電気303/電情303 数値解析 (6)

固有値問題

はじめに

- 黎明期から, コンピュータの重要な用途は数値計算
- 黎明期のプログラミング言語は COBOL (事務用) と Fortran (科学技術計算用); 登場は 20 世紀中盤と古いが, 今でも使われている

- 多くのプログラマが同じような数値計算をするためのコードを個別に書くのは無駄が多いので…
- 共通性が高い処理の部分が「サブルーチン集」として纏められるようになった
- 当時の記述言語は基本的に Fortran

- これからしばらく続く歴史に関する議論の典拠は以下の通り.

Elizabeth Jessup, Numerical Linear Algebra,
ftp://ftp.mcs.anl.gov/pub/petaflops/summer.study/linear.tex

www.netlib.org/

http://jp.mathworks.com/company/newsletters/
articles/matlab-incorporates-lapack.html

https://www.scilab.org/scilab/history

http://math-atlas.sourceforge.net/

https://software.intel.com/en-us/intel-mkl/

- コンピュータによる科学技術計算において, まず最初に必要となるのは, 線形代数に関連した計算 (線形計算).
- このために作られたサブルーチン集として, LINPACK と EISPACK が有名.

- LINPACK は 1979 年にリリースされた線形方程式を解くためのサブルーチン集.
- EISPACK は 1976 年にリリースされた固有値問題を解くためのサブルーチン集.

- 国際的に, 数値計算ソフトウェアの事実上の標準は MATLAB で…
- 無料で使える Scilab や GNU Octave は MATLAB の代替ソフトウェアなのであるが…

- MATLABは, 1970年代に, LINPACKとEISPACKに基づく対話的な計算ソフトとして出発した (今日では線形計算はMATLABのごく一部になっている).
- Scilabは1980年代にMATLABと類似したソフトとして開発された (当時の名称はBlaise),

- 今日では,LINPACK と EISPACK は LAPACK というパッケージに統合されている.
- 数値計算ライブラリの Fortran 以外の言語への移行は 20 世紀後半になってから徐々に進んでいる

- その他，ATLAS(Automatically Tuned Linear Algebra Software) という数値計算のライブラリを特定のコンピュータ向きに高効率化するソフトウェアも普及しているし, Intel は, Math Kernel Library というライブラリを公開している.

- 黎明期に既に LINPACK と EISPACK が併存したことからわかるように, 線形方程式を解くことと, 固有値問題を解くことは, 線形計算の二大重要トピック.
- 今回の講義では (この講義で指定した教科書では解説されていないが) 固有値問題の解法について概説する.

今回の講義の参考文献

- 杉原, 室田, 線形計算の数理, 岩波書店, 2009.
- 森, 数値解析, 第 2 版, 共立出版, 2002.
- 山本, 数値解析入門 [増補版], サイエンス社, 2003.
- 斎藤, 数値解析入門, 東京大学出版会, 2012.

- 久保田, 工学基礎 数値解析とその応用, 数理工学社, 2010.
- 伊理, 藤野, 数値計算の常識, 共立出版, 1985.
- 伊理, 韓, 線形代数, 教育出版, 1977.

固有値と固有ベクトル

- まず固有値と固有ベクトルについて復習する.
- 以下の議論では, $\boldsymbol{A}$ は n 次の正方行列であることを仮定する. 要素は実数でも複素数でもよい.

- n 次の正方行列 $\boldsymbol{A}$ に対し, 複素数 λ と n 次のベクトル $\boldsymbol{x} \neq \boldsymbol{0}$ が

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

という方程式を満たすとき, λ を $\boldsymbol{A}$ の固有値, $\boldsymbol{x}$ を固有値 λ に対応する固有ベクトルという.

- n 次行列 $\boldsymbol{A}$ の固有値は, 代数方程式

$$\det(s\boldsymbol{I} - \boldsymbol{A}) = 0$$

の根であり (ただし $\boldsymbol{I}$ は n 次の単位行列で s は変数), 重複度を含めて高々 n 個である.

- 固有値 λ に対応する固有ベクトル全体が張る線形部分空間を λ に対応する固有空間という.
- 行列 $\boldsymbol{A}$ の固有ベクトル全体が張る線形部分空間 (どの固有値に対応するかは問わない) を, $\boldsymbol{A}$ の固有空間という. $\boldsymbol{A}$ の固有空間は必ずしも全空間とは一致しない.

- $\boldsymbol{A}$ の固有空間を拡張して全空間を張るようにしたものを, 一般化固有空間という. これに対応するのが Jordan 標準形である.

- 固有値と固有ベクトルの典型的な応用は…
  - ▷ 線形連立微分方程式や線形連立差分方程式の求解および安定性解析
  - ▷ パターン認識
  - ▷ 建造物の共振の解析

  など様々．google によるページのランキングにも固有値が使われている．

固有値の数値計算

- 行列 $\boldsymbol{A}$ の固有値をすべて求めるには, 数学的には,

$$\det(s\boldsymbol{I} - \boldsymbol{A}) = 0$$

という代数方程式のすべての解を求めればよいのであるが, 数値計算という観点では, この方法は効率が悪い. その理由は…

- 行列式

$$\det(s\boldsymbol{I} - \boldsymbol{A})$$

の導出だけでも計算量が多く, さらに数値計算の誤差を受けやすい.

- 代数方程式の求解も数値計算の誤差を受けやすい

- よって, 固有値の計算には, 別の方法 (後述) が用いられる.

- 前回までの講義で述べてきた連立一次方程式の解法と比較すると:
  - ▷ 連立一次方程式が理論的には有限回の計算で解ける
  - ▷ 固有値問題を代数方程式を解く問題と解釈すると, 5 次以上の行列に対しては有限回の演算では解けない.

- よって, コンピュータによって行列の固有値を求める問題は, 本質的に近似の問題である.

- Scilab ではどのようにして固有値問題を解くかというと, 固有値を求めるための Scilab の関数 spec は LAPACK のライブラリを利用しており, オンラインマニュアルには以下のように記載されている.

行列の固有値計算は Lapack ルーチンに基づいています.

1. 行列が対称でない場合,
   DGEEV および ZGEEV.
2. 行列が対称の場合,
   DSYEV および ZHEEV.

DGEEV, ZGEEV, DSYEV, ZHEEV はそれぞれ実非対称行列, 複素非エルミート行列, 実対称行列, 複素エルミート行列を対象とするライブラリで, いずれも後で述べる QR 法に帰着させて問題を解いている.

- QR 法は, 小規模な固有値問題を解くときに使われる代表的な解法で, 後で述べる Schur 分解を近似的に求める解法である.

- QR 法のアルゴリズムは単純だが, その収束性に関する解析は極めて繁雑である (この講義では解析については述べない).

- 絶対値が最大 (あるいは最小) の固有値をひとつ求めるだけであれば, 解析も含めてもっと単純な解法があり, 冪乗法と呼ばれている.
- 冪乗法の利点は簡単なことであるが, 応用上, 絶対値が相対的に大きいいくつかの代表的な固有値を求めれば十分であることがあり, このような場合には, 実用上も十分役に立つ.

- この講義では, まず冪乗法とそのバリエーションについて説明してから, QR 法について説明する.

- QR 法は小規模な固有値問題を解くためのアルゴリズムであり, より大規模な固有値問題を解くためには, Arnoldi 法, Jacobi-Davison 法などといった, 別の解法が使われる. この講義では, これらについては名前を紹介するに留める (詳細は [杉原, 室田] 参照).

冪乗法

- 冪乗法 (累乗法とも呼ぶ) は, 初期値 $\boldsymbol{x}_0$ を適当に取り,

$$\lim_{k\to\infty} \frac{\boldsymbol{A}^k \boldsymbol{x}_0}{\|\boldsymbol{A}^k \boldsymbol{x}_0\|}$$

を近似的に求めることで絶対値最大の固有値に対応する固有ベクトルを得る方法である.

- 得られるのが固有値 (の近似値) ではなく固有ベクトル (の近似値) であることに注意.
- 固有ベクトル $\boldsymbol{v}$ が得られた時点で, 対応する固有値は,

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

となるスカラーを求めることにより得られるが, アルゴリズムの終了時点でこれを求めることも可能 (後述).

- 冪乗法が適用できるためには条件がある (以下ではこれを仮定する):
  - ▷ 行列 $\boldsymbol{A}$ の絶対値最大の固有値が唯一で, その重複度と対応する固有空間の次元が一致する.
- アルゴリズムの形で書き下すと, 冪乗法は次ページの通り.

- 冪乗法のアルゴリズム
  初期値 $\boldsymbol{x}^{(0)}$ を任意の単位ベクトルとし, 以下の繰り返し計算をおこなう $(k = 1, 2, \ldots)$.

$$\boldsymbol{y}^{k+1} = \boldsymbol{A}\boldsymbol{x}_k,$$

$$\boldsymbol{x}_{k+1} = \frac{\boldsymbol{y}^{(k+1)}}{\|\boldsymbol{y}^{(k+1)}\|}$$

- 上記のアルゴリズムは,
  - ▷ ベクトル $\boldsymbol{x}_k$ に行列 $\boldsymbol{A}$ を左から掛ける
  - ▷ 得られたベクトルを正規化する

  という手順の繰り返し. このため, 冪乗法と呼ばれる.

- 先のアルゴリズムでは停止条件が定められていないが, 実用的には, $\boldsymbol{x}_{k+1}$ と $\boldsymbol{x}_k$ がほぼ並行になった時点で停止すればよい.

- 先に述べたように, 固有ベクトル $\boldsymbol{v}$ が得られれば, 対応する固有値の値は

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

となるスカラーを求めることで得られるが, アルゴリズムをステップ k で終了した場合には, $\boldsymbol{x}_k \simeq \boldsymbol{v}$, $\boldsymbol{y}_{k+1} \simeq \lambda \boldsymbol{v}$ となっているので, そこから 固有値 λ を求めることもできる.

- 冪乗法は, 実は初期値に関して条件があり (後述), 任意に選んだ初期値が (運悪く) この条件を満たさない場合には, 初期値を取り直して計算を再度実行する必要がある.

- 行列 $\boldsymbol{A}$ の固有値のうち絶対値が最大のものを λ_1 とする. 仮定により, λ_j $(j \geq 2)$ を $\boldsymbol{A}$ の他の固有値としたとき, $|\lambda_1| > |\lambda_j|$ である.

- 冪乗法によって, $k \to \infty$ としたときに, 初期値が一定の条件を満たせば (後述), 固有値 λ_1 に対応する固有ベクトルが得られることを, Jordan 標準形を用いて確認する.

- 固有値 λ_j に対応する (一般化) 固有ベクトルを $\{\boldsymbol{v}_{j,k} : 1 \leq k \leq \rho_j\}$ とする (ρ_j は固有値 λ_j に対応するジョルダン細胞の次元).
- $|\lambda_1|$ は絶対値最大の唯一の固有値で, その重複度と固有空間の次元が一致することが仮定されていた. この次元を ρ_1 とし, λ_1 に対応する固有ベクトルを $\{\boldsymbol{v}_{1,k} : 1 \leq k \leq \rho_1\}$ とする.

- 初期値 $\boldsymbol{x}_0$ を一般化固有ベクトルで展開する.

$$\boldsymbol{x}^{(0)} = c_{1,1}\boldsymbol{v}_{1,1} + \cdots + c_{1,\rho_1}\boldsymbol{v}_{1,\rho_1} + c_{2,1}\boldsymbol{v}_{2,1} + \cdots$$

- 上記の式から λ_1 の固有ベクトルに対応する成分を抜き出し,

$$\boldsymbol{w} = c_{1,1}\boldsymbol{v}_{1,1} + \cdots + c_{1,\rho_1}\boldsymbol{v}_{1,\rho_1}$$

とおく.

- $\boldsymbol{w} \neq \boldsymbol{0}$ であれば, $\boldsymbol{w}$ も 固有値 λ_1 に対応する $\boldsymbol{A}$ の固有ベクトルである.
- このようにしたとき, $\boldsymbol{w} \neq \boldsymbol{0}$ となることが冪乗法がうまくいくための十分条件である.

- 冪乗法で λ_1 に対応する固有ベクトルが得られることの証明:

  ▷ $\forall j > 1, |\lambda_1| > |\lambda_j|$ だったから,

$$\frac{\boldsymbol{A}^k \boldsymbol{x}_0}{\lambda_1^k} = \boldsymbol{w} + \boldsymbol{\varepsilon}(k)$$

  で (ただし $c_1 \neq 0$),

$$\lim_{k \to \infty} \|\boldsymbol{\varepsilon}(k)\| = 0.$$

▷ 冪乗法の式を書き直すと,

$$\boldsymbol{x}_k = \frac{\boldsymbol{A}^k \boldsymbol{x}_0}{\|\boldsymbol{A}^k \boldsymbol{x}_0\|}$$

となっている.

▷ したがって,

$$\boldsymbol{x}_k = \frac{\lambda_1^k(\boldsymbol{w} + \boldsymbol{\varepsilon}(k))}{\|\lambda_1^k(\boldsymbol{w} + \boldsymbol{\varepsilon}(k))\|}$$

であり, $\lim_{k\to\infty} \|\boldsymbol{\varepsilon}(k)\| = 0$ であったから, $\boldsymbol{x}_k$ は λ_1 の固有ベクトルの方向に近付く ■

- $\lambda_1 = r_1 e^{i\theta_1}$ とすると, $\boldsymbol{x}_k$ が漸近するベクトルは

$$e^{ik\theta_1}\boldsymbol{w}$$

であることに注意. $\boldsymbol{w}$ は λ_1 に対応する固有ベクトルであり, 初期値から決まる. $\theta_1 \neq 0$ の場合は, $\boldsymbol{x}_k$ は $\boldsymbol{w}$ のスカラー倍に漸近するが, そのスカラーは (絶対値は 1 だが) k によって変わる.

冪乗法のバリエーション

- 応用上は, $\boldsymbol{A}$ の絶対値最小の固有値を求める必要がある場合もある. この場合にも, 冪乗法のバリエーションを利用することができる.
- ただし, やはり仮定は必要.

- $\boldsymbol{A}$ の絶対値最小の固有値が唯一で, 零ではなく, その重複度と対応する固有空間の次元が一致すると仮定する.

- 先の仮定が満たされるとき, 行列 $\boldsymbol{A}$ は正則である. また, $\boldsymbol{A}$ の固有値が

$$\lambda_1, \ldots, \lambda_k$$

であるとき, $\boldsymbol{A}^{-1}$ の固有値は,

$$\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_k}$$

である.

- したがって, 先の条件が満たされているとき, $\boldsymbol{A}^{-1}$ に冪乗法を適用することで, 絶対値最小の固有値に対応する固有ベクトルを求めることができる. これを逆冪乗法あるいは逆反復と呼ぶ.
- これを素直に書き下すと, 次のようになる.

- **逆冪乗法 (Ver. 1.0)**
  初期値 $\boldsymbol{x}^{(0)}$ を任意の単位ベクトルとし, 以下の繰り返し計算をおこなう ($k = 1, 2, \ldots$).

$$\boldsymbol{y}^{k+1} = \boldsymbol{A}^{-1}\boldsymbol{x}_k,$$

$$\boldsymbol{x}_{k+1} = \frac{\boldsymbol{y}^{(k+1)}}{\|\boldsymbol{y}^{(k+1)}\|}$$

- 数値計算の分野では逆行列の利用は
  - ▷ 数値計算の誤差の影響が大きい
  - ▷ 計算量が多い

  という理由で好まれないので, これを次のように変形する.

- **逆冪乗法 (Ver. 1.1)**
  初期値 $\boldsymbol{x}^{(0)}$ を任意の単位ベクトルとし, 以下の繰り返し計算をおこなう ($k = 1, 2, \ldots$).

  - ▷ 連立一次方程式 $\boldsymbol{A}\boldsymbol{y}^{k+1} = \boldsymbol{x}_k$ を $\boldsymbol{y}_{k+1}$ について解く
  - ▷ $\boldsymbol{x}_{k+1} = \dfrac{\boldsymbol{y}^{(k+1)}}{\|\boldsymbol{y}^{(k+1)}\|}$ とする

- 上記に関連してさらに注意を述べる．逆行列は, 行列に関する連立一次方程式

$$\boldsymbol{AX} = \boldsymbol{I}$$

を解くことに得られ, これはベクトルに関する連立一次方程式を $\boldsymbol{A}$ の次元と同じ数解くことを意味するため, 上記の計算の繰り返し回数が多い場合には, 計算量という観点では逆行列を回避することのメリットはなくなる.

- 逆冪乗法の終了判定法は冪乗法と同一である.

- α を複素数としたとき, 逆冪乗法を行列

$$\boldsymbol{A} - \alpha \boldsymbol{I}$$

に適用することで, α に最も近い固有ベクトルと, それに対応する固有値を求めることができる. ただし, これが可能であるための前提は, α が $\boldsymbol{A}$ の固有値ではなく, α に (複素平面で) もっとも近い固有値が唯一で, その重複度と対応する固有空間の次元が一致する.

- このアルゴリズムを, シフト付き逆冪乗法と呼ぶ.
- シフト付き逆冪乗法のアルゴリズムは次ページの通り.

- シフト付き逆冪乗法

  初期値 $\boldsymbol{x}^{(0)}$ を任意の単位ベクトルとし, 以下の繰り返し計算をおこなう $(k = 1, 2, \ldots)$.

  - ▷ 連立一次方程式 $(\boldsymbol{A} - \alpha \boldsymbol{I})\boldsymbol{y}^{k+1} = \boldsymbol{x}_k$ を $\boldsymbol{y}_{k+1}$ について解く
  - ▷ $\boldsymbol{x}_{k+1} = \dfrac{\boldsymbol{y}^{(k+1)}}{\|\boldsymbol{y}^{(k+1)}\|}$ とする

- シフト付き逆冪乗法の終了判定法は冪乗法と同一である.

- 冪乗法は $\boldsymbol{A}$ の固有ベクトルをひとつ求める方法であったが, 一定の条件の下で, これを拡張して, $\boldsymbol{A}$ が一定の条件 (後述) を満たす場合に, 2 番目の固有値に対応する固有ベクトルを求めることができる.
- $\boldsymbol{A}$ の固有値を $\{\lambda_j : j = 1, 2, \ldots, k\}$ とする.

- $k > 2$ かつ $|\lambda_1| > |\lambda_2|$ で, さらに, $\forall j > 2$, $|\lambda_2| > |\lambda_j|$ と仮定する. さらに, λ_1 に対応する固有空間の次元は 1 で, $\boldsymbol{A}$ の固有ベクトルをすべて集めたものが基底になっていると仮定する.
- 上記の仮定は, 冪乗法が適用できるための仮定と比較して, ずっと強くなっていることに注意.

- $\boldsymbol{A}^H$ を, $\boldsymbol{A}$ の Hermite 転置 (転置してから各要素の共役複素数を取ったもの) とする.

- $\boldsymbol{A}$ の固有値が

$$\lambda_1, \ldots, \lambda_k$$

であるとき, $\boldsymbol{A}^H$ の固有値は

$$\bar{\lambda}_1, \ldots, \bar{\lambda}_k$$

である. これは, $\boldsymbol{A}$ の Jordan 標準形を考えればわかる.

- $\boldsymbol{A}$ の固有値 λ_1 に対応する固有ベクトルを $\boldsymbol{v}_1$ とし, $\boldsymbol{A}^H$ の固有値 $\bar{\lambda}_1$ に対応する固有ベクトルを $\boldsymbol{w}_1$ とする. これらは冪乗法で求めることができる.

- $j \geq 2$ に対し, 固有値 λ_j に対応する固有ベクトルを $\{\boldsymbol{v}_{j,m} : 1 \leq m \leq \rho_j\}$ とする. ただし, ρ_j は 固有値 λ_j に対応する固有空間の次元である.

- 以上の条件が満たされるとき, $1 < j$ と $1 \leq m \leq \rho_j$ に対し,

$$\boldsymbol{w}_1^H \boldsymbol{v}_{j,m} = 0$$

である. これを示す.

- $\boldsymbol{w}_1^H \boldsymbol{v}_{j,m} = 0,\ 1 < j,\ 1 \leq m \leq \rho_j$ の証明:

$$\boldsymbol{w}_1^H \boldsymbol{A} = (\boldsymbol{A}^H \boldsymbol{w}_1)^H = \lambda_1 \boldsymbol{w}_1^H$$

だから, $1 < j,\ 1 \leq m \leq \rho_j$ に対し,

$$\lambda_1 \boldsymbol{w}_1^H \boldsymbol{v}_{j,m} = \boldsymbol{w}_1^H \boldsymbol{A} \boldsymbol{v}_{j,m} = \lambda_j \boldsymbol{w}_1^H \boldsymbol{v}_{j,m}.$$

$\lambda_1 \neq \lambda_j$ だから, $\boldsymbol{w}_1^H \boldsymbol{v}_{j,m} = 0.$ でなければならない. ■

- 一方, $\boldsymbol{A}$ の固有値を集めることで基底が構成できると仮定されていたから,

$$\boldsymbol{w}_1^H \boldsymbol{v}_1 \neq 0$$

である (証明は次ページ).

- 背理法により先の主張を示す．$\boldsymbol{w}_1^H \boldsymbol{v}_1 = 0$ であったと仮定する．すると，$\boldsymbol{w}_1^H$ は基底の全要素と直交することになり，ゆえに零ベクトルとなるので，$\boldsymbol{w}$ が $\boldsymbol{A}^H$ の固有ベクトルであったという仮定に反する． ■

- 固有ベクトルには零でない定数倍の自由度があるので, これを利用し,

$$\boldsymbol{w}_1^H \boldsymbol{v}_1 = 1$$

となるよう $\boldsymbol{w}_1$ を取り直しておく.

- 以上の準備のもとで, 行列

$$\boldsymbol{A} - \lambda_1 \boldsymbol{v}_1 \boldsymbol{w}_1^H$$

を考える. これは以下を満たす.

  - $(\boldsymbol{A} - \lambda_1 \boldsymbol{v}_1 \boldsymbol{w}_1^H)\boldsymbol{v}_1 = 0$
  - $1 < j, 1 \leq m \leq \rho_j$ に対し,

$$(\boldsymbol{A} - \lambda_1 \boldsymbol{v}_1 \boldsymbol{w}_1^H)\boldsymbol{v}_{j,m} = \lambda_m \boldsymbol{v}_{j,m}.$$

- 以上からわかるように, $\boldsymbol{A} - \lambda_1 \boldsymbol{v}_1 \boldsymbol{w}_1^H$ の固有ベクトルは $\boldsymbol{A}$ の固有ベクトルと一致するが, 対応する固有値が,

$$\lambda_1, \lambda_2, \ldots, \lambda_k$$

から

$$0, \lambda_2, \ldots, \lambda_k$$

に変わる.

- したがって,

$$\boldsymbol{A} - \lambda_1 \boldsymbol{v}_1 \boldsymbol{w}_1^H$$

に冪乗法を適用することで, 固有値 λ_2 に対応する固有ベクトルを求めることができる.

- この手順を **Hotelling の減次** と呼ぶ.

- Hotelling 減次は, $\boldsymbol{A}$ の固有ベクトルを集めたものが基底となっているという条件のもとで, 更に

$$|\lambda_1| > \cdots > |\lambda_k|$$

で, λ_1 から λ_{k-1} までの固有値に対応する固有空間の次元が 1 であれば, 固有値 λ_k までは繰り返し適用可能である.

- したがって, $\boldsymbol{A}$ が n 個の相異なる固有値

$$\lambda_1, \ldots, \lambda_n$$

を持ち, 適切に並べ直すことで

$$|\lambda_1| > |\lambda_1| > \cdots |\lambda_n|$$

となる場合には, Hotelling の減次を繰り返すことで, $\boldsymbol{A}$ のすべての固有値を求めることができる.

QR 法

- 小規模な行列の固有値の計算に用いられる代表的な方法が QR 法である.
- QR 法の説明に入る前に, いくつか言葉の準備をしておく.

- n 次正方行列 $\boldsymbol{Q}$ の列ベクトルが正規直交基底をなすとき, すなわち $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$ となるとき, $\boldsymbol{Q}$ を直交行列という.

- n 次正方行列 $\boldsymbol{Q}$ の列ベクトルが複素内積の意味で正規直交規定をなすとき, すなわち $\boldsymbol{Q}^H\boldsymbol{Q} = \boldsymbol{I}$ となるとき, $\boldsymbol{Q}$ をユニタリ行列という. ここに, $\boldsymbol{Q}^H$ は, $\boldsymbol{Q}$ の Hermite 転置 (転置してから各要素の共役複素数を取ったもの) である.
- QR 法の基礎となるのは, 次の定理であり, そこで得られる $\boldsymbol{A}$ の分解表現を, $\boldsymbol{A}$ の **Schur 分解** と呼ぶ.

- **定理 (Schur 分解):**　複素行列 $\boldsymbol{A}$ は, 適切なユニタリ行列 $\boldsymbol{U}$ を取ることにより,

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^H$$

という形に変換できる. ただし, $\boldsymbol{S}$ は上三角行列である ($\boldsymbol{A}$ は対角化可能でなくてもよい).

- 証明 (以下 8 ページ)
  - ▷ 行列 $\boldsymbol{A}$ の次元 n に関する帰納法による.
  - ▷ $n = 1$ のときには, 何もしなくても $\boldsymbol{A}$ はすでにこの形になっている (1 次元なので $\boldsymbol{U} = 1$ でよい).
  - ▷ $n - 1$ 次まで定理の主張が示されていると仮定し, n 次でも定理の主張が成り立つことを示す.

$\boldsymbol{v}$ を $\boldsymbol{A}$ のある固有値 λ に対応する固有ベクトルとし, $\boldsymbol{v}$ を正規化してベクトル $\boldsymbol{v}_1$ を作る. そして, $\boldsymbol{v}_1$ を含む正規直交基底 $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ を構成し

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_n \end{pmatrix}$$

とする. $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ が正規直交基底だから, $\boldsymbol{U}$ はユニタリ行列である.

$\boldsymbol{AU}$ は,

$$\boldsymbol{AU} = \begin{pmatrix} \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_n \end{pmatrix} \begin{pmatrix} \lambda & * \\ \boldsymbol{0}_{n-1} & \boldsymbol{A}_0 \end{pmatrix}$$

という形に整理できる．ただし, $\boldsymbol{A}_0$ は該当する成分を集めた $n-1$ 次の正方行列である． 記号 $*$ の部分も同様であるが, 重要でないので省略してある.

よって,

$$\boldsymbol{U}^H \boldsymbol{A} \boldsymbol{U} = \begin{pmatrix} \lambda & * \\ \boldsymbol{0}_{n-1} & \boldsymbol{A}_0 \end{pmatrix}$$

である.

$\boldsymbol{A}_0$ に帰納法の仮定を適用すると, あるユニタリ行列 $\boldsymbol{U}_0$ と上三角行列 $\boldsymbol{S}_0$ により,

$$\boldsymbol{A}_0 = \boldsymbol{U}_0 \boldsymbol{S}_0 \boldsymbol{U}_0^H$$

となる.

したがって,

$$\boldsymbol{U}^H \boldsymbol{A} \boldsymbol{U} = \begin{pmatrix} \lambda & * \\ \boldsymbol{0}_{n-1} & \boldsymbol{U}_0 \boldsymbol{S}_0 \boldsymbol{U}_0^H \end{pmatrix}$$

と書ける.

$$\boldsymbol{U}_1 = \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{U}_0 \end{pmatrix}$$

と定義すると, $\boldsymbol{U}_1$ はユニタリ行列である. したがって, $\boldsymbol{U}\boldsymbol{U}_1$ も ユニタリ行列である.

$$\boldsymbol{U}_1^H \boldsymbol{U}^H \boldsymbol{A} \boldsymbol{U} \boldsymbol{U}_1 = \begin{pmatrix} \lambda & * \\ \boldsymbol{0}_{n-1} & \boldsymbol{S}_0 \end{pmatrix}$$

となり, $\begin{pmatrix} \lambda & * \\ \boldsymbol{0}_{n-1} & \boldsymbol{S}_0 \end{pmatrix}$ は上三角行列なので, 次元が n の場合も定理の主張が n の場合にも示された. ■

- $\det \boldsymbol{U}(s\boldsymbol{I} - \boldsymbol{A})\boldsymbol{U}^H = \det(s\boldsymbol{I} - \boldsymbol{A})$ だから, Schur 分解は 固有値を変えない. また, 上三角行列では, 行列の固有値は対角線上に並ぶ.
- よって, 数学的には, $\boldsymbol{A}$ が Schur 分解できた時点で, $\boldsymbol{A}$ のすべての固有値が求められたことになる.

- 一方, 数値計算という観点では, Schur 分解は行列の固有値と固有ベクトルを必要とするので, 問題はまだ解けたわけではない.

- QR 法とは, 行列 $\boldsymbol{A}$ の Schur 分解を近似的に求める方法で ([杉原, 室田]), 行列の QR 分解に基づいている.
- まず QR 分解について述べる.

- QR 分解とは, 行列 $\boldsymbol{A}$ を ユニタリ行列と上三角行列の積で表現することをいう.

- **定理:** 任意の行列はQR分解できる.

- **証明 (以下 8 ページ):** 行列 $\boldsymbol{A}$ の次数 n に関する帰納法による.

  - ▷ $n = 1$ のときには, 証明すべきことは何もない.
  - ▷ $n-1$ 次まで主張が正しいと仮定して, n 次行列についても主張が正しいことを示す.

行列 $\boldsymbol{A}$ の第 j 列ベクトルを $\boldsymbol{a}_j$ とする.
方程式

$$\boldsymbol{a}_1^H \boldsymbol{w} = 0$$

は少なくとも $n-1$ 個の 1 次独立な解を持つが, これらを直交化して,

$$\boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$$

を作る.

これに, $\boldsymbol{w}_1$ を,

$$\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n\}$$

が正規直交基底となるよう付け加え,

$$\boldsymbol{Q}_0 = \begin{pmatrix} \boldsymbol{w}_1 & \boldsymbol{w}_2 & \cdots & \boldsymbol{w}_n \end{pmatrix}$$

とする.

$\boldsymbol{Q}_0$ は ユニタリ行列であり, $j > 1$ に対して $\boldsymbol{a}_1^H \boldsymbol{w} = 0$ だから,

$$\boldsymbol{Q}_0^H \boldsymbol{A} = \begin{pmatrix} \alpha_n & \boldsymbol{\beta}_n \\ \boldsymbol{0} & \boldsymbol{A}_{n-1} \end{pmatrix}$$

という形になる. ただし, α_n はスカラー, $\boldsymbol{\beta}_n$ は $n-1$ 次のベクトル, $\boldsymbol{A}_{n-1}$ は $n-1$ 次の正方行列である.

帰納法の仮定より,

$$\boldsymbol{A}_{n-1} = \boldsymbol{Q}_{n-1}\boldsymbol{R}_{n-1}$$

で, $\boldsymbol{Q}_{n-1}$ はユニタリ行列, $\boldsymbol{R}_{n-1}$ は上三角行列である.

$$\boldsymbol{Q}_n = \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{Q}_{n-1} \end{pmatrix}$$

とする.

$\boldsymbol{Q}_n$ はユニタリ行列で,

$$\boldsymbol{Q}_n \begin{pmatrix} \alpha_n & \boldsymbol{\beta}_n \\ \boldsymbol{0} & \boldsymbol{R}_{n-1} \end{pmatrix} = \begin{pmatrix} \alpha_n & \boldsymbol{\beta}_n \\ \boldsymbol{0} & \boldsymbol{Q}_{n-1}\boldsymbol{R}_{n-1} \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_n & \boldsymbol{\beta}_n \\ \boldsymbol{0} & \boldsymbol{A}_{n-1} \end{pmatrix}$$

となる.

$$\boldsymbol{R} = \begin{pmatrix} \alpha_n & \boldsymbol{\beta}_n \\ \boldsymbol{0} & \boldsymbol{R}_{n-1} \end{pmatrix}$$

とすると, $\boldsymbol{R}$ は上三角行列で,

$$\boldsymbol{Q}_0^H \boldsymbol{A} = \boldsymbol{Q}_n \boldsymbol{R}$$

だったから,

$$\boldsymbol{A} = \boldsymbol{Q}_0 \boldsymbol{Q}_n \boldsymbol{R}.$$

$$\boldsymbol{Q} = \boldsymbol{Q}_0 \boldsymbol{Q}_n$$

とおくと, $\boldsymbol{Q}$ はユニタリ行列で,

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$$

である.

■

- QR 分解が存在することに証明の核心部分は, Gram-Schmidt の直交化によって正規直交基底を得る部分だった.
- Gram-Schmidt の直交化は代数的な演算で, 有限回の計算で数値計算の誤差を除き厳密解を得ることができるから, 上述の QR 分解の手順をそのままプログラムとして書き下すこともできるが…

- Gram-Schmidt の直交化は数値計算の誤差に弱いので, これがこのままの形で数値計算に用いられることは稀.
- 実用的な方法として 修正 Gram-Schmidt 法, Housholder 変換や Givens 変換による方法が知られているが, 詳細は略す. 興味がある者は [杉原, 室田] pp. 240–244 を参照せよ.

- QR 法のアルゴリズムは極めて単純なものであり, 以下のように書き下せる ($k = 0, 1, 2, \ldots$)

- **QR 法のアルゴリズム**
  - ▷ (初期化)
    $\boldsymbol{A}_0 = \boldsymbol{A}$ とする
  - ▷ (ループ)
    - ◇ $\boldsymbol{A}_k$ を $\boldsymbol{A}_k = \boldsymbol{Q}_k \boldsymbol{R}_k$ のように QR 分解する
    - ◇ $\boldsymbol{A}_{k+1} = \boldsymbol{R}_k \boldsymbol{Q}_k$ とおく.

- QR 法のアルゴリズムは単純であるが, その解析は極めて繁雑.
- この講義では詳細は述べないが, 一定の条件のもとで, $\boldsymbol{A}_k$ は上三角行列に漸近し, その対角要素に $\boldsymbol{A}$ の固有値が並ぶことが示される (詳細はたとえば [斎藤]).

- QR 法は, そのままで使用すると収束性が悪いので, $\boldsymbol{A}$ を Hessenberg 形と呼ばれる形の行列に変換してから QR 法を適用するのが一般的である詳細はたとえば ([杉原, 室田]).

Scilab による実行例

- Scilab では固有値を求めるために `spec` という組み込み関数を用いる.
- `A` を行列としたとき, `spec(A)` により固有値が求められる.

- `[M,D]=spec(A)` とすると (結果を受けとる変数の名前は任意), `D` に `A` を対角化した結果 (対角要素に固有値が並んだ行列), `M` に対角化のための正則行列 (固有ベクトルを並べた行列) が返される.

$$\mathtt{M}^{-1}\mathtt{A} * \mathtt{M} = \mathtt{D}$$

である.

- Scilab には 行列を Jordan 標準形に変換する関数はない. より弱い機能であるが, `bdiag` という関数により行列を「ブロック対角化」することは可能.
- 以下, 実行例を挙げる.

- 固有値のみを求める:

```
--> A=[4 1;1 4];
--> spec(A)
 ans  =
    3.
    5.
```

- $\boldsymbol{A} = \begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix}$ の固有値は 3 と 5 で, 対応する正規化された固有ベクトルは $(1/\sqrt{2}, -1/\sqrt{2})^T$ と $(1/\sqrt{2}, 1/\sqrt{2})^T$ であるが, 確かに固有値が求められている.
- $1/\sqrt{2} \simeq 0.7071068$ に注意.

- 固有値と固有ベクトルを求める:

```
--> [M,D]=spec(A)
 M  =
  -0.7071068    0.7071068
   0.7071068    0.7071068
 D  =
   3.    0.
   0.    5.
```

- M の第 1 列に固有値 3 に対応する固有ベクトルが, M の第 2 列に固有値 5 に対応する固有ベクトルが並んでいることが確認できる (固有値にはスカラー倍の不定性があることに注意).

- 次に, この行列に対して冪乗法と逆冪乗法を試す. 初期値は $(1,0)^T$ とする. どちらの固有値に対する固有ベクトルも $(1,0)^T$ に直交しないので, この初期値に対して冪乗法と逆冪乗法を適用すると, それぞれ数値計算の誤差および符号の不定性を除き $(1/\sqrt{2}, 1/\sqrt{2})^T$ と $(1/\sqrt{2}, -1/\sqrt{2})^T$ が得られる筈である.

- 冪乗法 (100 解繰り返し)

```
--> x=[1;0];
--> for i=1:100
  >    y=A*x;
  >    x=y/norm(y);
  > end
```

```
--> x
 x  =
    0.7071068
    0.7071068
```

- 確かに，固有値 5 に対応する固有ベクトル $(1/\sqrt{2}, 1/\sqrt{2})^T$ が得られている．

- 逆冪乗法 (100 解繰り返し)

```
--> x=[1;0];
--> for i=1:100
  >    y=A\x;
  >    x=y/norm(y);
  > end
```

```
--> x
 x  =
   0.7071068
  -0.7071068
```

- 確かに，固有値 3 に対応する固有ベクトル $(1/\sqrt{2}, -1/\sqrt{2})^T$ が得られている．

- 続いて QR 分解と QR 法を試す.
- Scilab では QR 分解のために qr という組み込み関数を用いる.
- A を行列としたとき, [Q,R]=qr(A) とすると, A を QR 分解したときの Q と R が得られる.

$$\mathtt{A} = \mathtt{Q} * \mathtt{R}$$

である.

- QR 分解

```
--> [Q,R]=qr(A)
 Q  =
  -0.9701425  -0.2425356
  -0.2425356   0.9701425
 R  =
  -4.1231056  -1.940285
   0.          3.6380344
```

- A = Q * R となっていることを確認する.

```
--> Q*R
 ans  =
   4.    1.
   1.    4.
```

- QR 法 (100 回繰り返し)

```
--> X=A;
--> for i=1:100
  > [Q,R]=qr(X);
  > X=R*Q;
  > end
```

```
--> X
 X  =
    5.            -1.044D-16
    1.307D-22      3.
```

- この例では A の対角化までなされており, 対角要素に A の固有値が並んでいることが確認できる.