情報数値解析

第４回

行列の分解と連立１次方程式

第４回・行列の分解と連立１次方程式

- ✔ 講義の目的:
  数値計算の多くの場面では，最終的に線形計算，特に連立１次方程式を解く必要があります．この講義では，解法の基本となる行列の分解と連立１次方程式の数値解法について紹介します．

- ✔ 参考文献:

  - ✘ G.H. Golub, C.F. Van Loan:
    Matrix Computations, 3rd Edition, The Johns Hopkins University Press, 1996.
  - ✘ T.A. Davis:
    Direct Methods for Sparse Linear Systems, SIAM, 2006.
  - ✘ 杉原 正顕, 室田 一雄:
    線形計算の数理, 岩波書店, 2009.

講義のポイント

1. 行列を分解する意味
2. 連立 1 次方程式の数値解法
   - ✔ 数値解法の紹介
     - ✘ なぜ数多くの解法があるのか？
     - ✘ 最近の動向
   - ✔ 計算する際の注意点
     - ✘ 手法の選択
     - ✘ ソフトウェアの入手方法
     - ✘ 結果の検証，発表のマナー

行列を分解する意味

以下，簡単のため，A は $n \times n$ の実行列を表わすとします．行列 A を

$$A = A_1 A_2 \cdots A_k$$

と分解できると，

- ✔ 連立１次方程式の解が求まる
- ✔ 固有値・固有ベクトルが求まる
- ✔ 特異値・特異ベクトルが求まる
- ✔ 最小２乗問題（正規方程式）が解ける
- ✔ 行列式の値が求まる
- ✔ 正則性・正定値性が判定できる
- ✔ （不完全な分解でも）反復解法の収束性を高めることが期待される

代表的な行列の分解

LU 分解　$PA = LU$

P: 置換行列，L: 下三角行列，U: 上三角行列

Choleski 分解　$A = LL^T$, $A = LDL^T$

A: 正定値対称行列，L: 下三角行列，D: 対角行列

QR 分解　$A = QR$

Q: 直交行列，R: 上三角行列

特異値分解　$A = U\Sigma V^T$

U, V: 直交行列，$\Sigma \geq \mathbf{0}$: 対角行列

実 Schur 分解　$A = QSQ^T$

Q: 直交行列，S: 準上三角行列

LU 分解の演算数（1/5）

```
! n should be greater than 1
do k = 1, n-1
  w = 1/a(k,k) ! set pivot
  do i = k+1, n
    a(i,k) = w*a(i,k)
    do j = k+1,n
      a(i,j) = a(i,j) - a(i,k)*a(k,j) ! LU decomposition
    end do
  end do
end do
```

LU 分解の演算数（1/5）

```
! n should be greater than 1
do k = 1, n-1
  w = 1 / a(k,k) ! set pivot
  do i = k+1, n
    a(i,k) = w * a(i,k)
    do j = k+1,n
      a(i,j) = a(i,j) - a(i,k) * a(k,j) ! LU decomposition
    end do
  end do
end do
```

LU 分解の演算数（2/5）

```
! n should be greater than 1
do k = 1, n-1
  w = 1 / a(k,k) ! set pivot
  do i = k+1, n
    a(i,k) = w*a(i,k)
    do j = k+1,n
      a(i,j) = a(i,j) - a(i,k)*a(k,j) ! LU decomposition
    end do
  end do
end do
```

✔ 除算 / は $k = 1, \ldots, n-1$ に対し $n-1$ 回実行

LU 分解の演算数（3/5）

```
! n should be greater than 1
do k = 1, n-1
  w = 1/a(k,k) ! set pivot
  do i = k+1, n
    a(i,k) = w * a(i,k)
    do j = k+1,n
      a(i,j) = a(i,j) - a(i,k)*a(k,j) ! LU decomposition
    end do
  end do
end do
```

- ✔ 乗算 * は $k = 1, \ldots, n-1$ と動くとき，それぞれ $(n-1)$，$(n-2)$，$\cdots$，2，1 回実行
- ✔ $1 + 2 + \cdots + (n-2) + (n-1) = \dfrac{n(n-1)}{2}$

LU 分解の演算数（4/5）

```
! n should be greater than 1
do k = 1, n-1
  w = 1/a(k,k) ! set pivot
  do i = k+1, n
    a(i,k) = w*a(i,k)
    do j = k+1,n
      a(i,j) = a(i,j) - a(i,k) * a(k,j) ! LU decomposition
    end do
  end do
end do
```

- ✔ 減算 - および乗算 * は $k = 1, \ldots, n-1$ と動くとき，i, j のループで $(n-1)^2$，$(n-2)^2$，$\cdots, 2^2$，1^2 回実行
- ✔ $1^2 + 2^2 + \cdots + (n-2)^2 + (n-1)^2 = \dfrac{(n-1)n(2n-1)}{6}$

公式

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

Since $(k+1)^3 - k^3 = 3k^2 + 3k + 1$,

$$\begin{aligned} 2^3 - 1^3 &= 3 \cdot 1^2 + 3 \cdot 1 + 1 && (k=1) \\ 3^3 - 2^3 &= 3 \cdot 2^2 + 3 \cdot 2 + 1 && (k=2) \\ &\cdots \\ n^3 - (n-1)^3 &= 3 \cdot (n-1)^2 + 3 \cdot (n-1) + 1 && (k=n-1) \\ (n+1)^3 - n^3 &= 3 \cdot n^2 + 3 \cdot n + 1 && (k=n) \end{aligned}$$

$$\Longrightarrow (n+1)^3 - 1^3 = 3\sum_{k=1}^{n} k^2 + 3\sum_{k=1}^{n} k + n.$$

LU 分解の演算数（5/5）

```
do k = 1, n-1
  w = 1 / a(k,k) ! set pivot
  do i = k+1, n
    a(i,k) = w * a(i,k)
    do j = k+1,n
      a(i,j) = a(i,j) - a(i,k) * a(k,j) ! LU decomposition
    end do
  end do
end do
```

- ✔ 乗除算
$$\frac{(n-1)n(2n-1)}{6} + \frac{n(n-1)}{2} + (n-1) = \frac{(n-1)(n^2+n+3)}{3}$$
- ✔ 減算 $\dfrac{(n-1)n(2n-1)}{6}$

行列の分解によって判ること [例]

- 特異値分解 $A = U\Sigma V^T$ における Σ の正の要素数は A の階数: rank(A) に一致する.
- 対称行列に対し Gauss の消去法を部分ピボット選択なしで LU 分解することができ，かつ結果の上三角行列の対角成分がすべて正の時，A は正定値.
- 部分ピボット選択付き LU 分解において行変換の回数を p とすると

$$|A| = (-1)^p |U|$$

- A の最大特異値を $\sigma_1(A)$ とおけば

$$\|A\|_2 = \sigma_1(A)$$

連立１次方程式とは？

$n \times n$ 行列 $A = [a_{ij}]$ とベクトル $\boldsymbol{b} = [b_1, \ldots, b_n]^T$ に対し

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

をみたすベクトル $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ を求める問題

- ✔ 理学・工学・農学・社会科学・人文科学のあらゆる分野に現れる
- ✔ 実際の科学技術分野における数値計算の計算時間の大半は連立１次方程式を解くことに費やされているといわれている

連立１次方程式との関わり合い

研究・業務内容によって様々

それ自体研究対象とする

- ✔ 新しいアルゴリズムの提案
- ✔ 性能評価

それ自体研究対象ではないが深く関わらざるを得ない

- ✔ 高速化（並列化・収束性向上）
- ✔ 収束証明，精度保証

それ自体研究対象ではない

- ✔ 道具として利用できればいい
- ✔ アルゴリズムはきちんと把握したい

それ自体研究対象ではない場合でも

ブラックボックスとして利用するのではなく，最低限，連立１次方程式を解く手法の名前は知っておくべき

《理由》

連立１次方程式は解きたい問題の最終段階，もしくは非常に重要な局面で登場することが多い
→ A と $\boldsymbol{b}$ に「情報」が集約されている
→ $A\boldsymbol{x} = \boldsymbol{b}$ を解き損ねると，すべてが台無し

連立 1 次方程式

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

- ✔ 解がないかもしれない

$$\begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \Rightarrow \quad 0 = 1 \text{ となり矛盾}$$

- ✔ 解が無数にあるかもしれない

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} s \\ 1 - s \end{bmatrix}$$

- ✔ A が**正則** ($AA^{-1} = A^{-1}A = I$ となる A^{-1} が存在) ならば解 $\boldsymbol{x} = A^{-1}\boldsymbol{b}$ は一意に存在
- ✔ A が特異または特異に近い場合は，特別な議論が必要

連立 1 次方程式の解法

- ✔ 紀元前二世紀『九章算術』第八章「方程」
- ✔ 1809 年 Gauss『天体運動論』消去法を紹介
- ✔ 1823 年 Gauss 手紙の中で反復法について言及
- ✔ 1952 年 Hestenes and Stiefel 共役勾配法を発表

日本では中学二年の数学で習う（連立方程式）

代入法: 方程式を 1 つの文字について解き，他の方程式に代入して 1 つの文字を消去

加減法: それぞれの方程式の両辺を何倍かして，たしたりひいたりして 1 つの文字を消去

連立方程式の利用(できるかな？)

A さんは午前 10 時に家を出発し，自転車に乗って時速 12km で走り，午前 11 時 30 分に目的地に着く予定であった．ところが，途中で自転車が故障したので，そこからは時速 4km で歩いた．そのため，目的地に着いたのは出発してから 2 時間後の正午であった．家から自転車が故障した地点までの道のりを求めなさい．

岐阜県高等学校入試問題
『チャート式 中 2 数学』数研出版

次数

方程式の次数（$n \times n$ の "n"）は問題によって異なる

- ✔ 2 次元 Poisson 方程式の数値解法の例題 121
- ✔ 定常 Navier-Stokes 方程式の精度保証 11,400
- ✔ 代数的マルチグリッド法による電磁界解析 160,464
- ✔ Stokes 方程式の有限要素近似解 1,410,479
- ✔ 原子炉圧力モデルの弾性構造解析 10,328,853

次数が大きくなると，すべてをメモリに格納することは困難
《例》 $10,000,000 \times 10,000,000$ の行列を倍精度で宣言

$$8 \times 10,000,000 \times 10,000,000 \approx 800,000 \text{ GByte}$$

数値計算の手段

プログラム言語

大規模計算，数値計算ライブラリを含む資産が豊富

- ✔ Fortran
- ✔ C, C++
- ✔ Java

数値計算システム

手軽に利用できる．充実した機能

- ✔ MATLAB http://www.mathworks.co.jp/products/matlab/
- ✔ Scilab http://www.scilab.org/
- ✔ Octave http://www.octave.org/
- ✔ Mathematica, Maple, FreeFem++, etc.

プログラムに求められること

- ✔ 可読性 (readability)
- ✔ 移植性 (portability)
- ✔ 高精度 (high-accuracy)
- ✔ 頑強性 (robustness)
- ✔ 高性能 (high-performance)

長谷川 秀彦: “数値計算ライブラリ”
『これだけは知っておきたい 数学ツール』共立出版

数値計算ソフトウェアの調査方法

- ✔ Netlib
  - ✘ 高性能計算及び並列処理に関する研究活動, ソフトウェア, データベースなどの情報を無償で提供
  - ✘ `http://www.netlib.org/`
- ✔ Parallel and High Performance Applicational Software Exchange
  - ✘ Netlib の国内版
  - ✘ `http://phase.hpcc.jp/`
- ✔ Guide to Available Mathematical Software
  - ✘ ソフトウェアパッケージのデータベース
  - ✘ `http://gams.nist.gov/`

著名な線形計算ソフトウェア

- ✔ LAPACK http://www.netlib.org/lapack/
- ✔ Templates http://www.netlib.org/templates/
- ✔ NAG http://www.nag-j.co.jp/
- ✔ IMSL http://www.vnij.com/products/imsl/

....

- ✔ NUMPAC http://netnumpac.fuis.fukui-u.ac.jp/numpac/
- ✔ Super Matrix Solver

  http://www.v-t.jp/products/hpc/software/library/sms_top_main.html

....

- ✔ Numerical Recipes http://www.nr.com/

連立 1 次方程式の数値解法の分類

- ✔ 直接法
  - ✘ Gauss の消去法，Cholesly 分解, etc.
- ✔ 反復法
  - ✘ 定常反復法
    - ✓ SOR 法，Gauss-Seidel 法, etc.
  - ✘ 非定常反復法
    - ✓ CG 法，BiCG 法, GMRES 法, etc.
- ✔ 直接法と反復法の組み合わせ
  - ✘ 解の反復改良，不動点定理に基づく解の精度保証, etc.

直接法

ほとんどが行列の分解に基づく手法

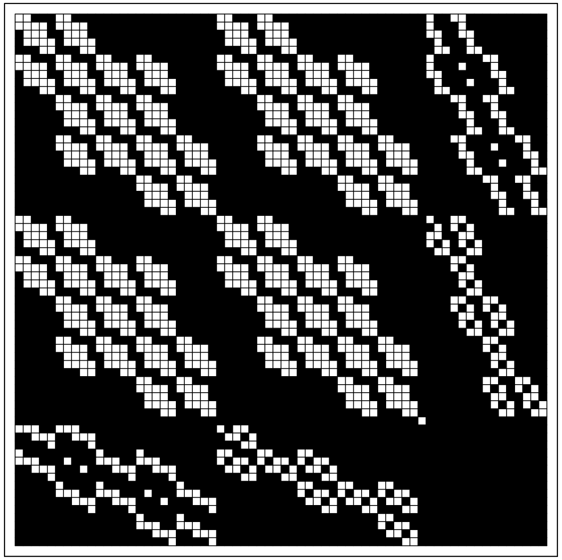
- ✔ LU 分解

$$A = LU, \quad L\text{: 下三角行列}, \quad U\text{: 上三角行列}$$

- ✔ A が正則 $\Rightarrow$ A が適当な列変換により LU 分解可能
- ✔ 多くの数値計算ソフトウェアで利用可能
- ✔ 行列に特異性がない限り（経験的に）安定に解ける
- ✔ 一般の行列で次数が多くない場合には迷わず選択！
  - ✘ 自分でプログラムを組む場合は，正規化とピボット選択に注意

直接法の問題点

1. LU 分解の計算量は $O(n^3)$　(n は次数)
   $\Rightarrow$ 次数が大きくなると計算時間が膨大になる
2. 分解を行なうと行列の疎な構造が崩れる
   $\Rightarrow$ より多くのメモリ空間が必要

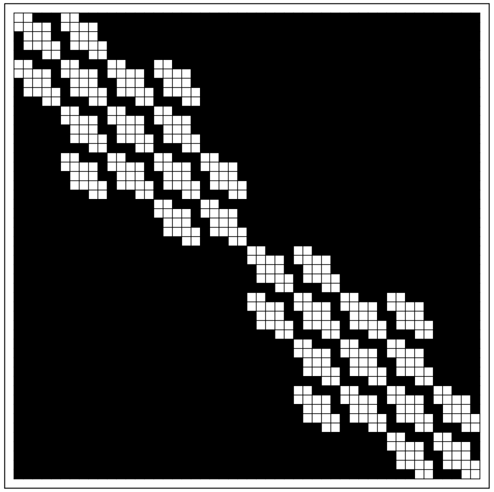


LU 分解前

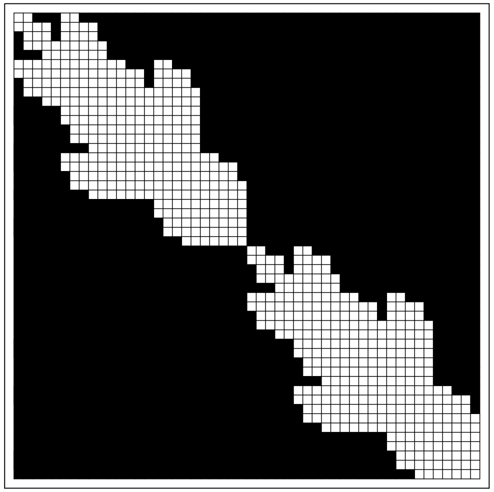
LU 分解後

疎行列構造の維持

行列が帯行列の場合，構造が比較的維持される



LU 分解前



LU 分解後

分解によるゼロ要素の損失（fill-in）を抑える研究

- ✔ reverse Cuthill-McKee algorithm
- ✔ approximate minimum degree algorithm
- ✔ graph-partitioning based ordering

反復法

真の解に収束していく近似解の列を逐次作成していく手法

$$\boldsymbol{x}^0 \to \boldsymbol{x}^1 \to \boldsymbol{x}^2 \to \cdots \to \boldsymbol{x}$$

- ✔ **定常反復法**
  一定（定常）の処理を繰り返すことによって近似解を作成する方法
  $$\boldsymbol{x}^k = \boldsymbol{x}^{k-1} + R(\boldsymbol{b} - A\boldsymbol{x}^{k-1}), \quad k \geq 1$$
- ✔ **非定常反復法**
  各反復ごとに変化する情報を取り込みながら計算を進める．多くは反復によって直交するベクトル列を作り出す
  $$\boldsymbol{x}^k \in \mathsf{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0\}$$

反復法の特徴

メモリの節約:　行列を分解する必要がないため，疎行列の構造を保つことができる（格納方法は様々）

$$A = \begin{bmatrix} 0 & 0 & 0 & 5 \\ 0 & 2 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 0 & 4 & 0 \end{bmatrix} \quad \Rightarrow \quad S = \begin{matrix} (3,1) & 1 \\ (2,2) & 2 \\ (3,2) & 3 \\ (4,3) & 4 \\ (1,4) & 5 \end{matrix}$$

高速:　基本の計算は「行列×ベクトル」と内積計算のため，収束が速ければ計算量が $O(n^2)$ になることが期待される

共役勾配法のアルゴリズム

set an initial vector $\boldsymbol{x}_0$; $\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$; $\boldsymbol{p}_0 := \boldsymbol{r}_0$;

for $k := 0, 1, \cdots$ until $\boldsymbol{r}_k = \boldsymbol{0}$ do

begin

$$\alpha_k := \frac{(\boldsymbol{r}_k, \boldsymbol{p}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)};$$

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k;$$

$$\boldsymbol{r}_{k+1} := \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k;$$

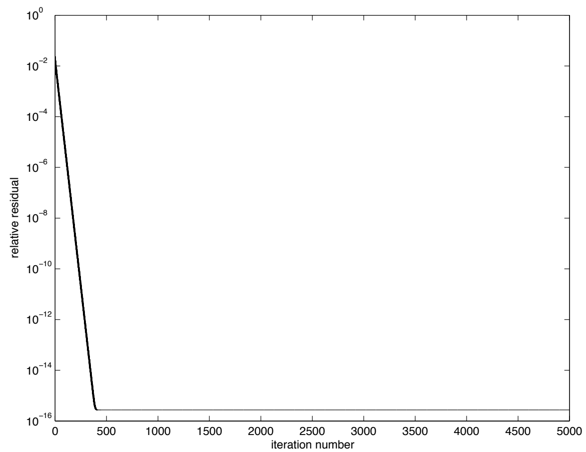
$$\beta_k := -\frac{(\boldsymbol{r}_{k+1}, A\boldsymbol{p}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)};$$

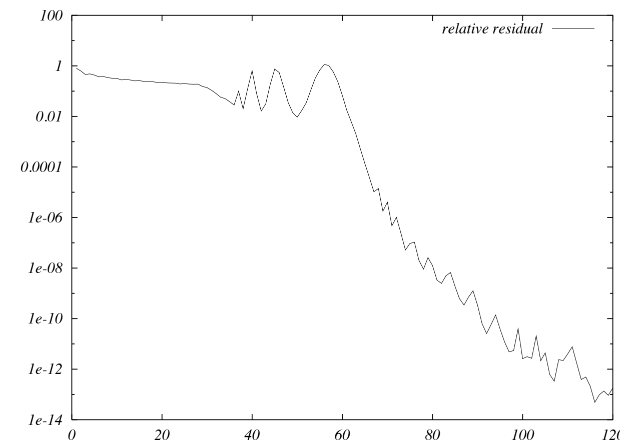
$$\boldsymbol{p}_{k+1} := \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$$

end

反復法の問題点

- ✔ 行列の性質（固有値・特異値など）に大きく依存する
  - ✘ 初期値 $\boldsymbol{x}_0$ によって収束特性が変わることがある
  - ✘ 右辺 $\boldsymbol{b}$ によって収束特性が変わることがある
  - ✘ 非定常反復法は丸め誤差の影響を受けやすい



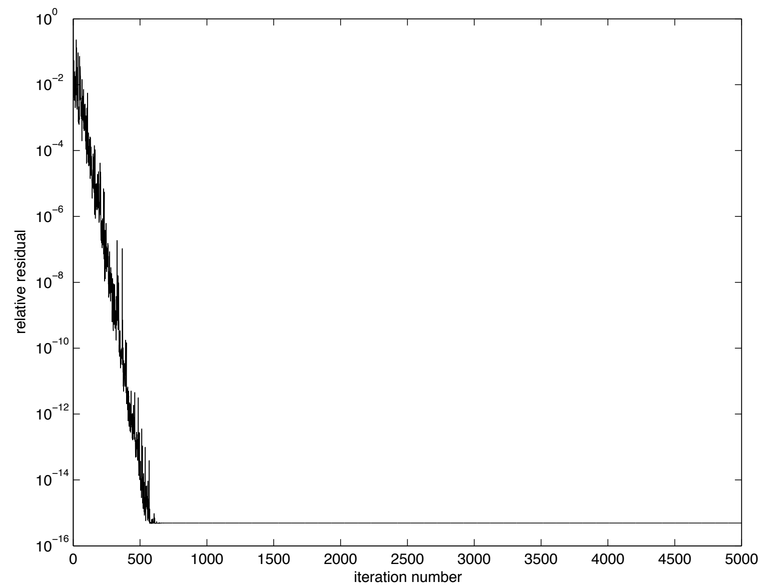


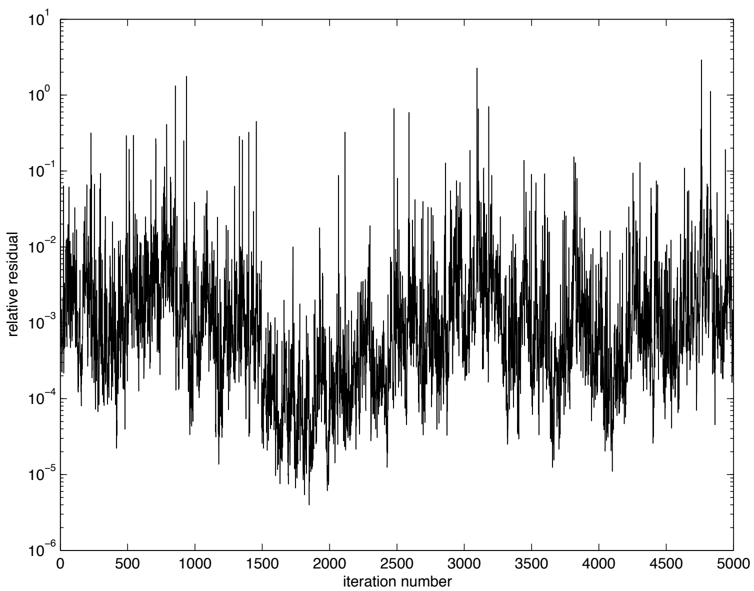
- ✔ 主要な非定常反復法については「最適な手法はない」という研究結果がある
  “How fast are nonsymmetric matrix iterations?”
  SIAM Journal on Matrix Analysis and Applications, Vol.13, 1992, 778-795.

反復法の収束判定基準

最大反復回数と停止条件の設定が必要

$$（例）\|A\boldsymbol{x}^k - \boldsymbol{b}\|/\|\boldsymbol{b}\| < \varepsilon$$





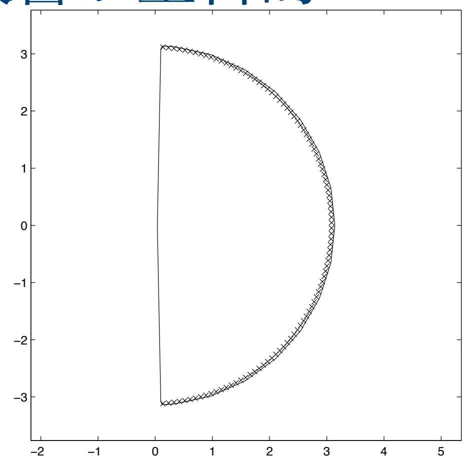
✔ "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition", SIAM, Philadelphia, PA, 1994. http://www.netlib.org/templates/Templates.html

前処理

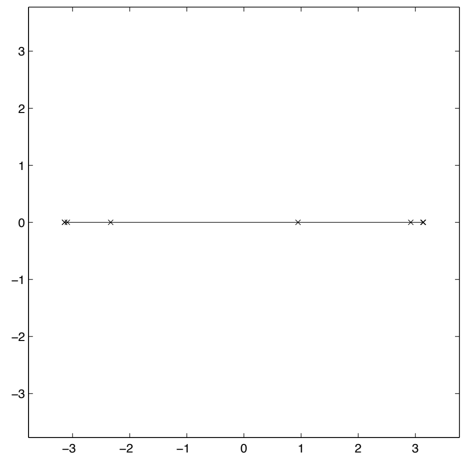
- 特に反復解法では（実質）不可欠

$$A\boldsymbol{x} = \boldsymbol{b} \quad \Leftrightarrow \quad \hat{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$$

- 固有値分布の改善が主目的



前処理前の固有値分布

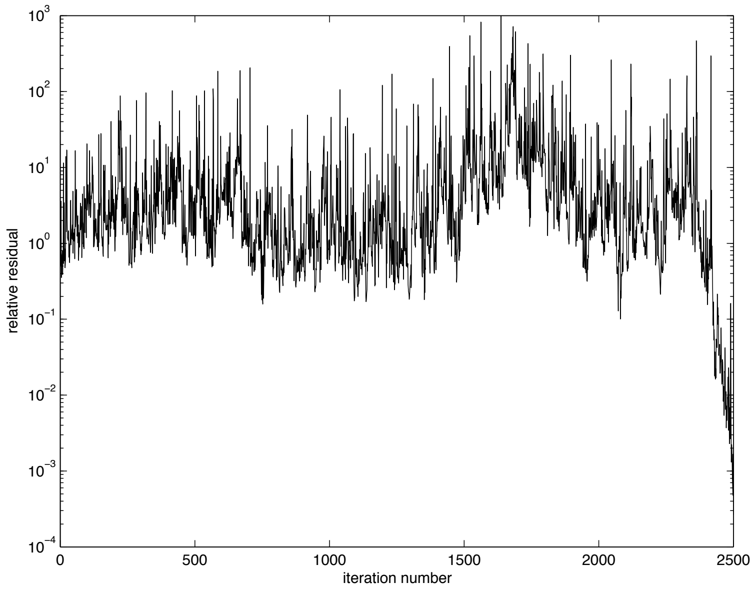


前処理後の固有値分布

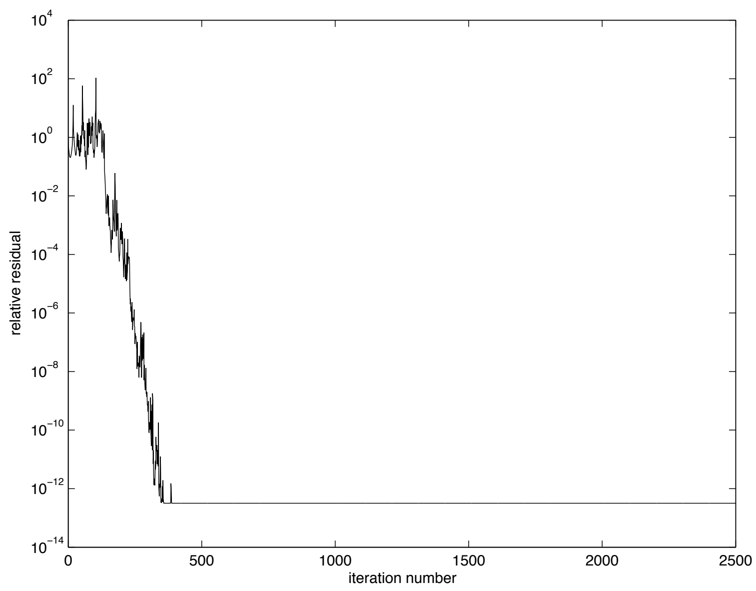
- 丸め誤算の影響を少なくしたり，疎行列の構造を保つためにも前処理が用いられる

代表的な前処理技法

- 不完全 LU 分解（不完全 Cholesky 分解）
- 近似逆行列
- 正規化（スケーリング）



前処理前



前処理後

手法の選択基準

1. 行列の性質を把握
   → 疎，バンド，対称，正定値，M 行列, etc.
2. 欲しい計算サイズがどれくらいか
3. （可能なら）固有値分布を可視化
4. 複数の解法を試す
5. 検算を行なう
   → （例）$\boldsymbol{x}$ を適当に決め，$\boldsymbol{b} := A\boldsymbol{x}$ で右辺を作成

《テスト行列集》

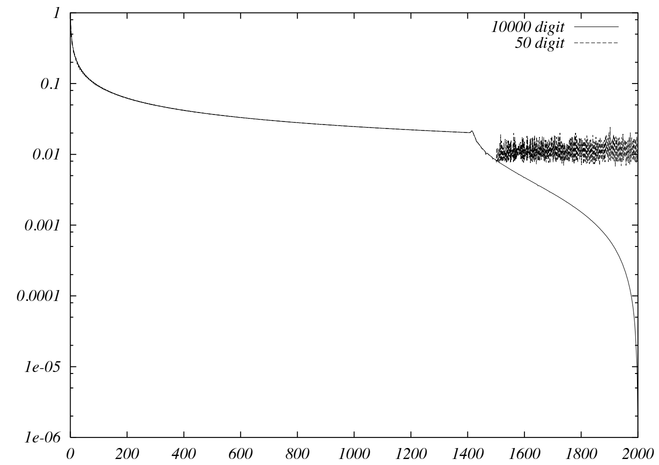
- ✔ "A collection of matrices for testing computational algorithms," John Wiley & Sons, 1969.
- ✔ Matrix Market `http://math.nist.gov/MatrixMarket/`
- ✔ Matrix Workshop `http://phase.hpcc.jp/MatrixWorkshop/`

解法の自作を目指す方へ

- ✔ 必ず複数の計算機環境で試す
  - ✘ 計算機構造，コンパイラ (含む並列ライブラリ)，オプションによって性能差が大きい場合がある
  - ✘ 文法違反などのデバッグにきわめて有効
- ✔ 有料ソフトウェアは速い！
  - ✘ チューニングされた MATLAB, LAPACK などに勝つのは至難の技
  - ✘  速ければすごい

多倍長演算

特に反復法の収束特性を調べる比較の手段として，4 倍精度やそれ以上の多倍長演算も一考を



- ✔ GMP `http://www.swox.com/gmp/`
- ✔ Omni `http://phase.hpcc.jp/Omni/Omni-doc/f77QReal.html`
- ✔ FMLIB `http://www.lmu.edu/acad/personal/faculty/dmsmith2/FMLIB.html`
- ✔ Exflib `http://www-an.acs.i.kyoto-u.ac.jp/~fujiwara/exflib/`

数値計算のマナー

- ✔ 倍精度を使う
  - ✘ 単精度は丸め誤算の影響大
  - ✘ 一変数に必要な記憶容量が倍になるなどの理由で速度は若干遅くなるかも
- ✔ 異なる環境で試す
  - ✘ 複数の入力データ
  - ✘ 異なる計算機，異なる OS，コンパイラ
- ✔ 発表の際は数値計算環境を明示
  - ✘ 計算機名，チップ（CPU）名，OS 名
  - ✘ 数値計算ソフト名，コンパイラ名, etc.

答え合わせ

$$\begin{bmatrix} 1 & 1 \\ \frac{1}{12} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 18 \\ 2 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 1 \\ 12 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 18 \end{bmatrix}$$