数値計算 Class-4 演習

21T2166D 渡辺大樹

2025 年 5 月 11 日

1　演習内容

Class-4 の演習では連立一次方程式の係数行列を下三角行列と上三角行列の積に分解し、連立一次方程式を解く、LU 分解というアルゴリズムを C で実装し、実際に行列を LU 分解していく。

LU 分解は前回 Class-3 で用いた my_library.h を改変したソースコード 1 と 2 で実装されている。

ソースコード 1　my_library_v2.h

```
1  #ifndef MY_LIBRARY_H
2  #define MY_LIBRARY_H
3  /* my_library version 2.0 */
4  void irekae(double a[][N + 1], int i, int n){
5      int m, j, k;
6      double key;
7      m = i;
8
9      /*  絶対値の最大のものを探す */
10     for (k = i + 1; k <= n; k++){
11         if (fabs(a[m][i]) < fabs(a[k][i])){
12             m = k;
13         }
14     }
15
16     /* 第 m  行と第  i 行 を入れ替える */
17     for (j = 1; j <= n + 1; j++){
18         key = a[m][j];
19         a[m][j] = a[i][j];
20         a[i][j] = key;
21     }
22 }
23 // makes matrix "a" upper triangular by Gaussian elimination // n is the
```

```
    number of rows
int gaussian_elimination(double a[][N + 1], int n){
    double p, q;
    int i, j, k;
    /*対角成分より下を掃き出して上三角行列の形に変形*/
    for (i = 1; i <= n; i++){
        irekae(a, i, n);
        p = a[i][i];
        if (fabs(p) < 1.0e-6){
            printf("一意解をもちません\n");
            return -1;
        }
        /* 第 i 行を (i,i)成分で割る */
        for (j = i; j <= n + 1; j++){
            a[i][j] = a[i][j] / p;
        }
        for (k = i + 1; k <= n; k++){
            q = a[k][i];
            for (j = 1; j <= n + 1; j++){
                a[k][j] = a[k][j] - a[i][j] * q;
            }
        }
    }
    return 0;
}
// requires "a" to be an upper triangular matrix
// x is the output vector
// n is the number of rows
void inverse_substitution(double a[][N + 1], double x[N], int n){
    int i, j;
    double s;

    /*逆進代入による計算*/
    for (i = n; i >= 1; i--){
        s = 0.0;
        for (j = i + 1; j <= n; j++){
            s += a[i][j] * x[j];
        }
        x[i] = a[i][n + 1] - s;
    }
}

```

```
65 /*行列の入力*/
66 int input_matrix(double a[N][N]){
67     int n, i, j;
68     char z, zz;
69 
70     while (1){
71         printf("行列の次数の入力 (1 < n < %d) n = ", N - 1);
72         scanf("%d%c", &n, &zz);
73 
74         if ((n <= 1) || (N - 1 <= n))
75             continue;
76 
77         printf("\n 行列 A の成分を入力します\n\n");
78 
79         for (i = 1; i <= n; i++){
80             for (j = 1; j <= n; j++){
81                 printf("a(%d, %d)=", i, j);
82                 scanf("%lf%c", &a[i][j], &zz);
83             }
84             printf("\n");
85         }
86         printf("正しく入力しましたか？ (y/n)");
87         scanf("%c%c", &z, &zz);
88         if (z == 'y')
89             break;
90     }
91     return n;
92 }
93 /*行列の出力*/
94 void print_matrix(double a[N][N], int n){
95     int i, j;
96     for (i = 1; i <= n; i++){
97         for (j = 1; j <= n; j++){
98             printf(" %10.6lf", a[i][j]);
99         }
100        printf("\n");
101    }
102 }
103 #endif
```

ソースコード 2　ludecomp.c

```
#include <stdio.h>
#include <math.h>
#define N 8
#include "my_library_v2.h"
int main(void){
    int i, j, n, k;
    static double p, q, a[N][N], l[N][N], u[N][N];

    /*** 行列の入力 ***/
    n = input_matrix(a);
    printf("入力: \n");
    print_matrix(a, n);
    printf("\n");
    /*LU 分解とガウス消去*/
    for (i = 1; i <= n; i++){
        p = a[i][i];

        if (fabs(p) < 1.0e-6){
            printf("この行列はLU 分解出来ません. \n");
            return -1;
        }
        for (j = i; j <= n + 1; j++){
            l[j][i] = a[j][i];
            a[i][j] = a[i][j] / p;
        }

        for (k = i + 1; k <= n; k++){
            q = a[k][i];

            for (j = 1; j <= n + 1; j++){
                a[k][j] = a[k][j] - a[i][j] * q;
            }
        }
        for (j = i; j <= n; j++){
            u[i][j] = a[i][j];
        }
    }
    /*** 下三角行列L の表示 ***/
    printf("L:\n");
    print_matrix(l, n);
    printf("\n");
```

```
42      /*** 上三角行列L の表示 ***/
43      printf("U:\n");
44      print_matrix(u, n);
45      return 0;
46  }
```

以上のコードで行われていること、特にソースコード 2 に実装されている LU 分解について解説していく。

LU 分解とは前述したとおり行列を下三角行列と上三角行列の積で表すことで、Class-3 で行ったガウス法と同様この手法でも連立一次方程式を解くことが可能である。

解説をわかりやすくするため、例として前回同様以下の連立方程式を用いる。

$$\begin{cases} 2x + 3y - z = 1 \\ x + y + 2z = 0 \\ 3x - y + z = 2 \end{cases} \tag{1}$$

(1) の連立方程式は行列とベクトルの積を用いて以下のように表すことができる。

$$\begin{bmatrix} 2 & 3 & -1 \\ 1 & 1 & 2 \\ 3 & -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} \tag{2}$$

この (2) の式を用いて、ソースコード 1 のアルゴリズムを説明する。

まずこの (2) 式から、係数行列を A、左辺の列ベクトルを $\mathbf{b}$ として

$$A_0 = \begin{bmatrix} A & \mathbf{b} \end{bmatrix} = \begin{bmatrix} 2 & 3 & -1 & 1 \\ 1 & 1 & 2 & 0 \\ 3 & -1 & 1 & 2 \end{bmatrix} \tag{3}$$

とおく。

A_0 について A_0 の $(1,1)$ 成分を軸としてその下を掃き出した行列を A_1 とすると

$$A_1 = \begin{bmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{5}{2} & -\frac{1}{2} \\ 0 & -\frac{11}{2} & \frac{5}{2} & \frac{1}{2} \end{bmatrix} \tag{4}$$

となる。

このとき三次の単位行列の第一列を A_0 の第一列と入れ替えた行列 B_1 を用いると A_0 と A_1 には

$$A_0 = B_1 A_1 = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{5}{2} & -\frac{1}{2} \\ 0 & -\frac{11}{2} & \frac{5}{2} & \frac{1}{2} \end{bmatrix} \tag{5}$$

が成り立つ。

次に A_1 の $(2,2)$ 成分を軸としてその下を掃き出した行列を A_2 とし、三次の単位行列の第二列を A_1 の第二列の対角成分以下と入れ替えた行列 B_2 を用いると A_1 と A_2 には

$$A_1 = B_2 A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 \\ 0 & -\frac{11}{2} & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 1 & -5 & 1 \\ 0 & 0 & -25 & 6 \end{bmatrix} \tag{6}$$

が成り立つ。

同様のこと (上記 (6) で行っていることを第三列として読み替えて) を第三列でも行えば A_2 は B_3, A_3 を用いて

$$A_2 = B_3 A_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -25 \end{bmatrix} \begin{bmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 1 & -5 & 1 \\ 0 & 0 & 1 & -\frac{6}{25} \end{bmatrix} \tag{7}$$

と表せる。

ここで行列 A_3 の第三列目までの上三角行列を U, 第四列を $\mathbf{b}'$ とおくと $A_3 = \begin{bmatrix} U & \mathbf{b}' \end{bmatrix}$ と表せる。式 (5)(6)(7) より

$$A_0 = B_1 B_2 B_3 \begin{bmatrix} U & \mathbf{b}' \end{bmatrix} \tag{8}$$

となる。ここで $L = B_1 B_2 B_3$ とおくと

$$L = B_1 B_2 B_3 = \begin{bmatrix} 2 & 0 & 0 \\ 1 & -\frac{1}{2} & 0 \\ 3 & -\frac{11}{2} & -25 \end{bmatrix} \tag{9}$$

となる。このことから

$$A_0 = L \begin{bmatrix} U & \mathbf{b}' \end{bmatrix} = \begin{bmatrix} LU & L\mathbf{b}' \end{bmatrix} \tag{10}$$

と式変形でき、$A_0 = [\,A \;\; \mathbf{b}\,]$ であったため両辺の行列の対応を考えることで

$$A = LU, \qquad \mathbf{b} = L\mathbf{b}' \tag{11}$$

が得られる。これが LU 分解の主な動きである。今は三次の連立方程式について説明したがこれは任意の n 次の行列、ないしは連立方程式に用いることができる。

この LU 分解は、これを行うことで $Ax = \mathbf{b}$ となっていた連立方程式を $Ux = \mathbf{b}'$ と解くことができる。

実際に上記の連立方程式を $Ux = \mathbf{b}'$ で解いてみる。この連立方程式を $Ux = \mathbf{b}'$ で表すと

$$\begin{bmatrix} 1 & \frac{3}{2} & -\frac{1}{2} \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 1 \\ -\frac{6}{25} \end{bmatrix} \tag{12}$$

となる。この式は逆進代入を使うことで簡単に解くことができる。第三行目から z は $z = -\frac{6}{25}$ と解けて、この解を第二行目に代入して $y = -\frac{1}{5}$、以上 y, z を第一行目に代入して $x = \frac{17}{25}$ と解ける。

このように $Ax = \mathbf{b}$ となっていた連立方程式を LU 分解を用いて $Ux = \mathbf{b}'$ と解くことで簡単に解を求めることができる。

2　演習とその結果

演習として教科書に載っている 2.10 の (1)(2) を解く。

2.1　問 2.10

問 2.10 の (1)(2) は以下の行列を LU 分解し、行列式の値を求める問題である。

(1)

$$\begin{bmatrix} 2 & 6 & 8 \\ -1 & 1 & -12 \\ 3 & 10 & 5 \end{bmatrix} \tag{13}$$

(2)

$$\begin{bmatrix} 2 & 4 & 0 \\ -3 & -2 & 12 \\ 1 & -3 & -14 \end{bmatrix} \tag{14}$$

行列式の値を求めるため、ソースコード 2 を以下ソースコード 3 のように書き加えた。加筆点には＋をつけている。

ソースコード 3　ludecomp_A.c

```
1     #include <stdio.h>
2  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
3         int i, j, n, k;
4         + double A = 1.0;
5         static double p, q, a[N][N], l[N][N], u[N][N];
6  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
7             for (j = i; j <= n; j++){
8                 u[i][j] = a[i][j];
9             }
10            + A *= l[i][i];
11 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
12        print_matrix(u, n);
13        + printf("A:%10.6lf" , A);
14        return 0;
15 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

このコードでは LU 分解でできた L の行列の対角成分の積が行列式の値となることを用いて計算している。このソースコードを動かすことで以下の結果が得られた。

(1)

L:
2.000000 0.000000 0.000000
-1.000000 4.000000 0.000000
3.000000 1.000000 -5.000000

U:
1.000000 3.000000 4.000000
0.000000 1.000000 -2.000000
0.000000 0.000000 1.000000
A:-40.000000

(2)

L:
2.000000 0.000000 0.000000
-3.000000 4.000000 0.000000
1.000000 -5.000000 1.000000

U:
1.000000 2.000000 0.000000

0.000000 1.000000 3.000000
0.000000 0.000000 1.000000
A:8.000000

この結果は教科書通りの答えとなっている。

3 考察

以上の演習からガウスの掃き出し法とは異なる、連立一次方程式の解き方を理解できた。LU 分解法は実際に連立方程式を線形計画法などで利用することを考えると、任意の係数行列の LU 分解を求めた後に左辺の列ベクトルを自由に決められるためガウスの掃き出し法よりも高速に問題を解くことができるため、その点ではガウスの掃き出し法よりも優れていると考える。

また実際にこれらのソースコードにいくつかの関数を加えて連立一次方程式を解こうとしたがあまりうまくいかなかった。C の勉強をして原因を探ってみたい。