数値計算 Class-3 演習

21T2166D 渡辺大樹

2024 年 5 月 2 日

1　演習内容

Class-3 の演習ではガウスの消去法、または掃き出し法とも呼ばれるアルゴリズムを C で実装し、任意の n 元一次連立方程式を解いた。

ガウスの消去法は以下ソースコード 1 と 2 で実装されている。

ソースコード 1　gauss.c

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define N 10
5
6  #include "my_library.h"
7
8  int main(void) {
9          int n,i,j;
10         static double a[N][N+1], x[N];
11         //static double a[N][N+1] =
               {{0.0,0.0,0.0,0.0,0.0},{0.0,2.0,3.0,-1.0,1.0},{0.0,1.0,1.0,2.0,0.0},{0.0,3.
12         char z, zz;
13
14         //n = 3;
15
16         while(1) {
17                 printf("ガウスの消去による連立方程式の解法\n");
18                 printf("何元連立法手式ですか (1 < n < 9) n = ");
19                 scanf("%d%c",&n, &zz);
20                 if ((n <= 1) || (N-1 <= n)) continue;
21                 printf("\n 係数を入力してください\n\n");
22                 for(i=1;i<=n;i++) {
```

```
23                      for(j=1; j<=n+1; j++) {
24                              printf("a(%d, %d)=", i, j);
25                              scanf("%lf%c",&a[i][j], &zz);
26                      }
27                      printf("\n");
28              }
29              printf("正しく入力しましたか？(y/n)");
30              scanf("%c%c", &z, &zz);
31              if (z == 'y') break;
32      }
33
34      int ret = gaussian_elimination(a, n);
35      if (ret != 0) {
36              return ret;
37      }
38      inverse_substitution(a, x, n);
39
40      // 解の表示
41      printf("\n 連立方程式の解\n\n");
42      for(i=1; i<=n ; i++) {
43              printf("x( %d ) = %10.61f\n", i, x[i]);
44      }
45      return 0;
46 }
```

ソースコード 2　my_library.h

```
#ifndef MY_LIBRARY_H
#define MY_LIBRARY_H

void irekae(double a[][N+1], int i, int n) {
        int m, j, k;
        double key;
        m = i;

        /*  絶対値の最大 (最小)のものを探す */
        for(k=i+1; k<=n;k++) {
                if (fabs(a[m][i]) > fabs(a[k][i])) {
                        m = k;
                }
        }

        /* 第 m  行と第  i 行 を入れ替える */
        for(j=1; j<=n+1; j++) {
                key = a[m][j];
                a[m][j] = a[i][j];
                a[i][j] = key;
        }
}

// 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;
                }
```

```
41                 for(k=i+1; k<=n; k++) {
42                         q = a[k][i];
43                         for(j=1; j<=n+1; j++) {
44                                 a[k][j] = a[k][j] - a[i][j] * q;
45                         }
46                 }
47         }
48         return 0;
49 }
50
51 // requires "a" to be an upper triangular matrix
52 // x is the output vector
53 // n is the number of rows
54
55 void inverse_substitution(double a[][N+1], double x[N], int n) {
56         int i,j;
57         double s;
58         /*逆進代入による計算*/
59         for(i=n; i>=1; i--) {
60                 s = 0.0;
61                 for(j=i+1; j<=n; j++) {
62                         s += a[i][j] * x[j];
63                 }
64                 x[i] = a[i][n+1] - s;
65         }
66 }
67 #endif
```

以上のコードで行われていること、特にソースコード 2 に実装されている関数の動作について、例を用いて解説していく。

まず、用いる例として以下の三元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。

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

この (2) の式を用いて、ソースコード 2 のアルゴリズムを説明する。

まずこの (2) 式を (3) 式のように変換し、この行列を $\mathbf{a}$ と表す。

$$\mathbf{a} = \left(\begin{array}{ccc|c} 2 & 3 & -1 & 1 \\ 1 & 1 & 2 & 0 \\ 3 & -1 & 1 & 2 \end{array}\right) \tag{3}$$

ソースコード 1 に $\mathbf{a}$ と $n = 3$ を入力するとまず、ソースコード 2 の gaussian_elimination() が呼び出され、irekae() 関数に $\mathbf{a}$ と n, そして $i = 1$ が入力される。

irekae() 関数では i 行のそれぞれの値について最大値を見つけ、最大値のある m 行と i 行の要素を全て入れ替える。この操作は最小値について行っても以下の動作に影響がない。

実際に $\mathbf{a}$ について $i = 1$ で irekae() 関数を動作させると、1 行目と 3 行目が入れ替わり、以下 (4) の式になる。(ソースコード 29-21)

$$\mathbf{a} = \left(\begin{array}{ccc|c} 3 & -1 & 1 & 2 \\ 1 & 1 & 2 & 0 \\ 2 & 3 & -1 & 1 \end{array}\right) \tag{4}$$

続いて、1 列目に関して $\mathbf{a}(1,1)$ の対角成分以下を掃き出す作業を行う。まず $(1,1)$ 成分を大きさ 1 にするため 1 行目の全要素を $(1,1)$ 成分で割る。(ソースコード 238-40)

この作業の後、1 行目以降の行,k に対して $(k,1)$ 成分が 0 となるよう、1 行目の全要素に $(k,1)$ を掛け算した後、k 行目から引く。これを for で $k = n$ まで繰り返すと以下 (5) の式が得られる。(ソースコード 241-46)

$$\mathbf{a} = \left(\begin{array}{ccc|c} 1 & -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ 0 & -\frac{4}{3} & \frac{5}{3} & -\frac{2}{3} \\ 0 & \frac{11}{3} & -\frac{5}{3} & -\frac{1}{3} \end{array}\right) \tag{5}$$

この操作を $i = 2, i = 3$ と繰り返すことで、以下 (6) のような対角成分が 1 である上三角行列が得られる。

$$\mathbf{a} = \left(\begin{array}{ccc|c} 1 & -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ 0 & 1 & -\frac{5}{11} & -\frac{1}{11} \\ 0 & 0 & 1 & -\frac{6}{25} \end{array}\right) \tag{6}$$

こうすることで例えば z の解は $z = -\frac{6}{25}$ と分かるようになる。ここから x と y も同様に求めていきたいがここでまた新たな関数として inverse_substitution() 関数を用いる。

例えば今 z の解は $z = -\frac{6}{25}$ と分かっているのでこの z の値を式 (6) の (2,3) 成分と掛け合わせた積を (2,4) 成分から引くことで y の値を $y = -\frac{1}{11} - \frac{5}{11} \times \frac{6}{25}$ すなわち $y = -\frac{1}{5}$ と求めることができる。

x でも同様の計算を行えば $x = \frac{17}{25}$ と計算できる。これは逆進代入と呼ばれる計算で実際に inverse_substitution() 関数として実装されている。

2 演習とその結果

演習として教科書に載っていた以下二問を解いた。

$$\begin{cases} 2x & -4y & +6z & = 1 \\ -x & +7y & -8z & = 0 \\ x & +y & -2z & = 3 \end{cases} \tag{7}$$

$$\begin{cases} 2x & +8y & +2z & -3w & =2 \\ 4x & +6y & -2z & -w & =1 \\ 2x & -4y & -2z & -w & =3 \\ x & -5y & +2z & +w & =-2 \end{cases} \tag{8}$$

結果として以下の出力が得られた。

$$\begin{aligned} x(1) &= \quad 1.800000000000000700000000000000000000000000000000000000000000000000000000000000 \\ x(2) &= -1.000000000000000000000000000000000000000000000000000000000000000000000000000000 \\ x(3) &= -1.100000000000000100000000000000000000000000000000000000000000000000000000000000 \end{aligned} \tag{9}$$

$$
\begin{aligned}
x(1) &= -0.16666666666666674000000000000000000000000000000000000000000000000000 \\
x(2) &= -0.16666666666666674000000000000000000000000000000000000000000000000000 \\
x(3) &= -0.54166666666666674000000000000000000000000000000000000000000000000000 \\
x(4) &= -1.58333333333333350000000000000000000000000000000000000000000000000000
\end{aligned}
\tag{10}
$$

この値は少数点以下 14 桁までは教科書記載の模範解答と一致しており、計算ができていることが示されている。