x}_k, \mathbf{x}_k)$
   (c) 収束判定
   (d) $\mathbf{x}_{k+1} := \mathbf{y}_{k+1}/\|\mathbf{y}_{k+1}\|$

べき乗法と同様，δ_k は $\lambda_n(A)$ へ，$\mathbf{x}_k$ はそれに属する固有ベクトルへと収束する。但し，途中で A を係数行列とする連立一次方程式を解く必要があるため，べき乗法より条件数 $\kappa(A)$ 倍精度が悪化する可能性がある。

11.3.1　数値例

べき乗法と逆べき乗法の数値例を以下に示す。

例題 11.3.1 (べき乗法と逆べき乗法の数値例)
実対称行列 A を

$$A = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 4 & 3 & 2 & 1 \\ 3 & 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

とする。この時，IEEE754 倍精度で計算すると，べき乗法の場合以下のような結果を得る。
$\lambda_1(A)$ の近似値が

```
Maximum Eigenvalue:   1.23435375196795842e+01
```

になっている時の固有ベクトルの近似値 $\mathbf{x}$，及び $A\mathbf{x}$ の各要素の $\mathbf{x}$ との比をそれぞれ出力すると

```
i       eigenvector[i]        A * eivenvector[i] / eigenvector[i]
0   2.23606797749978981e+00   1.23435375196795842e+01
1   2.05491504837138317e+00   1.23435375196779056e+01
2   1.70728512307438196e+00   1.23435375196750883e+01
3   1.22134111072129303e+00   1.23435375196720223e+01
4   6.36451305172487269e-01   1.23435375196696810e+01
```

となる。
同様に，逆べき乗法の場合以下のような結果を得る。

```
Minimum Eigenvalue:   2.71554129623403084e-01
    i       eigenvector[i]        A * eivenvector[i] / eigenvector[i]
    0   1.16523414829594052e+00   2.71554128916587478e-01
    1  -3.12574884108380457e+00   2.71554129050644799e-01
    2   4.09386037170360861e+00   2.71554129276193768e-01
    3  -3.76220016281536740e+00   2.71554129521628440e-01
    4   2.23606797749979025e+00   2.71554129709021375e-01
```

問題 11.3.1
アルゴリズム 19 及び 20 を，実対称行列 $A = A^T$ に適用すると，固有値の収束のスピードと，固有ベクトルの収束のスピードが食い違うことになる。その理由を説明せよ。(Hint: 実対称行列の固有ベクトルは互いに直交することを利用せよ。)

11.4 LR 分解法

実際には LU 分解法というべきものである[1]。

[1]固有値問題の時は U(upper triangular matrix) を R と呼ぶことが多いようである。Wilkinson の命名をそのまま受け継いだ結果のようである。

アルゴリズム 21 (LR 分解法)

1. $A_0 := A$ とする。
2. for $k = 0, 1, 2, ...$
   (a) $L_k R_k = A_k$ と LU 分解する。
   (b) $A_{k+1} := R_k L_k$

要は，LU 分解した後，それを逆に乗じて次の行列を作り，それを繰り返すというアルゴリズムである。その結果,

$$A_i \longrightarrow \begin{bmatrix} \lambda_1 & & & * \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

のように，対角成分に A の固有値が並んだ上三角行列へ収束する。以下，それを先の例題の対称行列に LR 分解法を適用した結果の数値例で示す。

```
Iterative Times:     0
 5.000e+00  4.000e+00  3.000e+00  2.000e+00  1.000e+00
 4.000e+00  4.000e+00  3.000e+00  2.000e+00  1.000e+00
 3.000e+00  3.000e+00  3.000e+00  2.000e+00  1.000e+00
 2.000e+00  2.000e+00  2.000e+00  2.000e+00  1.000e+00
 1.000e+00  1.000e+00  1.000e+00  1.000e+00  1.000e+00

Iterative Times:    10
 1.234e+01  1.788e+01  1.024e+01  3.536e+00  1.000e+00
 2.780e-09  1.448e+00  8.299e-01  2.864e-01  8.101e-02
 9.665e-13  5.034e-04  5.733e-01  1.979e-01  5.597e-02
 1.933e-14  1.007e-05  1.147e-02  3.491e-01  9.874e-02
 5.817e-16  3.030e-07  3.451e-04  1.050e-02  2.858e-01

Iterative Times:    20
 1.234e+01  1.788e+01  1.042e+01  3.670e+00  1.000e+00
 1.378e-18  1.449e+00  8.444e-01  2.973e-01  8.101e-02
 5.200e-26  5.466e-08  5.829e-01  2.052e-01  5.592e-02
 6.503e-30  6.835e-12  7.289e-05  3.521e-01  9.595e-02
 1.835e-32  1.929e-14  2.057e-07  9.939e-04  2.727e-01

Iterative Times:    50
 1.234e+01  1.788e+01  1.042e+01  3.683e+00  1.000e+00
 1.681e-46  1.449e+00  8.445e-01  2.983e-01  8.101e-02
```

```
8.751e-66  7.544e-20  5.830e-01  2.059e-01  5.592e-02
3.219e-76  2.775e-30  2.145e-11  3.533e-01  9.593e-02
3.488e-82  3.007e-36  2.324e-17  3.827e-07  2.716e-01
```

その結果，対角成分には

```
i        eigenvalues
0   1.2343537519677058e+01
1   1.4486905679664057e+00
2   5.8296449828208962e-01
3   3.5325293743569902e-01
4   2.7155447480851137e-01
```

のように固有値の近似値が並ぶことになる。

問題 11.4.1
LR 分解法は次に示す QR 分解法よりも不安定であると言われている。そのように考えられている理由を述べよ。

11.5 QR 分解法

行列 A が正則であれば，その列ベクトル $A = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_n]$ は n 個の一次独立なベクトルの組と見なせる。

複数の一次独立なベクトルの組を，正規直交基底に変換するアルゴリズムとして，Gram-Schmidt の直交化法がある。$\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n$ へ適用し，正規直交基底 $\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_n$ を作るアルゴリズムは次のようになる。

アルゴリズム 22 (Gram-Schmidt の直交化法)

1. for $i = 1, 2, ..., n$
   (a) $\mathbf{u}_i := \mathbf{a}_i - \sum_{j=1}^{i-1}(\mathbf{a}_i, \mathbf{q}_j)\mathbf{q}_j$
   (b) $\mathbf{q}_i := \mathbf{u}_i / \|\mathbf{u}_i\|_2$

このアルゴリズムを行列の形で表わすと，

$$[\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_n] = [\mathbf{q}_1\ \mathbf{q}_2\ \cdots\ \mathbf{q}_n] \begin{bmatrix} \|\mathbf{u}_1\|_2 & (\mathbf{a}_2, \mathbf{q}_1) & (\mathbf{a}_3, \mathbf{q}_1) & \cdots & (\mathbf{a}_n, \mathbf{q}_1) \\ & \|\mathbf{u}_2\|_2 & (\mathbf{a}_3, \mathbf{q}_2) & \cdots & (\mathbf{a}_n, \mathbf{q}_2) \\ & & \ddots & \ddots & \vdots \\ & & & \|\mathbf{u}_{n-1}\|_2 & (\mathbf{a}_n, \mathbf{q}_{n-1}) \\ & & & & \|\mathbf{u}_n\|_2 \end{bmatrix}$$

となる。これを

$$A = QR$$

と書くと，Q は直交行列 $QQ^T = I$ に，R は上三角行列になる。これを行列 A の QR 分解と呼ぶ。先の LR 分解法のアルゴリズムにこの QR 分解を用いたアルゴリズムを，QR 分解法と呼ぶ。

アルゴリズム 23 (QR 分解法)

1. $A_0 := A$ とする。
2. for $k = 0, 1, 2, ...$
   (a) $Q_k R_k = A_k$ と A_k を QR 分解する。
   (b) $A_{k+1} := R_k Q_k$

この結果も LR 分解法と同様，対角成分に固有値が並んだ上三角行列へ収束する。以下，先の例題の対称行列に QR 分解法を適用した数値例を示す。

```
Iterative Times:     0
 5.000e+00  4.000e+00  3.000e+00  2.000e+00  1.000e+00
 4.000e+00  4.000e+00  3.000e+00  2.000e+00  1.000e+00
 3.000e+00  3.000e+00  3.000e+00  2.000e+00  1.000e+00
 2.000e+00  2.000e+00  2.000e+00  2.000e+00  1.000e+00
 1.000e+00  1.000e+00  1.000e+00  1.000e+00  1.000e+00

Iterative Times:    10
 1.234e+01  4.965e-09  7.330e-13  6.536e-15  2.138e-14
 4.965e-09  1.449e+00  2.148e-04  1.426e-06  3.953e-08
 7.363e-13  2.148e-04  5.829e-01  3.868e-03  1.073e-04
 4.885e-15  1.426e-06  3.868e-03  3.521e-01  9.765e-03
 1.355e-16  3.953e-08  1.073e-04  9.765e-03  2.727e-01

Iterative Times:    20
 1.234e+01 -8.636e-16 -3.198e-15  4.067e-15  2.092e-14
 2.462e-18  1.449e+00  2.392e-08  1.061e-12  7.289e-15
 4.065e-26  2.392e-08  5.830e-01  2.572e-05  5.197e-08
 1.794e-30  1.055e-12  2.572e-05  3.532e-01  7.137e-04
 3.624e-33  2.133e-15  5.197e-08  7.137e-04  2.716e-01

Iterative Times:    40
 1.234e+01 -8.661e-16 -3.198e-15  4.249e-15  2.089e-14
 6.053e-37  1.449e+00  1.125e-15  6.065e-15  5.103e-15
 1.239e-52  2.965e-16  5.830e-01  1.146e-09  1.177e-14
 2.436e-61  5.830e-25  1.146e-09  3.533e-01  3.707e-06
 2.556e-66  6.117e-30  1.203e-14  3.707e-06  2.716e-01

Iterative Times:    50
 1.234e+01 -8.661e-16 -3.198e-15  4.250e-15  2.089e-14
 3.001e-46  1.449e+00  8.289e-16  6.065e-15  5.103e-15
```

```
6.841e-66  3.302e-20  5.830e-01  7.651e-12 -2.530e-16
8.976e-77  4.333e-31  7.650e-12  3.533e-01  2.671e-07
6.787e-83  3.276e-37  5.784e-18  2.671e-07  2.716e-01
```

その結果，対角成分には

```
i        eigenvalues
0   1.2343537519677058 3e+01
1   1.4486905697966419 0e+00
2   5.8296449829374008 5e-01
3   3.5325328289322233 1e-01
4   2.7155412933933720 2e-01
```

のように，LR 分解法と同様に固有値の近似解が並ぶ。

現在，固有値計算に適用されるアルゴリズムは QR 分解法をベースにしたものが多い。実際には，後で述べる行列の Reduction を行った上で，原点移動と呼ばれる手法を用いて収束を早めつつ，逐次行列のサイズを減らしていく，という工夫がなされている。

問題 11.5.1
アルゴリズム 22 によって生成される n 個ベクトルの組, $\mathbf{q}_1$, $\mathbf{q}_2$, ...,$\mathbf{q}_n$ が正規直交基底をなすことを示せ。

11.6 行列の Reduction

固有多項式を計算する方法をとる場合には，行列を必ずその値を簡単に計算できるように変形しておく必要がある。また，行列演算を繰り返し行う手法でも，0 の多い行列であればそれだけ計算量を減らすことが可能になる。このように，元の行列を相似変換によって 0 成分の多い行列に変換することを行列の Reduction と呼ぶ。

現在主として用いられているものとしては，次の Householder 法がある。

アルゴリズム 24 (Householder 法)

1. $A_0 := A$ とする。
2. for $i = 1, 2, ..., n-1$
   (a) $s^2 := \sum_{j=i+1}^{n} a_{ji}^2$
   (b) $s := \mathrm{sign}(a_{i+1,i}) \sqrt{s^2}$
   (c) $c := 1/(s^2 + a_{i+1,i}s)$
   (d) $\mathbf{w}_i := [0 \ ... \ 0 \ \ a_{i+1,i} + s \ \ a_{i+2,i} \ ... \ a_{ni}]^T$
   (e) $P_i := I - c\mathbf{w}_i\mathbf{w}_i^T$
   (f) $\mathbf{q}_i := cA\mathbf{w}_i - \frac{c}{2}\mathbf{w}_i(cA\mathbf{w}_i)^T\mathbf{w}_i$
   (g) $A_{i+1} := P_iA_iP_i = A_i - \mathbf{q}_i\mathbf{w}_i^T - \mathbf{w}_i\mathbf{q}_i^T$

このアルゴリズムにより，一般の実正方行列 A は，直交変換行列 $P = P_1 P_2 \cdots P_{n-1}$ によって

$$P^{-1}AP = \begin{bmatrix} \alpha_1 & & & * \\ \beta_1 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_n \end{bmatrix} \tag{11.3}$$

という形になる。これを Hessenberg 行列と呼ぶ。

A が対称行列であれば，変換後の行列も対称性を保つため

$$P^{-1}AP = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{bmatrix} \tag{11.4}$$

という対称な 3 重対角行列となる。

先の例題に Householder 法を適用すると，変換行列 P は

```
 1.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00
 0.000e+00 -7.303e-01  6.293e-01 -2.615e-01  4.828e-02
 0.000e+00 -5.477e-01 -3.146e-01  7.191e-01 -2.897e-01
 0.000e+00 -3.651e-01 -5.843e-01 -2.615e-01  6.759e-01
 0.000e+00 -1.826e-01 -4.045e-01 -5.883e-01 -6.759e-01
```

となり，変換後の 3 重対角行列 $P^{-1}AP$ は

```
 5.000e+00 -5.477e+00  0.000e+00  0.000e+00  0.000e+00
-5.477e+00  8.200e+00  8.124e-01  1.550e-16 -2.487e-17
 0.000e+00  8.124e-01  1.022e+00  1.910e-01  0.000e+00
 0.000e+00  0.000e+00  1.910e-01  4.701e-01  5.681e-02
 0.000e+00  0.000e+00  0.000e+00  5.681e-02  3.077e-01
```

となる。

対称行列用[2]の Reduction アルゴリズムとしては，Lanczos 法がある。

アルゴリズム 25 (実対称行列用 Lanczos 法)

1. $\|\mathbf{x}_1\|_2 = 1$ となるベクトル $\mathbf{x}_1$ を定める。
2. for $i = 1, 2, ..., n$
   (a) $\alpha_i := (\mathbf{x}_i, A\mathbf{x}_i)$
   (b) $\mathbf{y}_{i+1} := A\mathbf{x}_i - \beta_{i-1}\mathbf{x}_{i-1} - \alpha_i\mathbf{x}_i$ (ここで$\beta_0 = 0$)
   (c) $\beta_i := \|\mathbf{y}_{i+1}\|_2$

[2]非対称行列でも非対称の 3 重対角行列に変換する Lanczos 法のバリエーションが存在する。

(d) $\mathbf{x}_{i+1} := (1/\beta_i)\mathbf{y}_{i+1}$

先の例題に Lanczos 法を適用すると，$\mathbf{x}_1 = [1\ 0\ ...\ 0]^T$ とした時の変換行列 P は

```
 1.000e+00  0.000e+00  0.000e+00  4.011e-14  8.489e-12
 0.000e+00  7.303e-01 -6.293e-01  2.615e-01 -4.828e-02
 0.000e+00  5.477e-01  3.146e-01 -7.191e-01  2.897e-01
 0.000e+00  3.651e-01  5.843e-01  2.615e-01 -6.759e-01
 0.000e+00  1.826e-01  4.045e-01  5.883e-01  6.759e-01
```

となり，変換後の 3 重対角行列 $P^{-1}AP$ は

```
 5.000e+00  5.477e+00  0.000e+00  0.000e+00  0.000e+00
 5.477e+00  8.200e+00  8.124e-01  0.000e+00  0.000e+00
 0.000e+00  8.124e-01  1.022e+00  1.910e-01  0.000e+00
 0.000e+00  0.000e+00  1.910e-01  4.701e-01  5.681e-02
 0.000e+00  0.000e+00  0.000e+00  5.681e-02  3.077e-01
```

となる。

丸め誤差に対して頑健な Householder 法に比べ，Lanczos 法はその影響を受けやすいことが知られている。十分な計算精度が確保されていないと，変換後の 3 重対角行列の固有値を求めると，真の固有値にごく近い偽の固有値が現れてしまうことが報告されている。

演習問題

1. $A \in M_3(\mathbb{R})$ を

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

とする。次の問いに答えよ。

(a) A の絶対値最大固有値 λ_1 と，それに対応する固有ベクトル $\mathbf{v}_1$ を求めよ。但し λ_1 は 10 進 2 桁以上の精度を持つようにせよ。

(b) A の LU 分解を求めよ。

(c) A の絶対値最小固有値 λ_3 と，それに対応する固有ベクトル $\mathbf{v}_3$ を求めよ。但し λ_3 は 10 進 2 桁以上の精度を持つようにせよ。

2. 次の行列のカテゴリを述べ，全ての固有値と，それに対応する固有ベクトルを求めよ。

$$A_1 = \begin{bmatrix} 1 & -2 \\ -2 & 3 \end{bmatrix},\ A_2 = \begin{bmatrix} 1 & -2 \\ 0 & 3 \end{bmatrix},\ A_3 = \begin{bmatrix} 1 & 0 \\ -2 & 3 \end{bmatrix}$$

3. べき乗法及び逆べき乗法が収束するためには，A が対角化可能であることが前提となっている。では，A の Jordan 標準形に 2 次以上の Jordan ブロックが含まれている場合，数値解はどのような状態になるか？[3]

4. (11.4) 式の対称 3 重対角行列を H とする時，次の問いに答えよ。

   (a) この固有多項式 $|H - \lambda I| = 0$ の左辺の値は，まず $p_0(\lambda) = 1$, $p_1(\lambda) := \alpha_1 - \lambda$ として，

   $$p_i(\lambda) := (\alpha_i - \lambda)p_{i-1}(\lambda) - \beta_{i-1}^2 p_{i-2}(\lambda)$$

   という漸化式を逐次計算し，$p_n(\lambda) = |H - \lambda I|$ となることを確認せよ。

   (b) Hessenberg 行列に対しても同様の漸化式でその固有多項式の値を計算することが可能である。どのように計算すればよいか考えよ。

5. 行列 $A = [a_{ij}] \in M_n(\mathbb{R})$ の固有多項式と同一視できる monic な多項式 $q_n(x) = x^n + \sum_{i=0}^{n-1} c_i x^i$ の係数 $c_n, c_{n-1}, ..., c_0$ は次の Leverrier-Faddeev 法 [21, 10] によって得ることができる。

$$\begin{aligned}
N_1 &:= I \\
c_{n-1} &:= -\mathrm{trace}(AN_1) \\
N_2 &:= AN_1 + c_{n-1}I \\
c_{n-2} &:= -\frac{1}{2}\mathrm{trace}(AN_2) \\
N_3 &:= AN_2 + c_{n-2}I \\
c_{n-3} &:= -\frac{1}{3}\mathrm{trace}(AN_3) \\
&\vdots \\
c_1 &:= -\frac{1}{n-1}\mathrm{trace}(AN_{n-1}) \\
N_n &:= AN_{n-1} + c_1 I \\
c_0 &:= -\frac{1}{n}\mathrm{trace}(AN_n)
\end{aligned}$$

   ここで $\mathrm{trace}(A) = \sum_{i=1}^{n} a_{ii}$ である。

   これにより行列の固有値を代数方程式の解法で求め，その数値解を固有値問題の数値解と比較せよ。

[3] 2 次以上の Jordan ブロックが含まれる場合はべき乗法に限らず以下で述べるアルゴリズムでも固有値の精度が落ちることが報告されている。