Gauss の消去法と LU 分解

桂田 祐史

2005 年 4 月 26 日

目 次

1 Gauss の消去法を例で **1**

2 LU 分解とは何か **2**
2.1 定義 2
2.2 LU 分解の意義 3

3 Gauss の消去法と LU 分解 **4**
3.1 前進消去 4
3.2 後退代入 6
3.3 Gauss の消去法のプログラム 7
3.4 Gauss の消去法は LU 分解をしていること 7
3.5 具体的な計算 8
3.6 LU 分解のプログラム 9

4 これから書き足したいこと **10**
(基本的には『連立 1 次方程式 I, — 計算量と直接法 —』[1] からの抜き出し)

1 Gauss の消去法を例で

連立 1 次方程式の解法として、線形代数の教科書には **Cramer** (クラーメル) の公式や掃き出し法 (Jordan (ヨルダン) の消去法ともいう) が説明されていることが多いが、連立 1 次方程式の解法としての実用性は、Gauss の消去法の方が高い。

(Gauss の消去法なんか知らないと豪語 (?) する数学者がいたりするのだが、見ようによっては、Gauss の消去法は中学校で学ぶ加減法そのものであるとも考えられる。初めて学んだ方法が優れた方法であるのに、それを忘れてしまって、後から学んだあまり優れていない方法が大きい顔をしているのも変な話である。)

[1] http://www.math.meiji.ac.jp/~mk/labo/text/linear-eq-1.pdf

例として次の方程式を取りあげて説明しよう。

$$\begin{cases} 2x_1 + 3x_2 - x_3 = 5 \\ 4x_1 + 4x_2 - 3x_3 = 3 \\ -2x_1 + 3x_2 - x_3 = 1. \end{cases}$$

掃き出し法では係数行列と右辺のベクトルを並べた行列を作り、それに

1. ある行に 0 でない定数をかける。
2. 二つの行を入れ換える。
3. ある行に別の行を加える。

のような操作 — 行に関する基本変形と呼ぶ — をほどこして、連立方程式の係数行列に相当する部分を単位行列にするのであった。

$$\begin{pmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{5}{2} \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{5}{2} \\ 0 & -2 & -1 & -7 \\ 0 & 6 & -2 & 6 \end{pmatrix} \to$$

$$\to \begin{pmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{5}{2} \\ 0 & 1 & \frac{1}{2} & \frac{7}{2} \\ 0 & 3 & -1 & 3 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & -\frac{5}{4} & -\frac{11}{4} \\ 0 & 1 & \frac{1}{2} & \frac{7}{2} \\ 0 & 0 & -\frac{5}{2} & -\frac{15}{2} \end{pmatrix} \to \begin{pmatrix} 1 & 0 & -\frac{5}{4} & -\frac{11}{4} \\ 0 & 1 & \frac{1}{2} & \frac{7}{2} \\ 0 & 0 & 1 & 3 \end{pmatrix} \to$$

$$\to \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{pmatrix}, \quad \text{ゆえに} \begin{pmatrix} x_2 \\ x_3 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

Gauss の消去法も、前半の段階はこの方法に似ていて、同様の変形を用いて掃き出しを行なうのだが、以下のように対角線の下側だけを 0 にする。

$$\begin{pmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{pmatrix} \to \begin{pmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 6 & -2 & 6 \end{pmatrix} \to \begin{pmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & -5 & -15 \end{pmatrix}.$$

最後の行列は

$$2x_1 + 3x_2 - x_3 = 5, \quad -2x_2 - x_3 = -7, \quad -5x_3 = -15$$

ということを表しているので、後の方から順に

$$x_3 = \frac{-15}{-5} = 3, \quad x_2 = \frac{-7 + x_3}{-2} = 2, \quad x_1 = \frac{5 - 3x_2 + x_3}{2} = \frac{5 - 3 \times 2 + 3}{2} = 1$$

と解くことが出来る。前半の対角線の下側を 0 にする掃き出しの操作を前進消去 (**forward elimination**)、後半の代入により解の値を求める操作を後退代入 (**backward substitution**) と呼ぶ。

2　LU 分解とは何か

2.1　定義

正方行列 L の対角線よりも上にある成分がすべて 0 であるとき、L を下三角行列という。

$$L = \begin{pmatrix} u_{11} & & & 0 \\ u_{21} & u_{22} & & \\ \vdots & & \ddots & \\ u_{n1} & u_{n2} & \cdots & u_{nn} \end{pmatrix}.$$

正方行列 U の対角線よりも下にある成分がすべて 0 であるとき、L を上三角行列という。

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ 0 & & & u_{nn} \end{pmatrix}.$$

与えられた行列 A に対して、

$$A = LU$$

を満たす下三角行列 L と上三角行列 U を見い出すことを、A を LU 分解するという。

2.2　LU 分解の意義

A が $A = LU$ と LU 分解されているとき、連立 1 次方程式

$$Ax = b$$

は少ない計算量で解くことが出来る。以下、このことを説明する (早い話、$A^{-1} = (LU)^{-1} = U^{-1}L^{-1}$ であり、三角行列である L, U の逆行列が効率良く計算できるということである)。

$$LUx = b$$

は

$$Ly = b, \quad Ux = y$$

という二つの問題に分解される。

$Ly = b$ は

$$\begin{array}{lllll} \ell_{11}y_1 & & & & = b_1 \\ \ell_{21}y_1 & +\ell_{22}y_2 & & & = b_2 \\ \ell_{31}y_1 & +\ell_{32}y_2 & +\ell_{33}y_3 & & = b_3 \\ & & & & \\ \ell_{n1}y_1 & +\ell_{n2}y_2 & +\ell_{n3}y_3 & \cdots \quad +\ell_{nn}y_n & = b_n \end{array}$$

これは上から順に

$$\begin{aligned}
y_1 &= b_1/\ell_{11},\\
y_2 &= \left(b_2-\ell_{21}y_1\right)/\ell_{22},\\
y_3 &= \left(b_3-\ell_{31}y_1-\ell_{32}y_2\right)/\ell_{33},\\
&\cdots\\
y_i &= \left(b_i-\sum_{j=1}^{i-1}\ell_{i,j}y_j\right)/\ell_{ii},\\
&\cdots\\
y_n &= \left(b_n-\sum_{j=1}^{n-1}\ell_{i,j}y_j\right)/\ell_{nn}
\end{aligned}$$

と解くことが出来る。

これを計算するには、乗除算が $1+2+\cdots+n=n(n+1)/2$ 回必要である。

同様に $Ux=y$ は

$$\begin{aligned}
u_{11}y_1 \quad \cdots \quad +u_{1,n-2}y_{n-2} \quad +u_{1,n-1}y_{n-1} \quad +u_{1n}y_n &= b_1,\\
\cdots&\\
u_{n-2,n-2}y_{n-2} \quad +u_{n-2,n-1}y_{n-1} \quad +u_{n-2,n}y_n &= b_{n-1},\\
u_{n-1,n-1}y_{n-1} \quad +u_{n-1,n}y_n &= b_{n-1},\\
u_{n,n}y_n &= b_n
\end{aligned}$$

なので、下から順に

$$\begin{aligned}
x_n &= y_n/u_{nn}\\
x_{n-1} &= \left(b_{n-1}-u_{n-1,n}y_n\right)/u_{n-1,n-1}\\
x_{n-2} &= \left(b_{n-2}-u_{n-2,n-1}y_{n-1}-u_{n-2,n}y_n\right)/u_{n-2,n-2}\\
&\cdots\\
x_i &= \left(b_i-\sum_{j=i+1}^{n}u_{i,j}y_j\right)/u_{ii}\\
&\cdots\\
x_1 &= \left(b_1-\sum_{j=2}^{n}u_{1,j}y_j\right)/u_{11}
\end{aligned}$$

これも計算するには、乗除算が $1+2+\cdots+n=n(n+1)/2$ 回必要である。

まとめると、$n(n+1)/2+n(n+1)/2=n(n+1)$ 回の乗除算で連立 1 次方程式が解けることになる。

Gauss の消去法で連立 1 次方程式を解くには、$n^3/3$ 回程度の乗除算が必要であったことを思い出すと、LU 分解がいかに効率的かが分かる。

逆行列 A^{-1} が得られている場合に、$A^{-1}b$ を計算するのに n^2 回の乗算が必要であるから、LU 分解が得られていると、ほぼ逆行列が求まっているのと同等の効率で解が得られることが分かる。

注意　実際の応用では、行列 A は疎であることが多く、その場合、逆行列は疎であることが期待できないが、L, U は疎であるので、LU 分解を利用する方が断然効率的になる。

細かな注意　(少しでも逆行列に負けるのはしゃくなので) LU 分解では、L または U の対角成分をすべて 1 であるようにできるので、乗除算の回数は $n(n+1)/2 + n(n-1)/2 = n^2$ と、逆行列を用いる場合と、まったく同等に出来る。

3　Gauss の消去法と LU 分解

Gauss の消去法を復習しよう。 $Ax = b$ $(b \in \mathbf{R}^n,\ A \in M(n;\mathbf{R}))$ を解くために、最初に A, b を並べた行列 $(A\ b)$ を作り、

1. 前進消去
2. 後退代入

を施す。

3.1　前進消去

$k = 1, 2, \cdots, n-1$ の順に、第 k 行で第 i 行 $(i = k+1, \cdots, n)$ を掃き出すという操作を続けて、上三角行列に変形する。

$$(A\ b) = (A^{(1)}\ b^{(1)}),$$

$$(A^{(1)}\ b^{(1)}) \to (A^{(2)}\ b^{(2)}) \cdots \to (A^{(k)}\ b^{(k)}) \to (A^{(k+1)}\ b^{(k+1)}) \cdots \to (A^{(n)}\ b^{(n)})$$

$$A^{(n)} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \\ \mathbf{0} & & & u_{nn} \end{pmatrix} \stackrel{\text{def.}}{=} U, \quad b^{(n)} = y.$$

$A^{(k)} = (a_{ij}^{(k)})$, $b^{(k)} = (b_i^{(k)})$ とすると、

前進消去過程の具体的な計算手順

1.

$$a_{ij}^{(1)} \stackrel{\text{def.}}{=} a_{ij}, \quad b_i^{(1)} \stackrel{\text{def.}}{=} b_i \quad (i,j = 1,2,\cdots,n)$$

2. 以下 $A^{(k)} = (a_{ij}^{(k)})$, $b^{(k)} = (b_i^{(k)})$ を漸化式で定める。すなわち $k = 1,2,\cdots,n-1$ の順に

(1)
$$a_{ij}^{(k+1)} \stackrel{\text{def.}}{=} \begin{cases} a_{ij}^{(k)} & (1 \le i \le k;\ 1 \le j \le n) \\ a_{ij}^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)} & (k+1 \le i \le n;\ 1 \le j \le n), \end{cases}$$

(2)
$$b_i^{(k+1)} \stackrel{\text{def.}}{=} \begin{cases} b_i^{(k)} & (1 \le i \le k) \\ b_i^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)} & (k+1 \le i \le n), \end{cases}$$

3. $U = (u_{ij})$, $y = (y_i)$ とおくと、

$$u_{ij} = a_{ij}^{(n)}, \quad y_i^* = b_i^{(n)}.$$

プログラミングでは、一つの配列変数を用意し、$A^{(k)} = (a_{ij}^{(k)})$ $(k = 1,2,\cdots,n)$ を順番に上書きして計算していけばよい。C 言語風に表すと

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

3.2 後退代入

(3)
$$x_n = \frac{b_n^*}{a_{nn}^*},$$

(4)
$$x_k = \frac{b_k^* - \displaystyle\sum_{j=k+1}^{n} a_{kj}^* x_j}{a_{kk}^*}$$

```
x[n] = b[n] / a[n][n];
for (k = n - 1; k >= 1; k--) {
    s = b[k];
```

```
        for (j = k + 1; j <= n; j++)
            s = s - a[k][j] * x[j];
        x[k] = s / a[k][k];
    }
```

3.3 Gauss の消去法のプログラム

```
/*
 * gauss-ver1.c
 */

#include <matrix.h>

void gauss(int n, matrix a, vector x)
{
    int i, j, k;
    double q;
    /* 前進消去 */
    for (k = 1; k < n; k++)
        for (i = k + 1; i <= n; i++) {
            q = a[i][k] / a[k][k];
#ifdef  ORIGINAL
            for (j = 1; j <= n; j++)
#else
            for (j = k + 1; j <= n; j++)
#endif
                a[i][j] -= q * a[k][j];
            x[i] -= q * x[k];
        }
    /* 後退代入 */
    x[n] /= a[n][n];
    for (k = n - 1; k >= 1; k--) {
        for (j = k + 1; j <= n; j++)
            x[k] -= a[k][j] * x[j];
        x[k] /= a[k][k];
    }
}
```

掃き出しで 0 になると分かり切っているところの計算は省略できることに注意すると、前進消去の j についてのループを、1 でなく k+1 から始めても構わないことが分かる。

3.4 Gauss の消去法は LU 分解をしていること

$$(A^{(k)}\ b^{(k)}) \to (A^{(k+1)}\ b^{(k+1)})$$

における基本変形を考えると、第 k 行で、第 i 行 $(k+1 \le i \le n)$ を掃き出している。これは、左から基本行列をかけることになる。

$$A^{(k+1)} = L_n^{(k)} L_{n-1}^{(k)} \cdots L_{k+1}^{(k)} A^{(k)}, \quad b^{(k+1)} = L_n^{(k)} L_{n-1}^{(k)} \cdots L_{k+1}^{(k)} b^{(k)},$$

ここで $L_j^{(k)}$ は、第 k 行で第 j 行を掃き出す基本変形を表す行列である。

$$L_j^{(k)} = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ \cdots & & & & & \\ & q & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & \end{pmatrix}.$$

ここで

$$L^{(k)} \stackrel{\text{def.}}{=} L_n^{(k)} L_{n-1}^{(k)} \cdots L_{k+1}^{(k)}$$

とおくと、$L^{(k)}$ は第 k 行での掃き出し操作を表す下三角行列で、

$$A^{(k+1)} = L^{(k)} A^{(k)}, \quad b^{(k+1)} = L^{(k)} b^{(k)}.$$

さらに

$$\mathcal{L} \stackrel{\text{def.}}{=} L^{(n-1)} L^{(n-2)} \cdots L^{(2)} L^{(1)}$$

とおくと、

$$A^{(n)} = \mathcal{L} A^{(1)}, \quad b^{(n)} = \mathcal{L} b^{(1)}.$$

言い替えると

$$U = \mathcal{L} A, \quad y = \mathcal{L} b.$$

これから

$$L \stackrel{\text{def.}}{=} \mathcal{L}^{-1}$$

とおけば L は下三角行列で

$$A = LU.$$

つまり A の LU 分解が得られた。

3.5 具体的な計算

上の計算を振り替えると、U と

$$q_i^{(k)} \stackrel{\text{def.}}{=} \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)} \quad (k = 1, 2, \cdots, n-1;\ i = k+1, \cdots, n)$$

さえ記憶しておけば、新しい b が与えられたときに、$Ax = b$ を解くことが出来る。

実は

$$L = \begin{pmatrix} 1 & & & & \\ q_2^{(1)} & 1 & & & \\ q_3^{(1)} & q_3^{(2)} & & & \\ \cdots & & & & \\ & & & 1 & \\ q_n^{(1)} & q_n^{(2)} & \cdots & q_n^{(n-1)} & 1 \end{pmatrix}$$

であることが分かる。

3.6 LU 分解のプログラム

まずは素朴なプログラム。

```
/* lu-ver1.c */

#include <matrix.h>
#include "lu-ver1.h"

void lu(int n, matrix a, matrix L, matrix u)
{
  int i, j, k;
  double q;

  for (i = 1; i <= n; i++) {
    for (j = 1; j <= n; j++) {
      u[i][j] = a[i][j];
      L[i][j] = 0.0;
    }
    L[i][i] = 1.0;
  }
  for (k = 1; k < n; k++)
    for (i = k + 1; i <= n; i++) {
      q = u[i][k] / u[k][k];
      /* 定義式では j は 1 からだが、実は j を k からでも OK (k+1 は不可！) */
#ifdef ORIGINAL
      for (j = 1; j <= n; j++)
#else
      for (j = k; j <= n; j++)
#endif
        u[i][j] = u[i][j] - q * u[k][j];
      L[i][k] = q;
    }
}

/* 行列 A の LU 分解を用いて A x = b を解く */
void solve(int n, matrix L, matrix U, vector b, vector x)
{
  int i, j;
  double sum;
  vector y = new_vector(n + 1);
  /* 前進消去 */
  for (i = 1; i <= n; i++) {
    sum = b[i];
    for (j = 1; j < i; j++) sum -= L[i][j] * y[j];
    y[i] = sum / L[i][i];
  }
  /* 後退代入 */
  for (i = n; i >= 1; i--) {
    sum = y[i];
    for (j = i + 1; j <= n; j++) sum -= U[i][j] * x[j];
    x[i] = sum / U[i][i];
  }
  /* 不要になったメモリーを再利用可能なように解放する */
  free_vector(y);
}
```

L の対角成分は 1 なので、本当に記憶していなければならないのは、L の対角線の下側と、U の対角成分と対角線の上側の成分である。matrix a の内容を保存しなくて良いならば、次のようにして、L, U のうちの記憶すべき内容を a に上書きするコードが作れる。(j についてのループを k+1 から始めているのがミソである。)

```
/* lu-ver2.c */

#include <matrix.h>
#include "lu-ver2.h"

void lu(int n, matrix a)
{
  int i, j, k;
  double q;

  for (k = 1; k < n; k++)
    for (i = k + 1; i <= n; i++) {
      q = a[i][k] / a[k][k];
      for (j = k + 1; j <= n; j++)
        a[i][j] -= q * a[k][j];
      a[i][k] = q;
    }
}

/* 行列 A の LU 分解を用いて A x = b を解く */
void solve(int n, matrix a, vector b)
{
  /* 以下の計算手順はプリント 4.3.4 を見よ */
  int i, j;

  /* 前進消去 */
  for (i = 1; i <= n; i++)
    for (j = 1; j < i; j++)
      b[i] -= a[i][j] * b[j];
  /* 注意: L_{i,i}=1 なので割り算は必要ない。 */
  /* 後退代入 */
  for (i = n; i >= 1; i--) {
    for (j = i + 1; j <= n; j++)
      b[i] -= a[i][j] * b[j];
    b[i] /= a[i][i];
  }
}
```

4 これから書き足したいこと

- 計算量の解析 (帯行列の場合を含む)
- Cholesky 分解、QR 分解
- ピボットの選択
- LINPACK, LAPACK の奨め