# 数値計算法・数値解析

## 固有値問題：Jacobi 法

宮崎大学 工学部

第 10 回

# 固有値問題：Jacobi 法

n 次実対称行列 $A$ が与えられた時，
固有値・固有ベクトルを全て求める．

- 固有値問題では，対称行列を扱うことが多い
- べき乗法でも求められるが，全て求める場合は Jacobi 法の方が効率的

# 実対称行列の性質

n 次実対称行列 $A$

- n 個の固有値・固有ベクトルを持つ
- 固有値は全て実数
- 異なる固有ベクトル同士は全て直交する
- 直交行列 $P$ を使って

$$P^T AP = diag(\lambda_1, ..., \lambda_n)$$

とできる．ここで $\lambda_1, ..., \lambda_n$ は $A$ の固有値．$P$ は対応する固有ベクトルを並べた行列（各列が固有ベクトル）になっている．

# Jacobi 法の原理

- 直交行列（回転行列）$R$ を使って，$A$ のゼロでない非対角要素 $a_{pq}$ をゼロにする．
- これを全ての非対角要素がゼロになるまで繰り返す．

$$A_{k+1} = R_k^T A_k R_k$$

として，$A_k$ が対角行列になった時，

$$R_k^T ... R_1^T A R_1 ... R_k = P^T A P$$

即ち
$lim_{k \to \infty} A_k$ の対角要素が固有値，
$P = R_1 ... R_k$ の各列が固有ベクトル

# 要素 $a_{pq}$ をゼロにする回転行列

行列 A

$a_{pq}$ を 0 にする回転行列 $R$

$$
\begin{pmatrix}
a_{00} & a_{01} & \cdots & \cdots & \cdots & \cdots \\
a_{10} & a_{11} & \cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & a_{pp} & \cdots & a_{pq} & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & a_{qp} & \cdots & a_{qq} & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots
\end{pmatrix}
\begin{pmatrix}
1 & 0 & & \cdots & & 0 \\
0 & 1 & & & & \\
. & & cos\theta & & -\sin\theta & \\
. & & & 1 & & \\
. & & sin\theta & & \cos\theta & \\
0 & & & & & 1
\end{pmatrix}
$$

$$
\begin{aligned}
R_{pp} = R_{qq} &= \cos\theta =: c \\
R_{qp} = -R_{pq} &= \sin\theta =: s \\
\text{その他の対角要素} &= 1 \\
\text{その他の非対角要素} &= 0
\end{aligned}
$$

# 要素 $a_{pq}$ をゼロにする回転行列

$$
\begin{aligned}
A' &= R^T A R \\
a'_{pp} &= a_{pp}c^2 + 2a_{pq}sc + a_{qq}s^2 \\
a'_{qq} &= a_{pp}s^2 - 2a_{pq}sc + a_{qq}c^2 \\
a'_{pq} &= a'_{qp} = a_{pq}(c^2 - s^2) - (a_{pp} - a_{qq})sc \\
a'_{pj} &= a'_{jp} = ca_{pj} + sa_{qj} \quad (j = 1..N) \\
a'_{qj} &= a'_{jq} = -sa_{pj} + ca_{qj} \ \ (j = 1..N)
\end{aligned}
$$

ここで，$a'_{pq} = 0$ となるように，$s, c$ を定める．

# 要素 $a_{pq}$ をゼロにする回転行列

$$a'_{pq} = a_{pq}(c^2 - s^2) + (a_{qq} - a_{pp})sc = 0$$

$$\begin{cases} a_{pq}\cos 2\theta - (a_{pp} - a_{qq})/2 \sin 2\theta = 0 \\ \cos^2 2\theta + \sin^2 2\theta = 1 \end{cases}$$

$$\begin{aligned} \alpha &:= a_{pq} \\ \beta &:= (a_{pp} - a_{qq})/2 \end{aligned}$$

と置くと

$$\sin 2\theta = \frac{\alpha}{\beta}\cos 2\theta$$

$$\cos^2 2\theta + \frac{\alpha^2}{\beta^2}\cos^2 2\theta = 1$$

$$\cos 2\theta = \frac{|\beta|}{\sqrt{\alpha^2 + \beta^2}}$$

# 要素 $a_{pq}$ をゼロにする回転行列

$$\cos 2\theta = \frac{|\beta|}{\sqrt{\alpha^2 + \beta^2}} := \gamma$$

$$\begin{cases} s^2 = \sin^2\theta = \frac{1-\cos 2\theta}{2} = \frac{1-\gamma}{2} \\ c^2 = \cos^2\theta = \frac{1+\cos 2\theta}{2} = \frac{1+\gamma}{2} \end{cases}$$

$$\begin{cases} s = \sqrt{\frac{1-\gamma}{2}} \cdot sign \\ c = \sqrt{\frac{1+\gamma}{2}} \end{cases}$$

$\alpha \cdot \beta > 0$ の 時 $sign = 1$

$\alpha \cdot \beta \leq 0$ の 時 $sign = -1$

# Jacobi 法の手順

1. $A$ のゼロでない非対角要素 $a_{pq}$ をゼロにするような直交行列（回転行列）$R$ を求める．
2. $A$ を $R^TAR$ で置き換える．
3. 1, 2 を全ての非対角要素がゼロになるまで繰り返す．

$$A_k = R_k^T A_{k-1} R_k$$

として，$A_k$ が対角行列になった時，

$$A_k = R_k^T ... R_1^T A R_1 ... R_k = P^T A P$$

$lim_{k \to \infty} A_k$ の対角要素が固有値，
$P = R_1 ... R_k$ の各列が固有ベクトル

# 要素 $a_{pq}$ をゼロにする回転行列 $R$

行列 A

$$\begin{pmatrix} a_{00} & a_{01} & \cdots & \cdots & \cdots & \cdots \\ a_{10} & a_{11} & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & a_{pp} & \cdots & a_{pq} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & a_{qp} & \cdots & a_{qq} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

$a_{pq}$ を 0 にする回転行列 $R$

$$\begin{pmatrix} 1 & 0 & & & \cdots & & 0 \\ 0 & 1 & & & & & \\ . & & c & & & -s & \\ . & & & & 1 & & \\ . & & s & & & c & \\ 0 & & & & & & 1 \end{pmatrix}$$

$$\begin{aligned} R_{pp} = R_{qq} &=: c \\ R_{qp} = -R_{pq} &=: s \\ \text{その他の対角要素} &= 1 \\ \text{その他の非対角要素} &= 0 \end{aligned}$$

# 回転行列 $R$ の計算

$$R = \begin{pmatrix} 1 & & & & & 0 \\ & 1 & & & & \\ & & c & & -s & \\ & & & 1 & & \\ & & s & & c & \\ 0 & & & & & 1 \end{pmatrix}$$

$$\begin{aligned} \alpha &:= a_{pq} \\ \beta &:= (a_{pp} - a_{qq})/2 \\ \gamma &:= \frac{|\beta|}{\sqrt{\alpha^2 + \beta^2}} \end{aligned}$$

$$\begin{aligned} s &= \sqrt{\frac{1-\gamma}{2}} \cdot sign \\ c &= \sqrt{\frac{1+\gamma}{2}} \end{aligned}$$

$\alpha \cdot \beta > 0$ の 時 $sign = 1$

$\alpha \cdot \beta \leq 0$ の 時 $sign = -1$

# 例題

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

# Jacobi 法のプログラム

```
#define N      3
int main(int argc, char *argv[])
{
  double A[][3] = {{4,2,1}, {2,1,2}, {1,2,8}};
  double eigen_vecs[N][N];

  jacobi_eigen(A, eigen_vecs);

  printf("***␣eigen␣value\n");
  print_array(A, N);
  printf("***␣eigen␣vecs\n");
  print_array(eigen_vecs, N);

  return 0;
}
```

# Jacobi 法のプログラム

```
#define EPS  (1e-6)
void jacobi_eigen(double A[][N],
                  double eigen_vecs[][N])
{ int            cont_flg = 1;
  set_identity_mat(eigen_vecs);
  while(cont_flg == 1) {
    cont_flg = 0;
    for(int p = 0; p < N; p++) {
      for(int q = p+1; q < N; q++) {
        if (fabs(A[p][q]) < EPS) continue;
        cont_flg = 1;
        jacobi_diag(A, p, q, eigen_vecs);
  } } }

  return;
}
```

# Jacobi 法のプログラム

```
#define sqr(x)  ((x)*(x))
void jacobi_diag(double A[][N], int p, int q,
                 double eigen_vecs[][N])
{ double  app = A[p][p];
  double  aqq = A[q][q];
  double  apq = A[p][q];

  double  alpha  = apq;
  double  beta = (app - aqq)/2.0;
  double  gamma=fabs(beta)/
                  sqrt(sqr(alpha)+sqr(beta));
  double  s = sqrt((1.0 - gamma)/2.0);
  double  c = sqrt((1.0 + gamma)/2.0);

  if (alpha * beta <= 0.0) s = -s;
  ...
```

# Jacobi 法のプログラム

```
...
for(int j = 0; j < N; j++) {
    double apj = A[p][j];
    A[p][j] =  c * apj + s * A[q][j];
    A[q][j] = -s * apj + c * A[q][j];
}
for(int j = 0; j < N; j++) {
    A[j][p] = A[p][j];
    A[j][q] = A[q][j];
}

A[p][p] = app*sqr(c) + 2*apq*s*c + aqq*sqr(s);
A[q][q] = app*sqr(s) - 2*apq*s*c + aqq*sqr(c);
A[p][q] = A[q][p] = 0.0;
...
```

# Jacobi 法のプログラム

```
  ...
  // update eigen vectors
  for(int i = 0; i < N; i++) {
    double eip = eigen_vecs[i][p];
    eigen_vecs[i][p]= c*eip+s*eigen_vecs[i][q];
    eigen_vecs[i][q]=-s*eip+c*eigen_vecs[i][q];
  }

  return;
}
```

# Jacobi 法のプログラム

```
void set_identity_mat(double mat[][N])
{
  for(int i = 0; i < N; i++) {
    for(int j = 0; j < N; j++) {
        mat[i][j] = 0.0;
    }
    mat[i][i] = 1.0;
  }
  return;
}
```