数値計算法・数値解析

固有値問題：べき乗法

宮崎大学 工学部

第 9 回

固有値問題

n 次正方行列 A が与えられた時,

$$Ax = \lambda x$$

λ: 固有値
x : 固有ベクトル

を満たす n 次元ベクトル x, 実数 λ の組を求める問題

【使い途】

- 振動系・制御系システムの安定性解析
- 多次元データの次元圧縮
- ラグランジュの未定乗数法の解法

種々の問題が固有値問題に帰着でき，その固有値・固有ベクトルを求めることで元の問題が解ける！

固有値問題：手計算

固有値問題

$$Ax = \lambda x$$

1. 特性方程式

$$|A - \lambda I| = 0$$

   の解が固有値
2. 固有値が求まれば，固有ベクトルが求まる

例：手計算

$A = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}$ の固有値，固有ベクトルを求めよ．

固有値問題：べき乗法

最大の固有値とその固有ベクトルを求める

固有値 $\lambda_1 > \lambda_2 > \ldots > \lambda_n$,

対応する単位固有ベクトル $u_1, u_2, \ldots, u_n$ とする

【原理】n 次元ベクトル x について,

$$x_k = \frac{Ax_{k-1}}{\|Ax_{k-1}\|}$$

とすると,

$$\begin{aligned} \lim_{k\to\infty} x_k &= u_1 \\ \lim_{k\to\infty} x_k \cdot Ax_k &= \lambda_1 \end{aligned}$$

となる.

固有値問題：べき乗法【証明】

固有ベクトル u_i は一次独立なので，任意の n 次元ベクトル x_0 は，

$$x_0 = a_1 u_1 + ... + a_n u_n = \sum_j a_j u_j$$

と表せる．

$$\begin{aligned} x_k &= \frac{Ax_{k-1}}{\|Ax_{k-1}\|} = \frac{A^k x_0}{\|A^k x_0\|} = \frac{A^k \sum_j a_j u_j}{\|A \sum_j a_j u_j\|} = \frac{\sum_j a_j \lambda_j^k u_j}{\| \sum_j a_j \lambda_j^k u_j\|} \\ &= \frac{\sum_j a_j (\lambda_j/\lambda_1)^k u_j}{\| \sum_j a_j (\lambda_j/\lambda_1)^k u_j\|} \end{aligned}$$

$k \to \infty$ で $(\lambda_j/\lambda_1)^k = 0 (j = 2, ..., n)$ なので，

$$\begin{aligned} \lim_{k\to\infty} x_k &= \frac{a_1 u_1}{\|a_1 u_1\|} = u_1 \\ \lim_{k\to\infty} x_k \cdot Ax_k &= u_1 \cdot Au_1 = \lambda_1 u_1 \cdot u_1 = \lambda_1 \end{aligned}$$

固有値問題：べき乗法

入力： n 次正方行列 A
出力： 最大固有値 λ_1 とその固有ベクトル u_1

【手順】

1. 初期ベクトル $x = x_0$ を適当に定める
2. $y = Ax$ を計算
3. $l = x \cdot y$ を計算
4. $x = y/\|y\|$ (y を単位ベクトルに正規化)
5. 収束するまで 2~4 を繰り返す
6. 収束した際の
   l が最大固有値，x が最大固有ベクトル u_1

例題：べき乗法

$A = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}$ の固有値，固有ベクトルをべき乗法の 5 回反復で求めよ．ただし，初期ベクトル $x_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ とする．また，正規化は最後に行えばよい．

べき乗法：収束条件

λ の値が変化しなくなるまで続ける

即ち

$$|\lambda - \lambda_{old}| > \varepsilon|\lambda|$$

の間，続ける．

べき乗法のプログラム(1/4)

```
#define N     2
#define EPS   (1e-5)
#include <math.h>
int main(int argc, char *argv[])
{
  double  lambda;

  double  A[][N] = {{1, -1}, {-1, 2}};
  double  x[] = {1, 0};

  poweigen(A, x, &lambda);
  printf("***␣eigen␣1␣:␣%f␣|␣", lambda);
  print_vec(x);

  return 0;
}
```

べき乗法のプログラム(2/4)

```
int poweigen(double A[][N], double *x,
             double *eigen_value)
{ double      y[N];
  double      l = 0.0, old_l = 1.0;

  while(fabs(l - old_l) > EPS * fabs(l)){
    old_l = l;
    multi_mat_vec(y, A, x); // y = Ax
    l = inner_prod(x, y);   // l = x • y
    normalize_vec(y, x);    // x = y / |y|
  }
  *eigen_value = l;

  return 0;
}
```

べき乗法のプログラム (3/4)

```
int multi_mat_vec(double y[],
                  double A[][N], double x[])
{
  for(int i = 0; i < N; i++) {
    y[i] = 0.0;
    for(int j = 0; j < N; j++) {
      y[i] += A[i][j] * x[j];
  } }
}
double inner_prod(double x[], double y[])
{ double      prod = 0.0;
  for(int i = 0; i < N; i++) {
    prod += x[i] * y[i];
  }
  return prod;
}
```

べき乗法のプログラム (4/4)

```
int normalize_vec(double y[], double x[])
{
  double length = sqrt(inner_prod(y, y));
  for(int i = 0; i < N; i++) {
    x[i] = y[i] / length;
  }
  return 0;
}
int print_vec(double x[])
{
  for(int i = 0; i < N; i++) {
    printf("%6.3f␣", x[i]);
  }
  printf("\n");
  return 0;
}
```

複数の固有値・固有ベクトル

べき乗法：
最大固有値しか求められないの？

答え：YES

ただし，A が対称行列の場合，
２番目以降の固有値，固有ベクトルも求められる．

対称行列の固有値・固有ベクトル

$$\begin{cases} Au_1 &= \lambda_1 u_1 \\ \ldots & \\ Au_n &= \lambda_n u_n \end{cases} \Leftrightarrow \begin{aligned} AX &= X\Lambda \\ X &= (u_1 \ldots u_n) \\ \Lambda &= diag(\lambda_1, \ldots, \lambda_n) \end{aligned}$$

A が対称行列の時，固有ベクトルは互いに直交する．

$$\begin{aligned} \Leftrightarrow A &= X\Lambda X^T \\ &= \lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T + \ldots + \lambda_n u_n u_n^T \\ (A - \lambda_1 u_1 u_1^T) &= \lambda_2 u_2 u_2^T + \ldots + \lambda_n u_n u_n^T \end{aligned}$$

従って，$(A - \lambda_1 u_1 u_1^T)$ を計算して，べき乗法を適用すれば，$(A - \lambda_1 u_1 u_1^T)$ の最大固有値・固有ベクトル（即ち，A の２番目の固有値，固有ベクトル）が求まる．以下同様．