クレジット:
Mathematics and Informatics Center 計算機実験I 2020 藤堂眞治

本講義資料内に掲載している外部へのリンク先の著作物の利用に関しては、リンク先のそれぞれの権利者の定めるところに従ってください。

計算機実験 I (第 5 回)

藤堂眞治

2020/05/27

講義予定

- 2020-04-22: 第 1 回 環境整備
- 2020-05-07: 第 2 回 計算機実験の基礎
- 2020-05-13: 第 3 回 常微分方程式の解法
- 2020-05-20: 第 4 回 行列演算とライブラリ
- 2020-05-27: 第 5 回 連立一次方程式の解法
- 2020-06-03: 第 6 回 行列の対角化
- 2020-06-10: 第 7 回 疎行列に対する反復解法
- 2020-06-17: 第 8 回 特異値分解と最小二乗法

物理の問題にあらわれる行列演算

- 連立一次方程式・逆行列
  - 偏微分方程式の境界値問題
  - 非線形連立方程式に対するニュートン法
- 対角化・特異値分解
  - 固有値問題・行列関数
  - 最小二乗近似
- 計算機は大規模行列演算が得意
  - 直接法: $\sim 10^4$ 次元
  - 疎行列に対する反復解法: $\sim 10^9$ 次元
- 行列演算についてはライブラリがよく整備されている
  - それぞれの原理とその特徴を理解して正しく使うことが重要
  - 適切なライブラリを使うことで数十倍あるいはそれ以上速くなることも

ポアソン方程式の境界値問題

- 二次元ポアソン方程式

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y) \qquad 0 \le x \le 1,\ 0 \le y \le 1$$

- ディリクレ型境界条件: $u(x,y) = g(x,y)$ on $\partial\Omega$
- 有限差分法により離散化
  - x 方向、y 方向をそれぞれ n 等分: $(x_i, y_j) = (i/n, j/n)$
  - $(n+1)^2$ 個の格子点の上で $u(x_i, y_j) = u_{ij}$ が定義される
  - そのうち $4n$ 個の値は境界条件で定まる
  - ポアソン方程式を中心差分で近似 $(h = 1/n)$

$$\frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{h^2} = f_{ij}$$

残り $(n-1)^2$ 個の未知数に対する連立一次方程式

ポアソン方程式の境界値問題

- ノイマン型境界条件の場合
  - 境界上で $u(x,y)$ の微分が定義される。
  - 例) $\partial u(0,y)/\partial x = h(0,y)$
- 境界条件を差分近似で表す

$$\frac{u_{1j} - u_{0j}}{h} = h_{0j} \qquad j = 1 \cdots (n-1)$$

$(n+1)^2 - 4$ 個の未知数に対して、ポアソン方程式の差分近似とあわせて、合計 $(n-1)^2 + 4(n-1) = (n+1)^2 - 4$ 個の連立一次方程式

- 二次元グリッド上の点 (i,j) と長さ $(n+1)^2$ のベクトルの要素との対応関係をきちんと定義することが大事

逆行列の「間違った」求め方

- 線形代数の教科書に載っている公式

$$A^{-1} = \frac{\tilde{A}}{|A|}$$

  $|A|$: A の行列式、$\tilde{A}$: A の余因子行列
- $n \times n$ 行列の行列式や余因子を定義通り計算すると、計算量〜$O(n!)$
- したがって、上の方法で逆行列を計算すると、計算量〜$O(n!)$
- $n = 100$ の場合: $n! \approx 9.3 \times 10^{157}$

逆行列の「正しい」求め方

- 連立一次方程式 $A\mathbf{x} = \mathbf{e}_j$ を全ての $\mathbf{e}_j$ について解く
  - Gauss の消去法による連立一次方程式の解法: 計算量〜$O(n^3)$
  - Gauss の消去法の途中で出てくる下三角行列 (L) と上三角行列 (U) 行列を再利用 (LU 分解) すれば、逆行列全体を求めるための計算量も $O(n^3)$
  - $n = 100$ の場合: $n^3 = 10^6 \ll 9.3 \times 10^{157}$
- A の行列式も $O(n^3)$ で計算可

- 逆行列をベクトルに掛ける必要がある場合には、逆行列を明示的に求めるのではなく連立一次方程式を解く

$$\mathbf{x} = A^{-1}\mathbf{b} \quad \Rightarrow \quad A\mathbf{x} = \mathbf{b}$$

ガウスの消去法

- 解くべき連立方程式

$$
\begin{aligned}
a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\
a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \cdots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\
a_{31}^{(1)}x_1 + a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \cdots + a_{3n}^{(1)}x_n &= b_3^{(1)} \\
&\cdots \\
a_{n1}^{(1)}x_1 + a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \cdots + a_{nn}^{(1)}x_n &= b_n^{(1)}
\end{aligned}
$$

- ある行を定数倍しても、方程式の解は変わらない
- ある行の定数倍を他の行から引いても、方程式の解は変わらない

ガウスの消去法

- 1 行目を $m_{i1} = a_{i1}^{(1)}/a_{11}^{(1)}$ 倍して、i 行目 $(i \geq 2)$ から引く

$$
\begin{aligned}
a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\
a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 + \cdots + a_{2n}^{(2)}x_n &= b_2^{(2)} \\
a_{32}^{(2)}x_2 + a_{33}^{(2)}x_3 + \cdots + a_{3n}^{(2)}x_n &= b_3^{(2)} \\
&\cdots \\
a_{n2}^{(2)}x_2 + a_{n3}^{(2)}x_3 + \cdots + a_{nn}^{(2)}x_n &= b_n^{(2)}
\end{aligned}
$$

- ここで

$$
\begin{aligned}
a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} \qquad i \geq 2, j \geq 2 \\
b_i^{(2)} &= b_i^{(1)} - m_{i1}b_1^{(1)} \qquad i \geq 2
\end{aligned}
$$

ガウスの消去法

- 2 行目を $m_{i2} = a_{i2}^{(2)}/a_{22}^{(2)}$ 倍して、i 行目 $(i \geq 3)$ から引く

$$
\begin{aligned}
a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\
a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 + \cdots + a_{2n}^{(2)}x_n &= b_2^{(2)} \\
a_{33}^{(3)}x_3 + \cdots + a_{3n}^{(3)}x_n &= b_3^{(3)} \\
&\cdots \\
a_{n3}^{(3)}x_3 + \cdots + a_{nn}^{(3)}x_n &= b_n^{(3)}
\end{aligned}
$$

- ここで

$$
\begin{aligned}
a_{ij}^{(3)} &= a_{ij}^{(2)} - m_{i2}a_{2j}^{(2)} \qquad i \geq 3, j \geq 3 \\
b_i^{(3)} &= b_i^{(2)} - m_{i2}b_2^{(2)} \qquad i \geq 3
\end{aligned}
$$

ガウスの消去法

- 最終的には、左辺が右上三角形をした連立方程式となる

$$
\begin{aligned}
a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\
a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 + \cdots + a_{2n}^{(2)}x_n &= b_2^{(2)} \\
a_{33}^{(3)}x_3 + \cdots + a_{3n}^{(3)}x_n &= b_3^{(3)} \\
&\cdots \\
a_{n-1,n-1}^{(n-1)}x_{n-1} + a_{n-1,n}^{(n-1)}x_n &= b_{n-1}^{(n-1)} \\
a_{nn}^{(n)}x_n &= b_n^{(n)}
\end{aligned}
$$

- これを下から順に解いていけばよい (後退代入)

練習問題

- 次の連立方程式をガウスの消去法で (手で) 解け

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 18 \\ 24 \\ 31 \end{bmatrix}$$

ガウスの消去法のコード

```
for (k = 0; k < n; ++k) {
  for (i = k + 1; i < n; ++i) {
    for (j = k + 1; j < n; ++j) {
      mat_elem(a,i,j) -= mat_elem(a,k,j) * mat_elem(a,i,k) /
          mat_elem(a,k,k);
    }
    b[i] -= b[k] * mat_elem(a,i,k) / mat_elem(a,k,k);
  }
}
for (k = n-1; k >= 0; --k) {
  for (j = k + 1; j < n; ++j) {
    b[k] -= mat_elem(a,k,j) * b[j];
  }
  b[k] /= mat_elem(a,k,k);
}
```

- C 言語では配列の添字が 0 から始まることに注意
- mat_elem(a,i,j) は行列 a の (i,j) 成分を表す (cmatrix.h 中で定義)
- サンプルコード: gauss.c
- 入力ファイル: input1.dat

ピボット選択

- ガウスの消去法の途中で $a_{kk}^{(k)}$ が零になると、計算を先に進めることができなくなる
- 行を入れ替えても、方程式の解は変わらない $\Rightarrow$ k 行以降で、$a_{ik}^{(k)}$ が非零の行と入れ替える (ピボット選択)
- 実際のコードでは、情報落ちを防ぐため、$a_{kk}^{(k)}$ が零でない場合でも、$a_{ik}^{(k)}$ の絶対値が最大の行と入れ替える
- ピボット選択が必要となる例: input2.dat

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 8 & 5 \\ 3 & 6 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 30 \\ 33 \\ 45 \end{bmatrix}$$

- 行列が rank 落ちしている場合は、ピボット選択を行っても途中で 0 になる (cf. 特異値分解を用いた最小二乗解)

ガウスの消去法の行列表示

- $a_{kk}^{(k)}$ を用いた $a_{ik}^{(k)}$ $(i > k)$ の消去は、方程式の両辺に左から

$$M_k = \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ 0 & 0 & \ddots & & & & & \\ \vdots & \vdots & & 1 & & & & \\ \vdots & \vdots & & -m_{k+1,k} & 1 & & & \\ \vdots & \vdots & & -m_{k+2,k} & 0 & \ddots & & \\ \vdots & \vdots & & \vdots & \vdots & & 1 & \\ 0 & 0 & \dots & -m_{nk} & 0 & \dots & 0 & 1 \end{bmatrix}$$

を掛けるのと等価: $M_k A^{(k)} = A^{(k+1)}$、$M_k \mathbf{b}^{(k)} = \mathbf{b}^{(k+1)}$

LU 分解

- M_k の逆行列

$$L_k = M_k^{-1} = \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ 0 & 0 & \ddots & & & & & \\ \vdots & \vdots & & 1 & & & & \\ \vdots & \vdots & & m_{k+1,k} & 1 & & & \\ \vdots & \vdots & & m_{k+2,k} & 0 & \ddots & & \\ \vdots & \vdots & & \vdots & \vdots & & 1 & \\ 0 & 0 & \dots & m_{nk} & 0 & \dots & 0 & 1 \end{bmatrix}$$

から $L = L_1 L_2 \cdots L_{n-1}$ を定義すると、L は下三角行列、また $U = A^{(n)}$ (上三角行列) とすると、$A = LU$

LU 分解

- LU 分解による連立一次方程式の解法
  - 方程式は $A\mathbf{x} = LU\mathbf{x} = \mathbf{b}$ と書ける
  - まず、$L\mathbf{y} = \mathbf{b}$ を解いて、$\mathbf{y}$ を求める (前進代入)
  - 次に、$U\mathbf{x} = \mathbf{y}$ を解いて、$\mathbf{x}$ を求める (後退代入)
- 計算量はガウスの消去法と変わらない
- 一度 LU 分解をしておけば、異なる $\mathbf{b}$ に対する解も簡単に求められる

- 行列式は U の対角成分の積で与えられる (ピボット選択する場合は、行の入れ替えにより符号が変わることに注意)

LAPACK による連立一次方程式の求解

- LU 分解: dgetrf.h

```
void dgetrf_(int *M, int *N, double *A, int *LDA, int*IPIV,
  int *INFO);
```

  - 行列 A は、下三角部が L、上三角部が U で上書きされる。(対角にはは U の対角成分が入る。L の対角成分は全て 1 なので保存されない)
  - 配列 IPIV の第 i 要素には、i 行目を使ってガウス消去する際に何行目と入れ替えられたかが保存される。(i は 1 から数えることに注意)
- 前進代入と交代代入: dgetrs.h

```
void dgetrs_(char *TRANS, int *N, int *NRHS, double *A,
  int *LDA, int *IPIV, double *B, int *LDB, int *INFO);
```

  - 行列 A はと IPIV は dgetrf の結果をそのまま渡す。配列 B は右辺のベクトル、NRHS はベクトルの数 (1 でよい)
- LU 分解の例: lu_decomp.c
- dgesv は中で dgetrf と dgetrs を順に呼び出している

ピボット選択と置換行列

- 配列 IPIV の第 i 要素には、i 行目を使ってガウス消去する際に何行目と入れ替えられたかが保存される。(i は 1 から数えることに注意)
- i 行目と j 行目の入れ替えは、行列に左から置換行列 P_{ij} を掛ければよい
  - P_{ij} は (i,i), (j,j) 成分が 0、(i,j), (j,i) 成分が 1、それ以外は対角成分が 1 の行列。($i \neq j$ なら $\det P_{ij} = -1$、$i = j$ なら $\det P_{ij} = 1$)
  - $[P_{ij}]^2 = E$、ゆえに $[P_{ij}]^{-1} = P_{ij}$ (直交かつ対称)
- 例えば 3×3 行列 A を LU 分解して、IPIV が $(3,3,3)$ の場合
  - A に左から $P = P_{33}P_{23}P_{13}$ を掛けた行列が (ピボット選択なしの) LU 分解されていることになる

$$P = E \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

- $A = P^{-1}LU = P^tLU$ が成り立つ

直接法と反復法

- 直接法: 連立方程式を有限回数 ($\sim n^3$) の手間で直接解く
- 反復法: $A\mathbf{x} = \mathbf{b}$ を、等価な $\mathbf{x} = \phi(\mathbf{x}) = M\mathbf{x} + \mathbf{c}$ の形に変形し、適当な初期値 $\mathbf{x}_0$ から出発して、$\mathbf{x}^{(k+1)} = \phi(\mathbf{x}^{(k)})$ を繰り返して解を求める
  - 欠点: 有限回数では終わらない (あらかじめ定めた収束条件が満たされるまで反復)
  - 利点: 行列ベクトル積 $M\mathbf{x}$ が計算できさえすればよい。特に M が疎行列の場合には、$M\mathbf{x}$ は非常に高速に計算できる可能性がある。メモリの点でも有利
  - 利点: 直接法に比べて、プログラムも比較的単純になる場合が多い

反復法

- 行列 A を対角行列 D、左下三角行列 E、右上三角行列 F の和に分解

$$A\mathbf{x} = (D + E + F)\mathbf{x} = \mathbf{b}$$

- ヤコビ法: 対角成分以外を右辺に移す

$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - (E + F)\mathbf{x}^{(k)}) = -D^{-1}(E + F)\mathbf{x}^{(k)} + D^{-1}\mathbf{b}$$

- ガウスザイデル法: ヤコビ法で右辺の $\mathbf{x}$ の値として、各段階ですでに得られている最新のものを使う

$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - E\mathbf{x}^{(k+1)} - F\mathbf{x}^{(k)})$$

$$\mathbf{x}^{(k+1)} = -(D + E)^{-1}F\mathbf{x}^{(k)} + (D + E)^{-1}\mathbf{b}$$

反復法

- SOR (Successive Over-Relaxation) 法: ガウスザイデル法における修正量に 1 より大きな値 (ω) を掛け、補正を加速

$$\xi^{(k+1)} = D^{-1}(\mathbf{b} - E\mathbf{x}^{(k+1)} - F\mathbf{x}^{(k)})$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega(\xi^{(k+1)} - \mathbf{x}^{(k)})$$

  $\xi^{(k+1)}$ を消去すると

$$\begin{aligned}\mathbf{x}^{(k+1)} =& (I + \omega D^{-1}E)^{-1}\{(1-\omega)I - \omega D^{-1}F\}\mathbf{x}^{(k)} \\ &+ \omega(D + \omega E)^{-1}\mathbf{b}\end{aligned}$$

- 反復法は常に収束するとは限らない
- 行列 A が対角優位、あるいは正定値対称行列の場合には収束が保証される