# 情報数学C
*Mathematics for Informatics C*

第10回 行列の固有値計算
(べき乗法,ハウスホルダー変換,QR法)

情報メディア創成学類
藤澤誠



## 今回の講義内容

- **今日の問題**
- べき乗法
- ハウスホルダー変換
- QR法



## 今回の講義で解く問題

$$A\boldsymbol{v} = \lambda\boldsymbol{v}$$

を満たす$\lambda$と$\boldsymbol{v}$
(ただし$\boldsymbol{v} \neq \boldsymbol{0}$)



## 今回の講義で解く問題

**固有値(eigen value)と固有ベクトル(eigen vector)を求める問題(固有値問題)**

$n \times n$の正方行列$A$について

$$A\boldsymbol{v} = \lambda\boldsymbol{v} \qquad (\boldsymbol{v} \neq \boldsymbol{0})$$

を満たすスカラー値$\lambda$とベクトル$\boldsymbol{v}$を求めよ.

$\lambda$：行列$A$の**固有値**
$\boldsymbol{v}$：行列$A$の**固有ベクトル**

$n \times n$の正方行列$A$は重複&値がゼロのものを含めて
**$n$個の固有値(とそれに対応する$n$個の固有ベクトル)**
を持つ



## 今回の講義で解く問題

固有値問題を**数学的に解いてみよう**

$n \times n$の正方行列$A$について：$A\boldsymbol{v} = \lambda\boldsymbol{v}$

単位行列を$I$とすると式を変形して,

$$(A - \lambda I)\boldsymbol{v} = \boldsymbol{0}$$

前提条件で$\boldsymbol{v}$がゼロベクトルでない($\boldsymbol{v} \neq \boldsymbol{0}$)としているが,
もし,行列$(A - \lambda I)$の逆行列が存在してしまうと,

$$(A - \lambda I)^{-1}(A - \lambda I)\boldsymbol{v} = \boldsymbol{0}$$
$$\Longrightarrow \boldsymbol{v} = \boldsymbol{0}$$

となり,$\boldsymbol{v}$がゼロベクトルになってしまう.よって$(A - \lambda I)$には逆行列が存在しない,つまり,**行列式が0になる.**



## 今回の講義で解く問題

固有値問題を**数学的に解いてみよう**

$$\det(A - \lambda I) = 0$$

この式の行列式を展開すると:

$$\det(A - \lambda I) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0$$

⇩ $n$次の多項式になる

$$\lambda^n + c_1\lambda^{n-1} + \cdots + c_{n-1}\lambda + c_n = 0$$

これを**特性方程式**と呼び,この**多項式を解くことで$n$個の固有値が得られ**,得られた$\lambda$を$A\boldsymbol{v} = \lambda\boldsymbol{v}$**に代入することで固有ベクトル$\boldsymbol{v}$が得られる.**

# 今回の講義で解く問題

例) 行列$A=\begin{pmatrix}1 & 4\\ 3 & 2\end{pmatrix}$の固有値と固有ベクトルを求めよ.

$$\det(A-\lambda I)=\begin{vmatrix}1-\lambda & 4\\ 3 & 2-\lambda\end{vmatrix}=(1-\lambda)(2-\lambda)-3\times 4$$

展開して整理

$$\lambda^2-3\lambda-10=0$$

⇩ 因数分解

$$(\lambda+2)(\lambda-5)=0$$

固有値$\lambda=-2,5$

$A\boldsymbol{v}=\lambda\boldsymbol{v}\Rightarrow(A-\lambda I)\boldsymbol{v}=0$より

$\lambda=-2$について

$$\begin{pmatrix}3 & 4\\ 3 & 4\end{pmatrix}\begin{pmatrix}v_1\\ v_2\end{pmatrix}=0$$

$3v_1+4v_2=0$ ⇩ $\boldsymbol{v}=\begin{pmatrix}-4\\ 3\end{pmatrix}$

$\lambda=5$について

$$\begin{pmatrix}-4 & 4\\ 3 & -3\end{pmatrix}\begin{pmatrix}v_1\\ v_2\end{pmatrix}=0$$

$-v_1+v_2=0$ ⇩ $\boldsymbol{v}=\begin{pmatrix}1\\ 1\end{pmatrix}$

⇒ 固有ベクトルを**定数倍したものも固有ベクトル**



# 今回の講義で解く問題

固有値/固有ベクトルって何?

$A\boldsymbol{v}=\lambda\boldsymbol{v}$という定義から考えると,
「ベクトル$\boldsymbol{v}$を行列$A$で線形変換したときに, 変換した結果のベクトルの**向きは変わらず**, **スカラー倍($\lambda$倍)しただけ**のベクトルになる」ときにベクトル$\boldsymbol{v}$を固有ベクトルという.

先ほどの例だと

$\lambda=5,\boldsymbol{v}=\begin{pmatrix}1\\ 1\end{pmatrix}$の場合

$$A\boldsymbol{v}=\begin{pmatrix}1 & 4\\ 3 & 2\end{pmatrix}\begin{pmatrix}1\\ 1\end{pmatrix}=\begin{pmatrix}5\\ 5\end{pmatrix}=5\boldsymbol{v}$$

⇒ 行列$A$を固有ベクトルにかけても**方向は変わっていない**



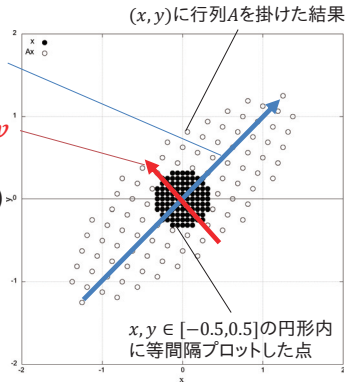
# 今回の講義で解く問題

$A=\begin{pmatrix}1 & 4\\ 3 & 2\end{pmatrix}$で実際に**線形変換**してみると…

$\lambda=5,\boldsymbol{v}=\begin{pmatrix}1\\ 1\end{pmatrix}$でのベクトル$\lambda\boldsymbol{v}$

$\lambda=-2,\boldsymbol{v}=\begin{pmatrix}-4\\ 3\end{pmatrix}$でのベクトル$\lambda\boldsymbol{v}$

固有ベクトルと固有値(の絶対値)は変換した点の**分布における長い方向と短い方向**, そして, それぞれの**長さ**を示している.



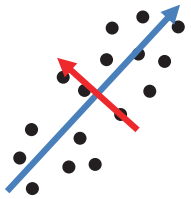



# 今回の講義で解く問題

固有値/固有ベクトルって何?

計測等で得られた点の分布を形成する**線形変換行列**$M$が求められれば,

- 行列$M$の最大/最小固有値の絶対値は分布の一番**長い**/**短い長さ**を示す
- 対応する固有ベクトルは**長い方向**/**短い方向**を示す



分布がどのようになっているのかを調べられる
⇒ **主成分分析**(PCA:Principal Componet Analysis)と呼ぶ
(行列$M$には共分散行列が使われることが多い)



# 今回の講義で扱う問題

固有値/固有ベクトルは**どんなところで使われる?**

**自然現象, 工学モデル**が行列の形で表されたときの**安定性解析や特徴解析**に使われる.

**統計**
得られた多次元データの固有値・固有ベクトルから主成分を調べるために用いられる(主成分分析:PCA). グラフ理論やマルコフ連鎖などでも固有値は使われる

**画像処理・CG・シミュレーション**
周囲の画素間の共分散行列の固有値を用いたテクスチャ特徴解析や固有ベクトルを用いた点群の局所座標設定(対称行列なら固有ベクトルが直交するので), 粒子法における楕円体カーネルの計算などなど

固有値/固有ベクトルで何かの問題を解くというよりは,
システムの**解析**や特性を調べての**高速化/効率化***に使われる.

*高次元のデータを固有値が高い方向のみ残して低次元化することで高速化する(次元削減)など



# 今回の講義で解く問題

**今回の講義資料における注意点**

- いつもはプログラムとの整合性のために, 0スタートの添え字を使っているが, 今回は数学的な数式変形が多いので**1スタートの添え字をほとんどのところで使っている**ことに注意

$$A=\begin{pmatrix}a_{11} & a_{12} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ a_{n1} & a_{n2} & \cdots & a_{nn}\end{pmatrix}$$

$n\times n$行列の表記の例
添え字(インデックス)が1から始まり$n$で終わる

# 固有値の数値計算法

**固有値を数値計算で求める方法**

- 対角化，直交ベクトルを上手く使うことで固有値を求める
  ⇒ **べき乗法**
- 特性方程式を2分法とかDKAで解けば良いのでは？
  ⇒ 行列が大きいと**計算時間的に厳しい**
  - **解きやすい形に行列を変形**してやる
    **⇒ ハウスホルダー変換**
  - ハウスホルダー変換を使って，行列を直交行列$Q$と上三角行列$R$に分解して固有値を求める
    **⇒ QR法**



# べき乗法

**べき乗法** は**対角化可能な行列**について**最大固有値とそれに対応する固有ベクトル**を求める方法
⇒ すべての固有値を求めるものではないけど，最大固有値だけがほしいときには効率的
⇒ まずは**行列の対角化の方法とその意味**を復習しよう！

**行列の対角化とは？**

$n \times n$の正方行列$A$に対して，正則な行列$P$を$P^{-1}AP$と前後から掛けたときに，$P^{-1}AP$**が対角行列になる**場合を**対角化**と呼ぶ

$$P^{-1}AP = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} = D$$



# べき乗法

対角化できると**何が良い**のか？

⇒ 行列の累乗計算が簡単になる
($n \times n$行列$A$の$m$乗($A^m$)の計算コストは$O(mn^3)$)

$$(P^{-1}AP)^m = P^{-1}AP\,P^{-1}AP \cdots P^{-1}AP$$

単位行列$I$になる

$$(P^{-1}AP)^m = P^{-1}A^mP$$

$$\Downarrow$$

$$A^m = PD^mP^{-1}$$

対角行列$D$の$m$乗は**対角要素をそれぞれ$m$乗すれば良いだけ**なので，計算コストは$O(mn)$，$P, P^{-1}$の計算コストを加えても$O(2n^3) + O(mn)$で$m$が大きいときほど効率的



# べき乗法

**対角化と固有値の関係**

$P^{-1}AP = D$を変形すると

$$AP = P\begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

行列$P$の各列成分を$\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n$とすると(つまり$P = (\boldsymbol{p}_1\ \boldsymbol{p}_2 \cdots \boldsymbol{p}_n)$)

$$A\boldsymbol{p}_i = \lambda_i \boldsymbol{p}_i$$

固有値の定義式$A\boldsymbol{v} = \lambda\boldsymbol{v}$と**全く同じ形**

⇒ $A$の固有ベクトル$\boldsymbol{v}_i$を縦ベクトルとして並べた行列を$P$とすると，$P^{-1}AP$は$A$の**固有値を対角成分に持つ対角行列**になる



# べき乗法

**対角化可能な行列**の条件は？

固有ベクトルを並べた行列$P$が正則である
⇒ 行列$A$の固有ベクトル$\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$が線形独立である*

- $n \times n$の行列$A$が**$n$個の重複しない固有値を持つ**場合は，異なる固有値の固有ベクトルはすべて線形独立なので
  **対角化可能**
  ⇒ $n$個の固有値を$\lambda_1, \lambda_2, \ldots, \lambda_n$ (絶対値が大きい順)とすると以下が成り立つ
  $$|\lambda_1| > |\lambda_2| > |\lambda_3| > \cdots > |\lambda_n|$$
- 行列$A$が重複する固有値を持つと**対角化不可能な場合がある**
  ⇒ これらから**べき乗法**のアルゴリズムを導いてみよう

*行列$P$が正則ならば，その列ベクトルは互いに線形独立という定理を使っている．



# べき乗法

**行列$A$が対角化可能**ならば，線形独立な固有ベクトルを使って，あるベクトル$\boldsymbol{x}^{(0)}$を任意の定数ベクトル$\boldsymbol{c} = (c_1, c_2, \ldots, c_n)^{\mathrm{T}}$を使って表せる

$$\boldsymbol{x}^{(0)} = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n$$
$$= P\boldsymbol{c}$$

($\boldsymbol{v}_i$が線形独立でないと$\boldsymbol{x}^{(0)}$になる係数列$\boldsymbol{c}$が唯一定まらない)

$\boldsymbol{x}^{(0)}$に行列$A$を掛けたものを$\boldsymbol{x}^{(1)}$とすると

$$\boldsymbol{x}^{(1)} = A\boldsymbol{x}^{(0)} = AP\boldsymbol{c} = PP^{-1}AP\boldsymbol{c} = P\begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}\boldsymbol{c}$$
$$= \lambda_1 c_1\boldsymbol{v}_1 + \lambda_2 c_2\boldsymbol{v}_2 + \cdots + \lambda_n c_n\boldsymbol{v}_n$$

ということは同様にすれば$\boldsymbol{x}^{(2)}$は

$$\boldsymbol{x}^{(2)} = \lambda_1^2 c_1\boldsymbol{v}_1 + \lambda_2^2 c_2\boldsymbol{v}_2 + \cdots + \lambda_n^2 c_n\boldsymbol{v}_n$$

# べき乗法

$k$回繰り返すと

$$\boldsymbol{x}^{(k)} = \lambda_1^k c_1 \boldsymbol{v}_1 + \lambda_2^k c_2 \boldsymbol{v}_2 + \cdots + \lambda_n^k c_n \boldsymbol{v}_n$$

$A$が対角化可能ならば$|\lambda_1| > |\lambda_2| > \cdots$であるので, $\lambda_1 \neq 0$ ($\lambda_1 = 0$だと**重複しない固有値が$n$個存在しない**ことになるので).
また, $c_1 \neq 0$とすると:

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

⇩ この部分に注目すると $|\lambda_1| > |\lambda_2|$より, $\lim_{k\to\infty}\left(\frac{\lambda_2}{\lambda_1}\right)^k = 0$

$$\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \lambda_1^k c_1 \boldsymbol{v}_1 \Longrightarrow \lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{v}_1$$

固有ベクトルを**定数倍したものも固有ベクトル**



# べき乗法

**べき乗法(power method)**

適当な初期値$\boldsymbol{x}^{(0)}$から

$$\boldsymbol{x}^{(k+1)} = A\boldsymbol{x}^{(k)}$$

により$\boldsymbol{x}$を更新していくと, 絶対値最大固有値$\lambda_1$の固有ベクトル$\boldsymbol{v}_1$が得られる

初期値はどう設定する?

- $c_1 \neq 0$ならなんでもよいが, $\lambda_1^k c_1$倍されることになるので$\lambda_1 > 1$だと$\boldsymbol{x}^{(k)}$が大きくなりすぎる可能性がある.
  ⇒ 固有ベクトルは定数倍OKなので $|\boldsymbol{x}^{(k)}| = 1$**となるように毎ステップ正規化**する
  ⇒ 当然**初期値も**$|\boldsymbol{x}^{(0)}| = 1$となるようにする(間違ってもすべて0とかにしない)
  ⇒ ちなみに$c_1 = 0$となるように初期値設定されると$\boldsymbol{v}_2$以降が求まることあり



# べき乗法

固有値$\lambda_1$は?

**レイリー(Rayleigh)商**というものを考えてやると**固有値$\lambda_1$も計算可能**

$$R(\boldsymbol{x}) = \frac{\boldsymbol{x}\cdot A\boldsymbol{x}}{\boldsymbol{x}\cdot\boldsymbol{x}} \Rightarrow R(\boldsymbol{v}_1) = \frac{\boldsymbol{v}_1 \cdot A\boldsymbol{v}_1}{\boldsymbol{v}_1\cdot\boldsymbol{v}_1} = \frac{\boldsymbol{v}_1\cdot\lambda_1\boldsymbol{v}_1}{\boldsymbol{v}_1\cdot\boldsymbol{v}_1} = \lambda_1$$

(固有値の定義$A\boldsymbol{v} = \lambda\boldsymbol{v}$より)

つまり, **毎反復**以下の式で**固有値が計算できる**

$$R(\boldsymbol{x}^{(k)}) = \frac{\boldsymbol{x}^{(k)}\cdot A\boldsymbol{x}^{(k)}}{\boldsymbol{x}^{(k)}\cdot\boldsymbol{x}^{(k)}} = \frac{\boldsymbol{x}^{(k)}\cdot\boldsymbol{x}^{(k+1)}}{\boldsymbol{x}^{(k)}\cdot\boldsymbol{x}^{(k)}} = \lambda^{(k)}$$

- 毎反復計算する必要はなさそう(最後の1回だけでよさそう)だけど, **収束判定**で$\lambda^{(k)}$が必要なので**毎回計算する**.
- $\boldsymbol{x}^{(k+1)}$は次ステップのために計算した値を使えば良い.
- $|\boldsymbol{x}^{(k)}| = 1$と**正規化されているなら実際は**$\lambda^{(k)} = \boldsymbol{x}^{(k)}\cdot\boldsymbol{x}^{(k+1)}$で計算できる



# べき乗法

収束判定はどうする?

固有値の定義$A\boldsymbol{v} = \lambda\boldsymbol{v}$から$|A\boldsymbol{v}_1 - \lambda_1\boldsymbol{v}_1| = 0$となればよい. 実際にはこの式の値が**設定した閾値$\varepsilon$より小さくなるまで繰り返す**とすると収束判定は:

$$|A\boldsymbol{x}^{(k)} - \lambda^{(k)}\boldsymbol{x}^{(k)}| < \varepsilon$$

計算コストを少しでも抑えるためにもう少し変形してみる.
平方根を使いたくないので両辺を2乗して**2乗距離で判定**するようにして, 更に$\boldsymbol{x}^{(k+1)} = A\boldsymbol{x}^{(k)}$**を代入**する.

$$|\boldsymbol{x}^{(k+1)} - \lambda^{(k)}\boldsymbol{x}^{(k)}|^2 < \varepsilon^2$$

⇩ 展開

$$|\boldsymbol{x}^{(k+1)}|^2 - 2\lambda^{(k)}\boldsymbol{x}^{(k+1)}\cdot\boldsymbol{x}^{(k)} + (\lambda^{(k)})^2|\boldsymbol{x}^{(k)}|^2$$

前ページの式から $\boldsymbol{x}^{(k+1)}\cdot\boldsymbol{x}^{(k)} = \lambda^{(k)}|\boldsymbol{x}^{(k)}|^2$

毎ステップ$|\boldsymbol{x}^{(k)}| = 1$となるように正規化している

⇩

$$|\boldsymbol{x}^{(k+1)}|^2 - (\lambda^{(k)})^2 < \varepsilon^2$$

**収束条件**



# べき乗法

べき乗法の計算手順

1. 初期値$\boldsymbol{x}^{(0)}$を設定
2. 以下の手順を収束するまで繰り返す($k = 0,1,2,\ldots$)
   a. $\boldsymbol{x}^{(k)}$を$|\boldsymbol{x}^{(k)}| = 1$となるように正規化 : $\boldsymbol{x}^{(k)} \leftarrow \boldsymbol{x}^{(k)}/|\boldsymbol{x}^{(k)}|$
   b. $\boldsymbol{x}^{(k+1)} = A\boldsymbol{x}^{(k)}$を計算
   c. $k$ステップ目の固有値$\lambda^{(k)}$を計算 : $\lambda^{(k)} = \boldsymbol{x}^{(k)}\cdot\boldsymbol{x}^{(k+1)}$
   d. $\left||\boldsymbol{x}^{(k+1)}|^2 - (\lambda^{(k)})^2\right| < \varepsilon^2$なら反復終了

実際のコードでは2-dでの$|\boldsymbol{x}^{(k+1)}|^2$を変数に格納しておいて次ステップに使う



# べき乗法

べき乗法のコード例

行列$A$が2次元配列に格納され, 初期値$x$ (正規化していない)が引数で与えられる

```
vector<double> x1(n); // v^(k+1)格納用
double l2 = dot(x, x, n), e;
int k;
for(k = 0; k < max_iter; ++k){
    // |x^(k)|=1となるように正規化
    double l = sqrt(l2); // |x^(k)|の計算
    x = mul_sv(1.0/l, x, n); // |x^(k)|で割る(1/|x|を掛ける)

    // x^(k+1) = A x^(k)の計算
    x1 = mul_mv(A, x, n);

    // 固有値λの計算(|x^(k)|=1で正規化されていることが前提)
    lambda = dot(x, x1, n);

    // 収束判定
    l2 = dot(x1, x1, n);
    if((e = fabs(l2-lambda*lambda)) < eps*eps) break;

    x = x1;
}
```

初期値の正規化用に$|\boldsymbol{x}^{(0)}|^2$を計算しておく

$\boldsymbol{x}^{(k)}$の正規化. mul_sv()はベクトルにスカラー値を掛ける関数

$\boldsymbol{x}^{(k)}$の更新:$\boldsymbol{x}^{(k+1)} = A\boldsymbol{x}^{(k)}$. mul_mv()は行列とベクトルの積を計算する関数

$\lambda^{(k)}$の計算. 正規化されている($|\boldsymbol{x}^{(k)}|^2 = 1$)ので$R(\boldsymbol{x}^{(k)})$の分子の計算のみで良いことに注意

収束判定. $|\boldsymbol{x}^{(k+1)}|^2$は次のステップの処理のために変数に格納しておこう. dot()はベクトル同士の内積を計算する関数

# べき乗法

べき乗法の実行結果例

$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 5 & 3 \end{pmatrix}$ ⇨ 固有値$\lambda = 5, 2, 0$

固有ベクトル$\boldsymbol{v} = \begin{pmatrix} 1 \\ 3 \\ 8 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}$

許容誤差$\varepsilon = 1 \times 10^{-6}$,初期値$(1, 1, 1)$

```
0 : lambda = 5.3333333,  v = 0.57735027, 0.57735027, 0.57735027
1 : lambda = 5.2830189,  v = 0.29138576, 0.38851434, 0.87415728
2 : lambda = 5.1333333,  v = 0.18257419, 0.36514837, 0.91287093
3 : lambda = 5.0555556,  v = 0.14213381, 0.35533453, 0.92386977
4 : lambda = 5.0225207,  v = 0.12649735, 0.35138154, 0.92764726
     ⋮（省略）
15 : lambda = 5.000001,  v = 0.11624807, 0.34874303, 0.92998102
16 : lambda = 5.0000004, v = 0.11624781, 0.34874296, 0.92998107
lambda_max = 5.0000004
v = 0.11624771, 0.34874293, 0.92998109
iter = 16, eps = 5.3974301e-07          17回の反復で処理終了
```

正解の最大固有値の固有ベクトルを正規化すると$\boldsymbol{v} = \begin{pmatrix} 0.11624764 \\ 0.34874292 \\ 0.92998111 \end{pmatrix}$なので,

最大固有値/固有ベクトルともに$10^{-6}$の精度まで求められている.



# 固有値の数値計算法

**固有値を数値計算で求める方法**

- 対角化, 直交ベクトルを上手く使うことで固有値を求める
  ⇒ **べき乗法**
- 特性方程式を2分法とかDKAで解けば良いのでは?
  ⇒ 行列が大きいと**計算時間的に厳しい**
  - **解きやすい形に行列を変形**してから求根問題として解く
    ⇒ **ハウスホルダー変換**
  - ハウスホルダー変換を使って, 行列を直交行列$Q$と上三角行列$R$に分解して固有値を求める
    ⇒ **QR法**



# 相似変換

行列$A$の対角化を思い出してみよう

$$P^{-1}AP = D = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

「対角行列の固有値=対角成分」である*ので,

**対角行列$D$の固有値は行列$A$と同じ**$(\lambda_1, \lambda_2, \ldots, \lambda_n)$

行列$A$を対角化すれば固有値は簡単に求まる?

⇒ 固有値を変えない対角化のために固有値・固有ベクトルが必要

⇒「**卵が先か鶏が先か**」という状況になってしまう**

対角化まで行かなくても**固有値が求めやすい形**に,
(固有値を変えないように)**行列$A$を変形**できないか?

*$\det(A - \lambda I) = 0$が$(\lambda_1 - \lambda)(\lambda_2 - \lambda)\cdots(\lambda_n - \lambda) = 0$となるので$\lambda = \lambda_1, \lambda_2, \ldots, \lambda_n$

**対称行列限定だけど対角化までしてしまうヤコビ法という方法もある



# 相似変換

**固有値**を変えない行列の変換:**相似変換**

行列$A$をある正則行列$P$を用いて$P^{-1}AP$と変換したとき,
$A$と$P^{-1}AP$の固有値は等しくなる*

**証明:** $A$と$P^{-1}AP$の特性方程式が一致すれば固有値は等しい

$$\det(P^{-1}AP - \lambda I) = \det(P^{-1}AP - \lambda \underbrace{P^{-1}IP}_{P^{-1}P = I\text{なので}}) = \det(P^{-1}(A - \lambda I)P)$$

$$= \underbrace{\det(P^{-1})\det(A - \lambda I)\det(P)}_{\text{行列式の性質}\det(AB) = \det(A)\det(B)\text{より}} = \underbrace{\frac{1}{\det(P)}}_{\text{行列式の性質}\det(A^{-1}) = 1/\det(A)\text{より}}\det(P)\det(A - \lambda I)$$

$$= \det(A - \lambda I)$$

(証明終了)

どのような行列に変換すると良いのだろうか?

実際に$3 \times 3$とかの行列の固有値を求めてみると分かるが,
**非対角成分に0が多い**と固有値は求めやすい

 *固有値は変わらないけど**固有ベクトルは変わる**ことに注意(行列自体が変わるので) 

# 相似変換

固有値が求めやすい行列の形は?

$3 \times 3$の行列を例にすると:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \overset{\text{特性方程式}}{\Longrightarrow} \det(A - \lambda I) = \begin{vmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{vmatrix}$$

この列を中心に行列式を**余因子展開**

$$(a_{11} - \lambda)\begin{vmatrix} a_{22} - \lambda & a_{23} \\ a_{32} & a_{33} - \lambda \end{vmatrix} - a_{21}\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} - \lambda \end{vmatrix} - a_{31}\begin{vmatrix} a_{12} & a_{13} \\ a_{22} - \lambda & a_{23} \end{vmatrix}$$

$a_{21} = a_{31} = 0$で後ろの2項が0になって消える. 更に$a_{32} = 0$ならば,

$$(a_{11} - \lambda)(a_{22} - \lambda)(a_{33} - \lambda) = 0$$

となり, **対角成分$a_{11}, a_{22}, a_{33}$が固有値**ということになる.

0にした$a_{21}, a_{31}, a_{32}$は左下の非対角成分

⇒ **上三角行列の固有値は対角成分に等しい**



# QR法

行列$A$を**直交行列**$Q$と**上三角行列**$R$に分解することを**QR分解**と呼ぶ.

$$A = QR$$

直交行列は**逆行列=転置行列**となる行列のこと: $Q^{-1} = Q^T$

(この性質は結構重要なので覚えておこう)

今やりたいことは$A$の**相似変換**なので, 直交行列$Q$で相似変換すると:

$$Q^T AQ = Q^T QRQ = RQ$$

変換された結果は$RQ$だけど, この手順を**何度も繰り返す**と,
$Q^T AQ$は**上三角行列に収束する**ことが分かっている*.

 *これの証明は参考文献「C言語による数値計算入門(サイエンス社)」のp208-209参照 

# QR法

$k$回目の反復における行列を$A^{(k)}$とすると

1. $A^{(k)}$を$Q^{(k)}R^{(k)}$に分解：
$$A^{(k)} = Q^{(k)}R^{(k)}$$
2. $R^{(k)}$と$Q^{(k)}$を逆に掛けて$A^{(k+1)}(= Q^T AQ)$を計算：
$$A^{(k+1)} = R^{(k)}Q^{(k)}$$

この手順を繰り返すことで固有値を計算する方法を **QR法** と呼ぶ

⇩

QR法には**QR分解が必要**

QR分解の方法にはいくつかある*がここでは**ハウスホルダー変換**を用いた方法を説明する(固有値を求める別の方法でも使うことが多いので)

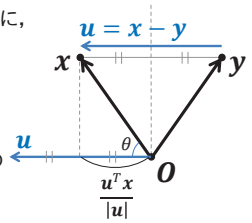

*他にはグラム・シュミットの直交化法を用いる方法やギブンス変換などがある


# ハウスホルダー変換

**ハウスホルダー変換**は**鏡映**を表す線形変換

右の図のように2つのベクトル$\boldsymbol{x}, \boldsymbol{y}$があるときに，
**$\boldsymbol{x}$と$\boldsymbol{y}$を結ぶ線の垂直二等分線による鏡映変換**を**行列**$H$として表したい

$$\boldsymbol{x} = H\boldsymbol{y} \quad \text{もしくは} \quad \boldsymbol{y} = H\boldsymbol{x}$$




$\boldsymbol{y}$から$\boldsymbol{x}$へのベクトル$\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{y}$を考え，
それを右図のように原点$O$に置くと$\boldsymbol{x}$との間の角度は内積で表され，

$$\boldsymbol{u} \cdot \boldsymbol{x} = \boldsymbol{u}^T\boldsymbol{x} = |\boldsymbol{u}||\boldsymbol{x}|\cos\theta$$

ベクトル$\boldsymbol{u}$の長さは元の定義から$|\boldsymbol{x}|\cos\theta$の2倍なので，$2\frac{\boldsymbol{u}^T\boldsymbol{x}}{|\boldsymbol{u}|}$になる．

$\boldsymbol{u}$方向の単位ベクトル$\boldsymbol{u}/|\boldsymbol{u}|$を使うと$\boldsymbol{u} = 2\frac{\boldsymbol{u}\boldsymbol{u}^T}{|\boldsymbol{u}|^2}\boldsymbol{x}$となる*ので，

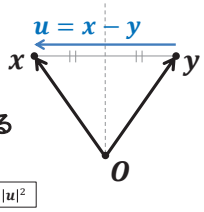
$$\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{u} = \boldsymbol{x} - 2\frac{\boldsymbol{u}\boldsymbol{u}^T}{|\boldsymbol{u}|^2}\boldsymbol{x} = \left(\boldsymbol{I} - 2\frac{\boldsymbol{u}\boldsymbol{u}^T}{|\boldsymbol{u}|^2}\right)\boldsymbol{x} = H\boldsymbol{x}$$


*$\boldsymbol{u}$は縦ベクトルなので$\boldsymbol{u}\boldsymbol{u}^T$は行列になることに注意しよう


# ハウスホルダー変換

**ハウスホルダー行列$H$**による変換を**ハウスホルダー変換**という

**ハウスホルダー行列**：$H = I - 2\frac{\boldsymbol{u}\boldsymbol{u}^T}{|\boldsymbol{u}|^2}$




- $\boldsymbol{x} = H\boldsymbol{y}$ と $\boldsymbol{y} = H\boldsymbol{x}$ のどちらも成り立つ
  ⇒ 同じ大きさの**2ベクトルを入れ替えられる**
- ハウスホルダー行列$H$は**対称行列**
$$H^T = H$$
- ハウスホルダー行列$H$は**直交行列**

$u^T u = |u|^2$ ⇩

$$H^T H = H^2 = \left(I - 2\frac{\boldsymbol{u}\boldsymbol{u}^T}{|\boldsymbol{u}|^2}\right)^2 = I - 4\frac{\boldsymbol{u}\boldsymbol{u}^T}{|\boldsymbol{u}|^2} + 4\frac{\boldsymbol{u}\boldsymbol{u}^T\boldsymbol{u}\boldsymbol{u}^T}{|\boldsymbol{u}|^2|\boldsymbol{u}|^2} = I$$
$$\Rightarrow H^{-1} = H^T$$

ハウスホルダー変換は**2つのベクトルの入れ替え**を可能にする変換

⇒ 行列$A$の1列目を$(b_{ii}, 0, 0, \ldots, 0)^T$というベクトルと入れ替えれれば…




# QR分解

**ハウスホルダー変換を用いたQR分解**

$3 \times 3$の行列$A$を試しにハウスホルダー変換でQR分解してみよう

$$A^{(0)} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & a_{23}^{(0)} \\ a_{31}^{(0)} & a_{32}^{(0)} & a_{33}^{(0)} \end{pmatrix}$$

⇦ 反復処理していくので上付きで反復回数をつけている．$k$番目の反復では$A^{(k)}$

⇩

まずはこの列ベクトルを$(a_{11}^{(1)}, 0, 0)$**となるように変換**する

$$\boldsymbol{x}_1 = \begin{pmatrix} a_{11}^{(0)} \\ a_{21}^{(0)} \\ a_{31}^{(0)} \end{pmatrix} \text{と} \quad \boldsymbol{y}_1 = \begin{pmatrix} a_{11}^{(1)} \\ 0 \\ 0 \end{pmatrix} \text{という2つのベクトルを考える}$$

2つのベクトルの大きさを同じ($|\boldsymbol{x}_1| = |\boldsymbol{y}_1|$)にするために

$$a_{11}^{(1)} = |\boldsymbol{x}_1| = \sqrt{\left(a_{11}^{(0)}\right)^2 + \left(a_{21}^{(0)}\right)^2 + \left(a_{31}^{(0)}\right)^2} \quad \text{とする}$$




# QR分解

$\boldsymbol{x}_1$と$\boldsymbol{y}_1$のハウスホルダー行列は，$\boldsymbol{u}_1 = \boldsymbol{x}_1 - \boldsymbol{y}_1$とすると

$$H^{(0)} = I - 2\frac{(\boldsymbol{x}_1 - \boldsymbol{y}_1)(\boldsymbol{x}_1 - \boldsymbol{y}_1)^T}{|\boldsymbol{x}_1 - \boldsymbol{y}_1|^2} = I - 2\tilde{\boldsymbol{u}}_1\tilde{\boldsymbol{u}}_1^T \quad \text{ここで } \tilde{\boldsymbol{u}}_1 = \frac{\boldsymbol{x}_1 - \boldsymbol{y}_1}{|\boldsymbol{x}_1 - \boldsymbol{y}_1|}$$

$H^{(0)}$を$A^{(0)}$に**左から掛ける**と

$$A^{(1)} = H^{(0)}A^{(0)} = (\underline{H^{(0)}\boldsymbol{x}_1}\ H^{(0)}\boldsymbol{x}_2\ \cdots\ H^{(0)}\boldsymbol{x}_n)$$
$$\underline{\boldsymbol{y}_1}$$

⇩

$$A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} \end{pmatrix}$$

：1列目の下部分が0になる形に変換できた

2列目以降の要素も値は変わっているので注意
(上付きの添え字が変わっていることに注意)




# QR分解

2列目についても同じことを行う

$$A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} \end{pmatrix}$$

変換すべきベクトルの次元が1つ小さくなっていることに注意

⇩

$$\boldsymbol{x}_2 = \begin{pmatrix} a_{22}^{(1)} \\ a_{32}^{(1)} \end{pmatrix} \Rightarrow \boldsymbol{y}_2 = \begin{pmatrix} a_{22}^{(2)} \\ 0 \end{pmatrix} \qquad H^{(1)} = I - 2\tilde{\boldsymbol{u}}_2\tilde{\boldsymbol{u}}_2^T, \quad \tilde{\boldsymbol{u}}_2 = \frac{\boldsymbol{x}_2 - \boldsymbol{y}_2}{|\boldsymbol{x}_2 - \boldsymbol{y}_2|}$$

ハウスホルダー行列$H$の大きさが$2 \times 2$になっていることに注意．
$A^{(1)}$の1列目は変えずに2列目以降に$H^{(1)}$を掛けるには

$$A^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & h_{11}^{(1)} & h_{12}^{(1)} \\ 0 & h_{21}^{(1)} & h_{22}^{(1)} \end{pmatrix} A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(2)} \end{pmatrix} = R$$

# QR分解

$A \to R$と変換してしまったが，やりたいことは$A = QR$**の変換**
(こちらはイコールになっていることに注意)

$A^{(k)}$の更新をまとめると：

$$A^{(1)} = H^{(0)}A^{(0)} \quad \Longrightarrow \quad A^{(0)} = \left(H^{(0)}\right)^{-1}A^{(1)}$$

$$R = A^{(2)} = H^{(1)}A^{(1)} = H^{(1)}\left(H^{(0)}A^{(0)}\right)$$

$$\Longrightarrow \quad A^{(0)} = \left(H^{(0)}\right)^{-1}\left(H^{(1)}\right)^{-1}R$$

$H$は直交行列なので，$Q = \left(H^{(0)}\right)^{T}\left(H^{(1)}\right)^{T}$とすれば*

$$A^{(0)} = QR$$ **QR分解できた!**

実際には$H^{(1)}$は$\begin{pmatrix} 1 & 0 & 0 \\ 0 & h_{11}^{(1)} & h_{12}^{(1)} \\ 0 & h_{21}^{(1)} & h_{22}^{(1)} \end{pmatrix}$という形にしていたことに注意．
この行列も直交行列になる．

*直交行列同士の積は直交行列になるのでここでの$Q$は直交行列になっている



# QR分解

$n \times n$の場合でもやることは同じ．

QR分解の手順をまとめると：

1. 上三角行列$R^{(0)}$を$A$で初期化，直交行列$Q^{(0)}$を単位行列で初期化：$R^{(0)} = A, Q^{(0)} = I$
2. 以下の手順を$k = 0 \sim n-2$まで繰り返す
   a. ベクトル$\boldsymbol{x}_k = \left(R_{k,k}^{(k)}, R_{k+1,k}^{(k)}, \dots, R_{n-1,k}^{(k)}\right)$の大きさ$a_{k,k}^{(k+1)} = |\boldsymbol{x}_k|$を計算
   b. $\boldsymbol{y}_k = \left(a_{k,k}^{(k+1)}, 0, \dots, 0\right)^{T}$を設定して，$\tilde{\boldsymbol{u}}_k = (\boldsymbol{x}_k - \boldsymbol{y}_k)/|\boldsymbol{x}_k - \boldsymbol{y}_k|$を計算
   (このときの$\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{u}_k$の要素数は$k \sim n-1$の$n-k$個)
   c. ハウスホルダー行列$H^{(k)} = I - 2\tilde{\boldsymbol{u}}_k\tilde{\boldsymbol{u}}_k^{T}$ を計算
   d. $R^{(k+1)} = H^{(k)}R^{(k)}$，$Q^{(k+1)} = Q^{(k)}\left(H^{(k)}\right)^{T}$を計算

前スライドまでの$A^{(k)}$を$R^{(k)}$と書き換えている．また，このスライドではインデックスが0スタートになっているので注意



# QR分解

QR分解のコード例

行列$R, Q, H$用の2次元配列$(n \times n)$が確保され，行列$A$も2次元配列で与えられている

```
R = A; unit(Q, n); // RをA^(0)，Qを単位行列で初期化
vector<double> u(n, 0.0);
for(int k = 0; k < n-1; ++k){
    // a_kk^(k+1)の計算 (ベクトルxの大きさ)
    double akk = 0.0;
    for(int i = k; i < n; ++i){
        akk += R[i][k]*R[i][k];
    }
    akk = sqrt(akk);

    // u=(x1-y1)/|x1-y1|の計算
    double l = 0.0;
    for(int i = k; i < n; ++i){
        u[i] = R[i][k]-(i == k ? akk : 0.0);
        l += u[i]*u[i];
    }
    if(fabs(l) < 1e-10) break;
    l = sqrt(l);
    for(int i = k; i < n; ++i) u[i] /= l;

    (次のページに続く)
```

$a_{k,k}^{(k+1)} = |\boldsymbol{x}_k| = \sqrt{\left(R_{k,k}^{(k)}\right)^2 + \left(R_{k,k+1}^{(k)}\right)^2 + \cdots + \left(R_{k,n-1}^{(k)}\right)^2}$の計算．
$k$が進むにつれてベクトルのサイズが小さくなっていくことに注意
(サイズ$n$の配列の後ろの方しか使わなくなる)

$\tilde{\boldsymbol{u}}_k = (\boldsymbol{x}_k - \boldsymbol{y}_k)/|\boldsymbol{x}_k - \boldsymbol{y}_k|$の計算．$x_k, y_k$は明示的に配列に格納せず，その値を直接計算している．ベクトルの正規化を忘れずに!



# QR分解

QR分解のコード例(前ページからの続き)

行列$R, Q, H$用の2次元配列$(n \times n)$が確保され，行列$A$も2次元配列で与えられている

```
    // ハウスホルダー行列H^(k)の計算(H=I-2uu^T)
    unit(H, n); // Hを単位行列で初期化
    for(int i = k; i < n; ++i){
        for(int j = k; j < n; ++j){
            H[i][j] -= 2*u[i]*u[j];
        }
    }

    // A^(k+1) = H^(k) A^(k)
    R = mul_mm(H, R, n);

    // Q^(k+1) = Q^(k) (H^(k))^T
    Q = mul_mm(Q, transpose(H, n), n);
}
```

ハウスホルダー行列
$H^{(k)} = I - 2\tilde{\boldsymbol{u}}_k\tilde{\boldsymbol{u}}_k^{T}$ の計算．
`unit()`は単位行列を作る関数

$R^{(k+1)} = H^{(k)}R^{(k)}$
$Q^{(k+1)} = Q^{(k)}\left(H^{(k)}\right)^{T}$
で上三角行列と直交行列を更新していく．`mul_mm()`は行列同士の積を計算する関数．`transpose()`は転置行列を計算する関数



# QR分解

QR分解の実行結果例

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 5 & 3 \end{pmatrix}$$

```
Q = 0.816497 -0.492366 0.301511
    0.408248 0.123091 -0.904534
    0.408248 0.86164 0.301511
R = 2.44949 3.67423 1.63299
    9.37286e-16 4.06202 2.70801
    -5.73968e-16 0 1.11022e-16
QR = 2 1 -4.10615e-16
     1 2 1
     1 5 3
QQ = 1 -5.55112e-17 -8.32667e-17
     -5.55112e-17 1 1.66533e-16
     -8.32667e-17 1.66533e-16 1
```

⇐ チェック用に$Q$と$R$の積を計算

⇐ $Q$が直交行列になっているかチェックするために$QQ^{T}$を計算

数値誤差はあるものの行列$A$は直交行列$Q$と上三角行列$R$に分解できている



# QR法

$k$回目の反復における行列を$A^{(k)}$とすると

1. $A^{(k)}$を$Q^{(k)}R^{(k)}$に分解：
   $$A^{(k)} = Q^{(k)}R^{(k)}$$
2. $R^{(k)}$と$Q^{(k)}$を逆に掛けて$A^{(k+1)}(= Q^{T}AQ)$を計算：
   $$A^{(k+1)} = R^{(k)}Q^{(k)}$$

この手順を繰り返すことで固有値を計算する方法を **QR法** と呼ぶ

ハウスホルダー変換でQR分解が可能になったので
今度はQR法で固有値を求めてみよう!

# QR法

QR法のコード例

行列$R, Q, H$用の2次元配列($n \times n$)が確保され，行列$A$も2次元配列で与えられている

```
vector< vector<double> > R(A), Q(A); // R,Q用配列を確保
vector<double> lambda(n, 0.0); // 収束判定用
double e;
int k;
for(k = 0; k < max_iter; ++k){
    // A^(k) ⇐ Q^(k) R^(k)
    qr_decomposition(A, Q, R, n);

    // A^(k+1) ⇐ R^(k) Q^(k)
    A = mul_mm(R, Q, n);

    // 収束判定
    e = 0;
    for(int i = 0; i < n; ++i) e += fabs(lambda[i]-A[i][i]);
    if(e/n < eps) break;

    for(int i = 0; i < n; ++i) lambda[i] = A[i][i];
}
```

$A^{(k)} = Q^{(k)}R^{(k)}$と
$A^{(k+1)} = R^{(k)}Q^{(k)} (= Q^T AQ)$
の計算

収束判定は前ステップの固有値の値の差を使う



# QR法

QR法の実行結果例

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 5 & 3 \end{pmatrix} \Rightarrow$$

固有値$\lambda = 5, 2, 0$

固有ベクトル$\boldsymbol{v} = \begin{pmatrix} 1 \\ 3 \\ 8 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}$

許容誤差$\varepsilon = 1 \times 10^{-6}$

```
0 : lambda = 4.16667, 2.83333, -1.39583e-16
1 : lambda = 5.33333, 1.66667, -7.98978e-17
2 : lambda = 5.20225, 1.79775, -7.98978e-17
3 : lambda = 5.08757, 1.91243, -7.98978e-17
4 : lambda = 5.03585, 1.96415, -7.98978e-17
     : (省略)
13 : lambda = 5.00001, 1.99999, -7.98978e-17
14 : lambda = 5, 2, -7.98978e-17
15 : lambda = 5, 2, -7.98978e-17
e = (5, 2, -7.98978e-17)
iter = 15, eps = 0
```

16回の反復で処理終了

べき乗法と反復回数自体はそんなに変わらないけど，**3つすべての固有値**が求まっている(ただし，毎反復QR分解をするので計算コストはこちらの方がずっと高い)



# 逆反復法

QR法で固有値が分かったけど**固有ベクトルは？**

**べき乗法**で求められる？

⇒ 最大固有値に対応する固有ベクトルのみ

行列$A$の最小固有値を$\lambda_{min}$とすると，その逆行列$B = A^{-1}$の最大固有値は$1/\lambda_{min}$となる*

⇒ 行列$B = A^{-1}$についてべき乗法を適用すると行列$A$の**最小固有値に対応する固有ベクトル**が求められる

**逆反復法**

逆反復法を使って，$\lambda_i$**それぞれに対応するように行列$B$を設定**してやれば，すべての固有値の固有ベクトルが求められそう！

*$A\boldsymbol{v} = \lambda\boldsymbol{v}$より$A^{-1}\boldsymbol{v} = \frac{1}{\lambda}\boldsymbol{v}$となるため．



# 逆反復法

特性方程式の導出を思い出すと$(A - \lambda I)\boldsymbol{v} = \boldsymbol{0}$で$\boldsymbol{v} = \boldsymbol{0}$とならないために$\det(A - \lambda I) = 0$とした．

⇒ QR法で求めた固有値は近似値なので，
正確には$(A - \lambda I)\boldsymbol{v} = \boldsymbol{0}$になっていない？

$i$番目の固有値を$\lambda_i$として，その近似値を$\hat{\lambda}_i$とすると

$$(A - \hat{\lambda}_i I)\boldsymbol{v}_i = (\lambda - \hat{\lambda}_i)\boldsymbol{v}_i \Rightarrow (A - \hat{\lambda}_i I)^{-1}\boldsymbol{v}_i = \frac{1}{(\lambda - \hat{\lambda}_i)}\boldsymbol{v}_i$$

($A\boldsymbol{v} = \lambda\boldsymbol{v}$より)

$B = (A - \hat{\lambda}_i I)^{-1}$とするとべき乗法で最大の$\frac{1}{(\lambda - \hat{\lambda}_i)}$の時の固有ベクトルが求まる．

⇒ これが$\lambda = \lambda_i$の時のものであればOK



# 逆反復法

$i$番目の固有値の近似値$\hat{\lambda}_i$がその他の近似値よりも真値$\lambda_i$に近いならば，

$$|\lambda_i - \hat{\lambda}_i| < |\lambda_j - \hat{\lambda}_i| \qquad (i \neq j)$$

が成り立つので，べき乗法を使った場合，$1/|\lambda_i - \hat{\lambda}_i|$が行列$B$の最大固有値となるので，

適当な初期値$\boldsymbol{y}^{(0)}$から

$$\boldsymbol{y}^{(k)} = B\boldsymbol{y}^{(k-1)} = (A - \hat{\lambda}_i I)^{-1}\boldsymbol{y}^{(k-1)}$$

により$\boldsymbol{y}$を更新していくと，固有値$\lambda_i$の固有ベクトル$\boldsymbol{v}_i = \lim_{k\to\infty} \boldsymbol{y}^{(k)}$が得られる



# 逆反復法

実際に解かないといけないのは

$$(A - \hat{\lambda}_i I)\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)}$$

という線形システムになる．

⇒ $(A - \hat{\lambda}_i I)$の部分は$k$反復中は変わらないので
予め**LU分解**しておき，線形システムを$\boldsymbol{y}^{(k)}$について解く

⇒ べき乗法と同じく$\boldsymbol{y}^{(k)}$は毎ステップ正規化する

⇒ レイリー商$R(\boldsymbol{y}^{(k)})$で固有値$1/|\lambda_i - \hat{\lambda}_i|$が求まるので
その変化が閾値以下になったら反復終了(収束条件)とする

べき乗法を繰り返すだけなのでコード例は省略

# QR法＋逆反復法

QR法＋逆反復法の実行結果例

$A = \begin{pmatrix} 1 & -2 & -2 \\ -2 & 2 & 0 \\ -2 & 0 & 0 \end{pmatrix}$ ⇨ 固有値$\lambda = 4, -2, 1$

固有ベクトル$\boldsymbol{v} = \begin{pmatrix} -2 \\ 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ -2 \end{pmatrix}$

許容誤差$\varepsilon = 1 \times 10^{-6}$

```
Q =  0.333333 -0.666667 -0.666667
    -0.666667 0.333333 -0.666667
    -0.666667 -0.666667 0.333333
R = 3 -2 -0.666667
    6.66134e-16 2 1.33333
    5.55112e-16 0 1.33333
e = (4, -2, 1)
iter = 12, eps = 2.68221e-07

v1 = (-0.666667, 0.666667, 0.333333)
v2 = (0.666667, 0.333333, 0.666667)
v3 = (0.333333, 0.666667, -0.666667)
iter = 2, eps = 3.67933e-10
```

固有ベクトル

逆反復法ではLU分解(ピボット交換なし)を使ったので$(A - I\lambda)$の対角が0になると解が求められないことあり．ちゃんとピボット交換を使おう．
(計算例の行列がここまでのものと違うのはそのせい)



# QR法

QR法は反復内でQR分解を行うので**計算効率が良くない**
⇒ **高速化手法**が提案されている

- **ウィルキンソンの移動法**(原点シフト)

  $A^{(k)}$の左下要素が0に近づくにつれて対角との値の差が大きくなり，収束しにくくなるということが発生する．

  ⇒ 対角要素でも絶対値が最小になるもの($\mu^{(k)}$)で減算して対角の値を小さくすれば良い
  QR法の更新式を以下のように書き換えることで反復を減らす

$$A^{(k)} - \mu^{(k)} I = Q^{(k)} R^{(k)}$$
$$A^{(k+1)} = R^{(k)} Q^{(k)} + \mu^{(k)} I$$



# QR法

QR法は反復内でQR分解を行うので**計算効率が良くない**
⇒ **高速化手法**が提案されている

- **ヘッセンベルグ行列**への変換

  QR分解は$n$が大きくなると計算に時間が掛かるが，$A$を以下のようなヘッセンベルグ行列に変換してからQR法を適用すると，より高速にQR分解できることが分かっている．

$$H = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n-1} & a_{2,n} \\ 0 & a_{3,2} & a_{3,3} & \cdots & a_{3,n-1} & a_{3,n} \\ 0 & 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & \cdots & 0 & a_{n,n-1} & a_{n,n} \end{pmatrix}$$

ヘッセンベルグ行列への変換はハウスホルダー変換やギブンス変換などでできる
⇒ ヘッセンベルグ行列を用いたQR法のコードはサンプルにも含まれている



# その他の方法

その他の固有値計算方法

- **対称行列**限定の方法
  - **ハウスホルダー法**
    ハウスホルダー変換で**三重対角行列**に相似変換した後，特性方程式を**DKA法**や**二分法+ニュートン法**で解く方法．
    解のある範囲を**Gerschgorin(ゲルシュゴリン)の定理**で求め，**Sturm(スツルム)の定理**で解が1つだけの範囲を判定，ニュートン法で根を求める
  - **ヤコビ法**
    **回転行列**を利用して，対称行列を**対角行列**になるまで相似変換して固有値を求める方法

⇒ ハウスホルダー法，ヤコビ法どちらもgithubのサンプルにも含まれているので気になる人はそちらも参照してください．



# 講義内容のまとめ

- 今日の問題：固有値，固有ベクトル
- べき乗法
  - 最大固有値に対応する固有ベクトルを求める
  - レイリー商による固有値計算
- ハウスホルダー変換
  - 相似変換と固有値を求めやすい行列への変換
  - 鏡映変換を用いた相似変換
- QR法
  - 上三角行列へ変換することで固有値を簡単に求める
  - 逆反復法による固有ベクトル算出



# 講義全体を通して

今回でこの講義は終了です(レポートは残っているけど)．

皆さんが今後他の講義や研究，仕事などで出てきた**数式をどうやってコンピュータで実装しようか**と悩んだときに，この講義を思い出してくれると幸いです．

講義で示した例は比較的単純なものばかりですが，複雑なものにも適用可能です．
大事なのは**どの問題にどの方法を適用するか？**ということだと覚えておいてください．

# Appendix

(以降のページは補足資料です)



## 線形独立と線形従属

ベクトルの集合$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$に対して

$$c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n = 0$$

となる係数が$c_1 = c_2 = \cdots = c_n = 0$以外にないときそのベクトル集合は**線形独立**であるという.

逆に上の式の解に$c_i \neq 0$となるものが1つでも含まれていたら, そのベクトル集合は**線形従属**であるという.

線形独立の場合は, あるベクトル$\boldsymbol{x}$を

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

として表した場合, $(c_1, c_2, \ldots, c_n)$は**一意に定まる**($\boldsymbol{v}_i$は線型代数学における**基底**になる)