# 4. 連立一次方程式の解法

## 4.1 LU 分解法

同じ係数行列 A（サイズ n×n）をもつ m 組の連立 1 次方程式

$$AX = B$$　　（ただし A=[$a_{ij}$]は n 行 n 列の正則行列，B=[$b_{ij}$]と X=[$x_{ij}$]は n 行 m 列の行列）

を同時に解く。行列 A,B を並置して 1 個の配列 A’（n 行 n+m 列）を作成し，$a_{i,n+j}=b_{ij}$ (i=1,…,n; j=1,…,m) として A’の配列成分を改めて $a_{ij}$ と記す。

$$A'=[A|B]=\left[\begin{array}{cccc|cccc} a_{11} & a_{12} & \cdots & a_{1n} & b_{11} & b_{12} & \cdots & b_{1m} \\ a_{21} & a_{22} & & a_{2n} & b_{21} & b_{22} & & b_{2m} \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_{n1} & b_{n2} & \cdots & b_{nm} \end{array}\right]=\left[\begin{array}{cccc|cccc} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} & a_{1,n+2} & \cdots & a_{1,n+m} \\ a_{21} & a_{22} & & a_{2n} & a_{2,n+1} & a_{2,n+2} & & a_{2,n+m} \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} & a_{n,m+2} & \cdots & a_{n,n+m} \end{array}\right]$$

以下のアルゴリズムを実行すると，$A'$ の $B$ 部に解 $X$ の値が入る。

**【アルゴリズム 4.1.1】**　LU 分解法の素直なアルゴリズム

LU 分解と前進消去の同時進行(原型)：

```
for j=2 to n+m                                  {k=1} {L: l_i1=a_i1}
    a_1j:=a_1j/a_11                             {U: u_1j=a_1j/l_11}  {Y: y_1j=b_1j/l_11 }
end
for k=2 to n                                    {k=2, 3, …, n}
    for i=k to n                                {L: l_ik = a_ik - Σ_{l=1}^{k-1} l_il u_lk }
        for l=1 to k-1
            a_ik:=a_ik-a_il a_lk
        end
    end
    for j=k+1 to n+m                            {U: u_kj = (a_kj-Σ_{l=1}^{k-1} l_kl u_lj)/l_kk }
        for l=1 to k-1                          {Y: y_kj = (b_kj-Σ_{l=1}^{k-1} l_kl y_lj)/l_kk }
            a_kj:=a_kj-a_kl a_lj
        end
        a_kj:=a_kj/a_kk
    end
end
```

後退代入：

```
for k=n-1, n-2, …, 1                            {k=n} {X: x_nj = y_nj }
    for j=1 to m                                {右辺ベクトルの数だけ繰り返す}
        for l=k+1 to n                          {k≠n} {X: x_kj=y_kj - Σ_{l=k+1}^{n} u_kl x_lj }
            a_k,n+j:=a_k,n+j-a_kl a_l,n+j
        end
    end
end
```

**【プログラム 4.1.1】**

```
1 /* Solve Linear Equation System AX=B                    */
2 /*           By LU Decomposition Method                 */
3 /*                    ( Original Algorithm )            */
4 /*   AX = B                                             */
```

```
 5 /*    A : Coefficient Matrix                  size n*n  */
 6 /*    X : Solution Matrix        (m Vectors)  size n*m  */
 7 /*    B : Right-hand-side Matrix (m Vectors)  size n*m  */
 8 /*                                                      */
 9 /*   HERE A'=[A,B]                                      */
10
11 # include <stdio.h>
12
13 #define MMAX 20
14 #define NMAX 30
15
16 void main(void)
17 {
18     float a[NMAX][NMAX+MMAX];
19     int n,m;
20     int i,j,k,l;
21
22 /* Read Data */
23     printf("Input Matrix Size: n, m ? ");          /* 行列データの画面入力 */
24     scanf("%d%d", &n, &m);                         /* サイズ：n, m */
25     printf("\nInput Coefficient Matrix A\n");      /* 行列 A の画面入力 */
26     for(i=0;i<n;i++){
27        printf("? A(%2d, j), j=1,%2d : ", i+1, n);
28        for(j=0; j<n; j++)
29            scanf("%g", &a[i][j]);
30     }
31     printf("\nInput R.H.S. Vectors B\n");           /* 行列 B の画面入力 */
32     for(i=0;i<n;i++){
33        printf("? B(%2d, j), j=1,%2d : ", i+1, m);
34        for(j=0; j<m; j++)
35            scanf("%g", &a[i][n+j]);
36     }
37
38 /* Write Data */
39     printf("\nLinear Equation System: AX=B\n");
40     printf("* Size of Matrix: n,m=%d,%d\n", n,m);
41     printf("* A'=[A|B]\n");  /* 行列 A'=[A|B] の画面への出力 */
42     for(i=0;i<n;i++){
43         for(j=0;j<n+m;j++)
44             printf("%6.2g", a[i][j]);
45         printf("\n");
46     }
47
48 /* LU Decomposition and Forward reduction */
49     for(j=1;j<n+m;j++)
50         a[0][j] /= a[0][0];
51     for(k=1;k<n;k++){
52         for(i=k;i<n;i++)
53             for(l=0;l<k;l++)
54                 a[i][k] -= a[i][l]*a[l][k];
55         for(j=k+1;j<n+m;j++){
```

```
56              for(l=0;l<k;l++)
57                  a[k][j] -= a[k][l]*a[l][j];
58              a[k][j] /= a[k][k];
59          }
60      }
61
62      printf("\nMatrix A' after LU decomposition and Forward reduction\n");   /* Debug Write */
63      for(i=0;i<n;i++){
64          for(j=0;j<n+m;j++)
65              printf("%15.6e", a[i][j]);
66          printf("\n");
67      }
68
69  /* Backward Substitution */
70      for(k=n-1;k>=0;--k){
71          for(j=0;j<m;j++){
72              for(l=k+1;l<n;l++){
73                  a[k][n+j] -= a[k][l]*a[l][n+j];
74              }
75          }
76      }
77
78  /* Print Result */
79      printf("\nSolutions\n");
80      for(i=0;i<n;i++){
81          for(j=0;j<m;j++)
82              printf("%15.6e", a[i][n+j]);
83          printf("\n");
84      }
85  }
```

**【アルゴリズム 4.1.2】**　LU 分解法のコンパクトなアルゴリズム

LU 分解と前進消去の同時進行：

```
for k=1 to n
    for j=k+1 to n+m
        a_kj:=a_kj/a_kk                          {U（ただし対角成分 1 を除く）}
        for i=k+1 to n
            a_ij:=a_ij-a_ik a_kj                 {L, U と Y}
        end
    end
end
```

後退代入：

```
for k=n, n-1, …, 1
    for j=1 to m
        for i=1 to k-1
            a_i,n+j:=a_i,n+j-a_ik a_k,n+j
        end
    end
```

end

次のプログラムでは，入力データをファイルから読んでいる。

**【プログラム4.1.2】**

```
 1 /* Solve Linear Equation System AX=B                    */
 2 /*            By LU Decomposition Method                */
 3 /*                      ( Compact Algorithm )           */
 4 /*   AX = B                                             */
 5 /*    A : Coefficient Matrix                  size n*n  */
 6 /*    X : Solution Matrix        (m Vectors)  size n*m  */
 7 /*    B : Right-hand-side Matrix (m Vectors)  size n*m  */
 8 /*                                                      */
 9 /*   HERE A'=[A,B]                                      */
10 /*                                                      */
11 /*   Input Data is read from File                       */
12
13 # include <stdio.h>
14
15 #define MMAX 20
16 #define NMAX 30
17
18 float a[NMAX][NMAX+MMAX];
19 int   n,m;
20
21 void input(void)                    /* 関数：A, Bの係数をファイルから入力 */
22 {
23     int i,j;
24     FILE *fp;                              /* ファイルポインターの宣言 */
25
26 /* Read Data */
27                                          /* ファイルオープンとエラー処理 */
28     if( (fp=fopen("c4matrix.dat","r"))==NULL ){
29         printf("ファイルをオープンできません。\n");
30         exit(1);
31     }
32                                         /* 行列データのファイル入力 */
33     fscanf(fp,"%d%d",&n, &m);                        /* サイズ：n, m */
34     for(i=0;i<n;i++){                                      /* 行列 A */
35        for(j=0; j<n; j++)
36            fscanf(fp,"%g", &a[i][j]);
37     }
38     for(i=0;i<n;i++){                                      /* 行列 B */
39        for(j=0; j<m; j++)
40            fscanf(fp,"%g", &a[i][n+j]);
41     }
42                                                /* ファイルクローズ */
43     fclose(fp);
44                                         /* 行列データの出力*/
45     printf("Linear Equation System: AX=B\n");
```

```
46      printf("* Size of Matrix: n,m=%d,%d\n", n,m);
47      printf("* A'=[A|B]\n");  /* 行列 A'=[A|B] の画面への出力 */
48      for(i=0;i<n;i++){
49          for(j=0;j<n+m;j++)
50              printf("%6.2g", a[i][j]);
51          printf("\n");
52      }
53  }
54
55  void main(void)                     /* メイン関数 */
56  {
57      int i,j,k,l;
58
59      input();
60
61  /* LU Decomposition and Forward reduction */
62      for(k=0;k<n;k++){
63         for(j=k+1;j<n+m;j++){
64            a[k][j] /= a[k][k];
65            for(i=k+1;i<n;i++){
66               a[i][j] -= a[i][k]*a[k][j];
67            }
68         }
69      }
70
71      printf("\nMatrix A' after LU decomposition and Forward reduction\n");   /* Debug Write */
72      for(i=0;i<n;i++){
73          for(j=0;j<n+m;j++)
74              printf("%15.6e", a[i][j]);
75          printf("\n");
76      }
77
78  /* Backward Substitution */
79      for(k=n-1;k>=0;--k){
80          for(j=0;j<m;j++){
81              for(i=0;i<k;i++){
82                 a[i][n+j] -= a[i][k]*a[k][n+j];
83              }
84          }
85      }
86
87  /* Print Result */
88      printf("\nSolutions\n");
89      for(i=0;i<n;i++){
90          for(j=0;j<m;j++)
91              printf("%15.6e", a[i][n+j]);
92          printf("\n");
93      }
94  }
```

**【実行例】**

問題１）デバッグ（プログラムの誤りを修正すること）用

$$Ax=b \qquad A=\begin{bmatrix}1 & 2 & 3\\ 2 & 7 & 12\\ 1 & 4 & 10\end{bmatrix} \qquad b=\begin{bmatrix}6\\ 21\\ 15\end{bmatrix}$$

問題２）

$$AX=B \qquad A=\begin{bmatrix}2 & 5 & 3 & -1\\ 5 & 2 & 1 & 1\\ 1 & -2 & 3 & -1\\ 2 & 4 & 1 & 5\end{bmatrix} \qquad B=\begin{bmatrix}2 & 1 & 0 & 0 & 0\\ 3 & 0 & 1 & 0 & 0\\ 15 & 0 & 0 & 1 & 0\\ -8 & 0 & 0 & 0 & 1\end{bmatrix}$$

問題１に対する実行例）　プログラム4.1.1（入力データをキーボードから入力）

```
―――――――――――――――――――実行開始―――――――――――――――――――
Input Matrix Size: n, m ? 3 1

Input Coefficient Matrix A
? A( 1, j), j=1, 3 : 1 2 3
? A( 2, j), j=1, 3 : 2 7 12
? A( 3, j), j=1, 3 : 1 4 10

Input R.H.S. Vectors B
? B( 1, j), j=1, 1 : 6
? B( 2, j), j=1, 1 : 21
? B( 3, j), j=1, 1 : 15

Linear Equation System: AX=B
* Size of Matrix: n,m=3,1
* A'=[A|B]
     1     2     3     6
     2     7    12    21
     1     4    10    15

Matrix A' after LU decomposition and Forward reduction
   1.000000e+00   2.000000e+00   3.000000e+00   6.000000e+00
   2.000000e+00   3.000000e+00   2.000000e+00   3.000000e+00
   1.000000e+00   2.000000e+00   3.000000e+00   1.000000e+00

Solutions
   1.000000e+00
   1.000000e+00
   1.000000e+00
―――――――――――――――――――おしまい―――――――――――――――――――
```

問題２に対する実行例）プログラム4.1.2（入力データをファイルから読む）

入力データ（ファイル名：c4matrix.dat）：

```
----------入力データ----------
 4  5
 2  5  3 -1
 5  2  1  1
 1 -2  3 -1
 2  4  1  5
 2  1  0  0  0
 3  0  1  0  0
15  0  0  1  0
-8  0  0  0  1
------------------------------
```

```
----------実行開始----------
Linear Equation System: AX=B
* Size of Matrix: n,m=4,5
* A'=[A|B]
     2     5     3    -1     2     1     0     0     0
     5     2     1     1     3     0     1     0     0
     1    -2     3    -1    15     0     0     1     0
     2     4     1     5    -8     0     0     0     1

Matrix A' after LU decomposition and Forward reduction
  2.000000e+00   2.500000e+00   1.500000e+00  -5.000000e-01   1.000000e+00   5.000000e-01   0.000000e+00   0.000000e+00   0.000000e+00
  5.000000e+00  -1.050000e+01   6.190476e-01  -3.333333e-01   1.904762e-01   2.380952e-01  -9.523810e-02   0.000000e+00   0.000000e+00
  1.000000e+00  -4.500000e+00   4.285714e+00  -4.666667e-01   3.466667e+00   1.333333e-01  -1.000000e-01   2.333333e-01   0.000000e+00
  2.000000e+00  -1.000000e+00  -1.380952e+00   5.022222e+00  -1.000000e+00  -1.150443e-01  -4.646018e-02   6.415930e-02   1.991151e-01

Solutions
  1.000000e+00  -5.309733e-02   2.477876e-01  -8.849561e-03  -6.194691e-02
 -2.000000e+00   1.504425e-01  -3.539823e-02  -1.415929e-01   8.849559e-03
  3.000000e+00   7.964602e-02  -1.216814e-01   2.632743e-01   9.292036e-02
 -1.000000e+00  -1.150443e-01  -4.646018e-02   6.415930e-02   1.991151e-01
----------おしまい----------
```

## 4.2 ガウスの消去法

4.1節と同様，$AX = B$ を解くにあたり，配列を $A'=[A|B]$ にとる。

### 4.2.1 ガウスの消去法（基本）

**【アルゴリズム 4.2.1】**

前進消去：

```
for k=1 to n-1
    for i=k+1 to n
        w:= a_ik／a_kk
        for j=k+1 to n+m                    {j=k は a_ik=0 と自明なので省く }
            a_ij:=a_ij- wa_kj                {a^(k+1)_ij= a^(k)_ij - (a^(k)_ik a^(k)_kk)a^(k)_kj }
        end                                  {b^(k+1)_ij= b^(k)_ij - (a^(k)_ik a^(k)_kk)b^(k)_kj }
    end
end
```

後退代入：

```
for k=n, n-1, …, 1
    for j=1 to m                            {右辺ベクトルの数だけ繰り返す}
        if k<n then                         {k<n: x_kj=(b^(k)_kj - Σ_{l=k+1}^{n} a^(k)_kl x_lj)/a^(k)_kk }
            for l=k+1 to n                  {k=n: x_kj= b^(k)_kj/a^(k)_kk }
                a_k,n+j:=a_k,n+j-a_kl a_l,n+j
            end
        end
        a_k,n+j:=a_k,n+j/a_kk
    end
end
```

後退代入はつぎのようにコンパクトなアルゴリズムに書き換えられる：

```
for k=n, n-1, …, 1
    for j=1 to m                            {右辺ベクトルの数だけ繰り返す}
        a_k,n+j:=a_k,n+j／a_k,k              {x_kj=(b^(k)_kj - Σ_{l=k+1}^{n} a^(k)_kl x_lj)/a^(k)_kk }
        for i=1 to k-1
            a_i,n+j:=a_i,n+j-a_ik a_k,n+j
        end
    end
end
```

**【プログラム 4.2.1】**

LU分解法のプログラム 4.1.2 に記した関数 input を用い，ファイルからデータを入力する。
以下にガウスの消去法（後退代入にはコンパクトなアルゴリズムを使用）の main 関数を記す。他はプログラム 4.1.2 と同じ。

```
 1 void main(void)                  /* メイン関数 */
 2 {
 3     int i,j,k;
 4     float w;
 5
```

```
 6     input();
 7
 8 /* Forward reduction */
 9     for(k=0;k<n-1;k++){
10        for(i=k+1;i<n;i++){
11           w=a[i][k]/a[k][k];
12           for(j=k+1;j<n+m;j++){
13              a[i][j] -= w*a[k][j];
14           }
15        }
16     }
17
18 /* Backward Substitution */
19     for(k=n-1;k>=0;--k){
20         for(j=0;j<m;j++){
21             a[k][n+j] /= a[k][k];
22             for(i=0;i<=k-1;i++){
23                 a[i][n+j] -= a[i][k]*a[k][n+j];
24             }
25         }
26     }
27
28 /* Print Result */
29     printf("\nSolutions\n");
30     for(i=0;i<n;i++){
31         for(j=0;j<m;j++)
32             printf("%15.6e", a[i][n+j]);
33         printf("\n");
34     }
35 }
```

**【実行例】**

教科書の例題4.1を解く。

$$Ax=b \qquad A=\begin{bmatrix}2 & 4 & 2\\ 1 & 3 & 4\\ 3 & 8 & 11\end{bmatrix} \qquad b=\begin{bmatrix}4\\ 7\\ 18\end{bmatrix}$$

入力データ

```
------------------------入力データ-----------------------
  3  1
  2  4  2
  1  3  4
  3  8 11
  4
  7
 18
---------------------------------------------------------

------------------------実行開始-----------------------
Linear Equation System: AX=B
* Size of Matrix: n,m=3,1
* A'=[A|B]
     2     4     2     4
```

```
     1      3      4      7
     3      8     11     18

Solutions
  -3.000000e+00
   2.000000e+00
   1.000000e+00
------------------------おしまい-----------------------
```

### 4.2.2 掃出し法（あるいはガウス・ジョルダン消去法）

**【アルゴリズム 4.2.2】**

掃き出しのプロセス：

```
for k=1 to n
    w:=1/a_kk
    for j=k to n+m                    {a^(k+1)_kj= a^(k)_kj/a^(k)_kk }
        a_kj:=a_kjw                   {b^(k+1)_kj= b^(k)_kj/a^(k)_kk }
    for i=1 to n
        if(i≠k) then
            w:= a_ik
            for j=k to n+m
                a_ij:=a_ij- wa_kj     {a^(k+1)_ij= a^(k)_ij - a^(k)_ik a^(k+1)_kj }
            end                       {b^(k+1)_ij= b^(k)_ij - a^(k)_ik b^(k+1)_kj }
        end
    end
end
```

**【プログラム 4.2.2】**

LU 分解法のプログラム 4.1.2 に記した関数 input を用い，ファイルからデータを入力する。
以下に main 関数を記す（他はプログラム 4.1.2 と同じ）。

```
 1 void main(void)          /* メイン関数 */
 2 {
 3     int i,j,k;
 4     float w;
 5
 6     input();
 7
 8 /* Sweep-out */
 9     for(k=0;k<n;k++){
10        w = 1/a[k][k];
11        for(j=k;j<(n+m);j++)
12           a[k][j] *= w;
13        for(i=0;i<n;i++){
14            if(i!=k){
15               w = a[i][k];
16               for(j=k;j<(n+m);j++)
17                   a[i][j] -= w*a[k][j];
18            }
19        }
20     }
21
```

```
22 /* Print Result */
23     printf("\nSolutions\n");
24     for(i=0;i<n;i++){
25         for(j=0;j<m;j++)
26             printf("%15.6e", a[i][n+j]);
27         printf("\n");
28     }
29 }
```

**【実行例】**

LU 分解法の実行例で記した問題 2 を解く。

```
-------------------------実行開始-----------------------
Linear Equation System: AX=B
* Size of Matrix: n,m=4,5
* A'=[A|B]
     2     5     3    -1     2     1     0     0     0
     5     2     1     1     3     0     1     0     0
     1    -2     3    -1    15     0     0     1     0
     2     4     1     5    -8     0     0     0     1

Solutions
   1.000000e+00  -5.309733e-02   2.477876e-01  -8.849546e-03  -6.194691e-02
  -2.000000e+00   1.504425e-01  -3.539823e-02  -1.415929e-01   8.849557e-03
   3.000000e+00   7.964602e-02  -1.216814e-01   2.632743e-01   9.292036e-02
  -1.000000e+00  -1.150443e-01  -4.646018e-02   6.415930e-02   1.991151e-01
-------------------------おしまい-----------------------
```

**4.2.3 ピボット選択付ガウス消去法（あるいはピボット選択付 LU 分解法（ただし L の対角成分はすべて 1））**

**【アルゴリズム 4.2.3】**

LU 分解と前進消去：

```
for i=1 to n
    p_i=i                                          {置換情報 p の初期設定}
for k=1 to n-1
    i=k,k+1,…n に対し，|a_ik|>|a_kk|となる r を探す
    if r≠k then                                    {k 行と r 行の p,L,A,B の値を入れ換える}
        p_k と p_r の値を入れ換える
        k 行の a_kj と r 行の a_rj の値を入れ換える（j=1,2,…,n+m）
    end
    for i=k+1 to n
        a_ik:= a_ik／a_kk                          {L（ただし対角成分 1 を除く）}
        for j=k+1 to n+m
            a_ij:=a_ij- a_ika_kj                   {L,U と Y}
        end
    end
end
```

後退代入は，アルゴリズム 4.2.1 におけるものと同じ

**【プログラム 4.2.3】**

LU 分解法のプログラム 4.1.2 に記した関数 input を用い，ファイルからデータを入力する。
以下に main 関数を記す（他はプログラム 4.1.2 と同じ）。

```
 1 void main(void)                                                    /* メイン関数 */
 2 {
 3     int p[NMAX],i,j,k,r;
 4     float amax,w,akk1,ptemp,atemp;
 5
 6     input();
 7
 8     for(i=0;i<n;i++)                /* Index for Pivoting */
 9         p[i]=i;
10                         /* Forward Substitution with Pivoting */
11     for(k=0;k<n-1;k++){
12         amax=fabs(a[k][k]);
13         r=k;
14         for(i=k+1;i<n;i++){      /* search r such that a_{r,k}=max a_{i,k} i=k,..n */
15             if( fabs(a[i][k]) > amax ){
16                 amax=fabs(a[i][k]);
17                 r=i;
18             }
19         }
20         if(r!=k){      /* exchange column r and column k */
21             ptemp=p[r];
22             p[r] =p[k];
23             p[k] =ptemp;
24             for(j=0;j<n+m;j++){
25                 atemp   = a[r][j];
26                 a[r][j] = a[k][j];
27                 a[k][j] = atemp;
28             }
29         }
30         for(i=k+1;i<n;i++)
31                 a[i][k] = a[i][k]/a[k][k];
32         for(i=k+1;i<n;i++){
33                 for(j=k+1;j<n+m;j++)
34                     a[i][j] -= a[i][k]*a[k][j];
35             }
36     }
37                                          /* Backward Substitution */
38     for(k=n-1;k>=0;--k){
39         akk1=1/a[k][k];
40         for(j=0;j<m;j++){
41             for(i=k+1;i<n;i++){
42                 a[k][n+j] -= a[k][i]*a[i][n+j];
43             }
44             a[k][n+j] *=akk1;
45         }
46     }
47                                      /* print result */
48     printf("\nPermutation\n");
49     for(i=0;i<n;i++){
50         printf("%3d", p[i]);
51         if(i%10==9) printf("\n");
52     }
```

```
53      printf("\n");
54      printf("\nLU decomposition\n");
55      for(i=0;i<n;i++){
56          for(j=0;j<n;j++)
57              printf("%15.6e", a[i][j]);
58          printf("\n");
59      }
60      printf("\nSolution\n");
61      for(i=0;i<n;i++){
62          for(j=0;j<m;j++)
63              printf("%15.6e", a[i][n+j]);
64          printf("\n");
65      }
66  }
```

**【実行例】**

ガウスの消去法（基本）における実行例で扱った問題（教科書例題4.1）を解く。

```
-------------------------実行開始-----------------------
Linear Equation System: AX=B
* Size of Matrix: n,m=3,1
* A'=[A|B]
     2     4     2     4
     1     3     4     7
     3     8    11    18

Permutation
  2  0  1

LU decomposition
   3.000000e+00   8.000000e+00   1.100000e+01
   6.666667e-01  -1.333333e+00  -5.333333e+00
   3.333333e-01  -2.499999e-01  -9.999996e-01

Solution
  -3.000000e+00
   2.000000e+00
   9.999998e-01
-------------------------おしまい-----------------------
```

## 4.3 三項方程式の解法

三項方程式

$$c_i x_{i-1} + a_i x_i + b_i x_{i+1} = d_i \qquad (i=1, 2, \cdots, n)$$

は，つぎのように行列表示される。

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

$$A = \begin{bmatrix} a_1 & b_1 & & & & \\ c_2 & a_2 & b_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & & & c_n & a_n \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_{n-1} \\ d_2 \end{bmatrix}$$

これは 4.1，4.2 節に示した数値解法により解くことができる。つぎのアルゴリズムは 4.1 節の LU 分解によるものであり，これを実行すると右辺ベクトルの $\boldsymbol{d}$ 部に解 $\boldsymbol{x}$ の値が入る。なお，$c_1$, $b_n$の値は使用しないが，通常 $c_1=0$, $b_n=0$ を入れておく。

**【アルゴリズム 4.3.1】**

LU 分解と前進消去の同時進行：

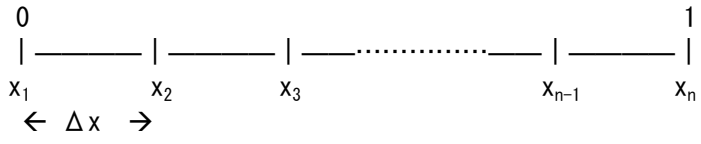
```
b1:=b1/a1
d1:=d1/a1
for i=2 to n
    ai:=ai-ci*bi-1
    di:=(di-ci*hi-1)/ai
    bi:=bi/ai
end
```

後退代入：

```
for i=n-1, n-2, …, 1
    di:=di-bi*di+1
end
```

**【問題】**　常微分方程式 $d^2u/dx^2=1$ を条件 $u(0)=0$, $u(1)=0$ のもとで解くと，解 $u(x)=0.5x(x-1)$ を得るが，これを差分法で求め，差分近似解と誤差を出力する。

x の区間[0,1]を n-1 等分して各小区間幅を Δx とし，$x_i=(i-1)\Delta x$, i=1,2,…, n,（ただし $\Delta x=1/(n-1)$）とおく。




2 階導関数は

$$u''(x_i) \fallingdotseq \left( u(x_i+\Delta x) - 2\,u(x_i) + u(x_i-\Delta x) \right)/(\Delta x)^2$$

と近似される（第 9 章：数値微分を参照のこと）ので，$U_i=u(x_i)$ と記せば，i=2,3,…,n-1 に対し

$$U_{i+1} - 2\,U_i + U_{i-1} = (\Delta x)^2$$

が成立し，i=1 と i=n に対しては以下の条件が課せられる。

$$U_1 = 0$$
$$U_n = 0$$

n 個の未知数 $U_i$, i=1,2,…,n に対し，代数方程式が n 個得られたので，連立一次方程式により $U_i$ が求められる。具体的に代数式を列挙すると

$x=x_1=0$　のとき　$U_1 = 0$

$x=x_2$　のとき　$U_1 - 2U_2 + U_3 = (\Delta x)^2$

$x=x_3$　のとき　$U_2 - 2U_3 + U_4 = (\Delta x)^2$

$x=x_4$　のとき　$U_3 - 2U_4 + U_5 = (\Delta x)^2$

：　　：

$x=x_{n-1}$　のとき　$U_{n-2} - 2U_{n-1} + U_n = (\Delta x)^2$

$x=x_n=1$　のとき　$U_n = 0$

となり，三項方程式の形をしていることがわかる。

**【プログラム 4.3.1】**

```
 1 /* three-term equations */
 2
 3 #include <stdio.h>
 4
 5 #define NMAX 101
 6
 7 void main(void)
 8 {
 9     float a[NMAX],b[NMAX],c[NMAX],d[NMAX],
10           xmin=0.0,xmax=1.0,dx,x,error;
11     int   n,i;
12
13     printf("number of points: n ? ");
14     scanf("%d", &n);
15     dx=(xmax-xmin)/(n-1);
16                                                    /* data */
17     c[0]= 0.0;
18     a[0]= 1.0;
19     b[0]= 0.0;
20     d[0]= 0.0;
21     for(i=1;i<n-1;i++){
22         c[i]= 1.0;
23         a[i]=-2.0;
24         b[i]= 1.0;
25         d[i]= dx*dx;
26     }
27     c[n-1]= 0.0;
28     a[n-1]= 1.0;
29     b[n-1]= 0.0;
30     d[n-1]= 0.0;
31
32     printf("\nThree-Term equations:\n i,c[i],a[i],b[i],d[i]\n"); /* output */
33     for(i=0;i<n;i++)
34         printf("%2d %g %g %g %g\n", i,c[i], a[i], b[i], d[i]);
35                                                              /* forward */
36     b[0] /= a[0];
37     d[0] /= a[0];
38     for(i=1;i<n;i++){
39         a[i] -= c[i]*b[i-1];
40         d[i] -= c[i]*d[i-1];
41         d[i] /= a[i];
42         b[i] /= a[i];
43    }
```

```
44                                                          /* backward */
45      for(i=n-2;i>=0;i--)
46          d[i] -=b[i]*d[i+1];
47
48      printf("\n i \tx \t Solution\t Error\n");
49      for(i=0;i<n;i++){
50          x=i*dx;
51          error=0.5*x*(x-1)-d[i];
52          printf("%2d\t%15.6g\t%15.6g\t%15.6e\n",  i,x,d[i],error);
53      }
54 }
```

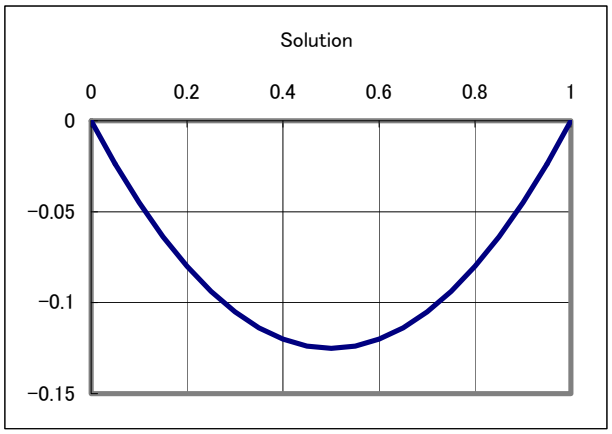
**【実行例】**

```
------------------------実行開始-----------------------
number of points: n ? 21

Three-Term equations:
 i,c[i],a[i],b[i],d[i]
 0 0 1 0 0
 1 1 -2 1 0.0025
 2 1 -2 1 0.0025
 3 1 -2 1 0.0025
 4 1 -2 1 0.0025
 5 1 -2 1 0.0025
 :
 :
18 1 -2 1 0.0025
19 1 -2 1 0.0025
20 0 1 0 0

 i       x        Solution         Error
 0               0               0     0.000000e+00
 1            0.05        -0.02375    -2.495944e-09
 2             0.1          -0.045    -6.258488e-09
 3            0.15        -0.06375    -9.424984e-09
 4             0.2           -0.08    -1.758337e-08
 5            0.25        -0.09375    -2.328306e-08
 6             0.3          -0.105    -2.652407e-08
 7            0.35        -0.11375    -1.985580e-08
 8             0.4           -0.12    -1.817942e-08
 9            0.45        -0.12375    -2.149492e-08
10             0.5          -0.125    -2.235174e-08
11            0.55        -0.12375    -2.074987e-08
12             0.6           -0.12    -1.668930e-08
13            0.65        -0.11375    -1.017004e-08
14             0.7          -0.105    -8.642673e-09
15            0.75        -0.09375    -4.656613e-09
16             0.8           -0.08    -5.662441e-09
17            0.85        -0.06375    -4.209578e-09
18             0.9          -0.045    -4.023313e-09
19            0.95¥        -0.02375     2.346933e-09
20               1               0     7.450581e-09
------------------------おしまい-----------------------
```

近似解のグラフ



## 4.4 反復法

### 4.4.1 ヤコビ法

**【アルゴリズム 4.4.1】**

```
for i=1 to n                        {初期値の設定}
    x_i := 与えられた初期値
end
for k=0,1,2,…                       {収束するまで反復する}
    for i=1 to n
        x^temp:=b_i
        for j=1 to n
            if(i≠j) then
                x^temp:=x^temp-a_ij x_i
            end
        end
        x^temp:=x^temp/a_ii
        x^new_i:=x^temp
    end
                        ←=====①
    for i=1 to n
        x_i:=x^new_i
    end
                        ←=====②
end
```

収束判定は1ノルムを用いて行い，解の変化量が次式を満たすとき収束したとする。

$$\frac{\| \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \|_1}{\| \boldsymbol{x}^{(k+1)} \|_1} = \frac{\sum_{i=1}^{n} | x_i^{(k+1)} - x_i^{(k)} |}{\sum_{i=1}^{n} | x_i^{(k+1)} |} < \varepsilon$$

アルゴリズムの①のところに分子・分母のノルムの計算（下記）を挿入する

```
norm_dx:=0
norm_x :=0
for i=1 to n
    norm_dx:=norm_dx + |x^newi-xi|
    norm_x :=norm_x  + |x^newi|
end
```

あるいは上記を，直前の(for i=1 to n)のループと同時進行させる

アルゴリズムの②のところに次の収束判定を入れる

```
If norm_dx/norm_x < ε    break（for k=0,1,2,…のループから出る）
```

**【プログラム 4.4.1】**

以下に示すプログラムでは，関数 input1 においてファイルからデータを入力する。

```
1 /* Solve Linear Equation System Ax=b */
2 /*          by Jacobi Method         */
3
4 #include <stdio.h>
5 #include <math.h>
```

```
 6
 7 #define KMAX 50
 8 #define NMAX 20
 9
10 float eps, a[NMAX][NMAX], b[NMAX];
11 int n;
12
13 void input1(void)                    /* 入力データを読み込む関数 */
14 {
15     int i, j;
16     FILE *fp;                              /* ファイルポインターの宣言 */
17                                          /* ファイルオープンとエラー処理 */
18     if( (fp=fopen("c4iter.dat","r"))==NULL ){
19         printf("ファイルをオープンできません。\n");
20         exit(1);
21     }
22
23     fscanf(fp,"%g %d", &eps, &n);       /* 収束判定許容誤差と行列サイズの入力 */
24     for(i=0;i<n;i++){                              /* 行列データのファイル入力 */
25         for(j=0;j<n;j++)
26             fscanf(fp,"%g",&a[i][j]);
27     }
28     for(i=0;i<n;i++)                             /* 右辺ベクトルのファイル入力 */
29             fscanf(fp,"%g",&b[i]);
30     fclose(fp);                                          /* ファイルクローズ */
31
32     printf("Solve Linear Equation System: Ax=b\n");
33     printf("* Matrix size: n=%d \t Tolerance: eps=%g\n\n",n,eps); /* 画面への出力 */
34     printf("* A'=[A|b]\n");                        /* 行列 A'=[A|b] の画面への出力 */
35     for(i=0;i<n;i++){
36         for(j=0;j<n;j++)
37             printf("%15.6e", a[i][j]);
38         printf("%15.6e\n", b[i]);
39     }
40     printf("\n");
41 }
42
43 void main(void)                     /* メイン関数 */
44 {
45     int i, j, k;
46     float dxnorm, xnorm,xx,x[NMAX],xnew[NMAX];
47
48     input1();
49
50     printf("Iteration by Jacobi Method\n");
51                                          /* initial value */
52     for(i=0;i<n;i++)
53         x[i]=0;
54                                          /* iteration */
55     for(k=0;k<KMAX;k++){
56         printf("k=%d\t",k);
57         for(i=0;i<n;i++)
58             printf("%15.6e", x[i]);
59         printf("\n");
60         dxnorm=0;
61         xnorm =0;
62         for(i=0;i<n;i++){
```

```
63              xx=b[i];
64              for(j=0;j<n;j++)
65                  if(i!=j) xx -= a[i][j]*x[j];
66              xx /= a[i][i];
67              dxnorm += fabs(xx-x[i]);
68              xnorm  += fabs(xx);
69              xnew[i]= xx;
70          }
71          for(i=0;i<n;i++)
72              x[i]=xnew[i];
73          if(dxnorm/xnorm<eps) break;
74      }
75      if(k>=KMAX)
76          printf("not convergent?\n");
77                                          /* output solution */
78      printf("\nSolution:");
79      for(i=0;i<n;i++)
80          printf("%15.6g",x[i]);
81      printf("\n");
82  }
```

### 4.4.2 ガウス・ザイデル法

**【アルゴリズム 4.4.2】**

```
for i=1 to n                    {初期値の設定}
    x_i := 与えられた初期値
end
for k=0,1,2,…                   {収束するまで反復する}
    for i=1 to n
        x^temp:=b_i
        for j=1 to n
            if(i≠j) then
                x^temp:=x^temp-a_ij x_i
            end
        end
        x^temp:=x^temp/a_ii
        x_i:=x^temp
    end
end
```

**【プログラム 4.4.2】**

ヤコビ法のプログラム 4.4.1 に記した関数 input1 を用い，ファイルからデータを入力する。
以下に main 関数を記す（他はプログラム 4.4.1 と同じ）。

```
1  void main(void)                  /* メイン関数 */
2  {
3      int i, j, k;
4      float dxnorm, xnorm,xx,x[NMAX];
5
6      input1();
7
8      printf("Iteration by Gauss-Seidel Method\n");
9                                       /* initial value */
```

```
10      for(i=0;i<n;i++)
11          x[i]=0;
12                                          /* iteration */
13      for(k=0;k<KMAX;k++){
14          printf("k=%d\t",k);
15          for(i=0;i<n;i++)
16              printf("%15.6e", x[i]);
17          printf("\n");
18          dxnorm=0;
19          xnorm =0;
20          for(i=0;i<n;i++){
21              xx=b[i];
22              for(j=0;j<n;j++)
23                  if(i!=j) xx -= a[i][j]*x[j];
24              xx /= a[i][i];
25              dxnorm += fabs(xx-x[i]);
26              xnorm  += fabs(xx);
27              x[i]=xx;
28          }
29          if(dxnorm/xnorm<eps) break;
30      }
31      if(k>=KMAX)
32          printf("not convergent?\n");
33                                          /* output solution */
34      printf("\nSolution:");
35      for(i=0;i<n;i++)
36          printf("%15.6g",x[i]);
37      printf("\n");
38  }
```

### 4.4.3 SOR法（加速緩和法）

**【アルゴリズム 4.4.3】**

```
for i=1 to n                    {初期値の設定}
    x_i := 与えられた初期値
end
for k=0,1,2,…                   {収束するまで反復する}
    for i=1 to n
        x^temp:=b_i
        for j=1 to n
            if(i≠j) then
                x^temp:=x^temp-a_ij x_i
            end
        end
        x^temp:=x^temp/a_ii
        x_i:=x_i + ω(x^temp - x_i )
    end
end
```

**【プログラム 4.4.3】**

ヤコビ法のプログラム 4.4.1 に記した関数 input1 を用い，ファイルからデータを入力する。
以下に main 関数を記す（他はプログラム 4.4.1 と同じ）。

```
 1 void main(void)                    /* メイン関数 */
 2 {
 3     int i, j, k;
 4     float dxnorm, xnorm,xx,dx,x[NMAX];
 5     float omega=1.2;
 6
 7     input1();
 8
 9     printf("Iteration by SOR\n");
10                                          /* initial value */
11     for(i=0;i<n;i++)
12         x[i]=0;
13                                          /* iteration */
14     for(k=0;k<KMAX;k++){
15         printf("k=%d\t",k);
16         for(i=0;i<n;i++)
17             printf("%15.6e", x[i]);
18         printf("\n");
19         dxnorm=0;
20         xnorm =0;
21         for(i=0;i<n;i++){
22             xx=b[i];
23             for(j=0;j<n;j++)
24                 if(i!=j) xx-= a[i][j]*x[j];
25             xx /= a[i][i];
26             dx  = xx-x[i];
27             xx  = x[i] + omega*dx;
28             dxnorm += fabs(dx);
29             xnorm  += fabs(xx);
30             x[i]=xx;
31         }
32         if(dxnorm/xnorm<eps) break;
33     }
34     if(k>=KMAX)
35         printf("not convergent?\n");
36                                          /* output solution */
37     printf("\nSolution:");
38     for(i=0;i<n;i++)
39         printf("%15.6g",x[i]);
40     printf("\n");
41 }
```

**【実行例】**

次の問題を，ヤコビ法，ガウス・ザイデル法，SOR 法で解け。

$$Ax = b \qquad A = \begin{bmatrix} -5 & 1 & -2 & 1 \\ -1 & 5 & 1 & 1 \\ 1 & 2 & 5 & 2 \\ 2 & 1 & -1 & -4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ 6 \\ -2 \\ -4 \end{bmatrix}$$

入力データ（ファイル名：c4iter.dat）：

```
------------------------入力データ-----------------------
  1.0e-6
```

```
   4
  -5   1  -2   1
  -1   5   1   1
   1   2   5   2
   2   1  -1  -4
   9
   6
  -2
  -4
```

---

実行例１）ヤコビ法

```
-------------------------実行開始-----------------------
Solve Linear Equation System: Ax=b
* Matrix size: n=4        Tolerance: eps=1e-06

* A'=[A|b]
  -5.000000e+00   1.000000e+00  -2.000000e+00   1.000000e+00   9.000000e+00
  -1.000000e+00   5.000000e+00   1.000000e+00   1.000000e+00   6.000000e+00
   1.000000e+00   2.000000e+00   5.000000e+00   2.000000e+00  -2.000000e+00
   2.000000e+00   1.000000e+00  -1.000000e+00  -4.000000e+00  -4.000000e+00

Iteration by Jacobi Method
k=0         0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
k=1        -1.800000e+00   1.200000e+00  -4.000000e-01   1.000000e+00
k=2        -1.200000e+00   7.200000e-01  -9.200000e-01   5.000001e-01
k=3        -1.188000e+00   1.044000e+00  -6.480000e-01   8.100000e-01
k=4        -1.170000e+00   9.300000e-01  -9.040000e-01   8.290000e-01
k=5        -1.086600e+00   9.810000e-01  -8.696000e-01   8.735000e-01
k=6        -1.081260e+00   9.819000e-01  -9.244800e-01   9.193500e-01
k=7        -1.049958e+00   9.847740e-01  -9.442480e-01   9.359650e-01
k=8        -1.038153e+00   9.916650e-01  -9.583040e-01   9.572765e-01
k=9        -1.026890e+00   9.925749e-01  -9.719460e-01   9.684158e-01
k=10       -1.019023e+00   9.953280e-01  -9.790183e-01   9.776852e-01
 :
 :
k=34       -1.000006e+00   9.999985e-01  -9.999936e-01   9.999931e-01
k=35       -1.000004e+00   9.999989e-01  -9.999955e-01   9.999951e-01
k=36       -1.000003e+00   9.999992e-01  -9.999967e-01   9.999964e-01

Solution:              -1        0.999999       -0.999998        0.999997
-------------------------おしまい-----------------------
```

実行例２）ガウス・ザイデル法

```
-------------------------実行開始-----------------------
Solve Linear Equation System: Ax=b
* Matrix size: n=4        Tolerance: eps=1e-06

* A'=[A|b]
  -5.000000e+00   1.000000e+00  -2.000000e+00   1.000000e+00   9.000000e+00
  -1.000000e+00   5.000000e+00   1.000000e+00   1.000000e+00   6.000000e+00
   1.000000e+00   2.000000e+00   5.000000e+00   2.000000e+00  -2.000000e+00
   2.000000e+00   1.000000e+00  -1.000000e+00  -4.000000e+00  -4.000000e+00

Iteration by Gauss-Seidel Method
```

```
k=0        0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
k=1       -1.800000e+00   8.400000e-01  -3.760000e-01   4.040000e-01
k=2       -1.400800e+00   9.142400e-01  -6.471360e-01   6.899440e-01
k=3       -1.220309e+00   9.473767e-01  -8.108665e-01   8.294064e-01
k=4       -1.120297e+00   9.722327e-01  -8.965963e-01   9.070589e-01
k=5       -1.065503e+00   9.848068e-01  -9.436457e-01   9.493616e-01
k=6       -1.035708e+00   9.917152e-01  -9.692891e-01   9.723970e-01
k=7       -1.019462e+00   9.954860e-01  -9.832609e-01   9.849558e-01
k=8       -1.010607e+00   9.975396e-01  -9.908767e-01   9.918004e-01
k=9       -1.005781e+00   9.986590e-01  -9.950275e-01   9.955310e-01
k=10      -1.003151e+00   9.992691e-01  -9.972898e-01   9.975643e-01
 :
 :
k=20      -1.000007e+00   9.999983e-01  -9.999937e-01   9.999944e-01
k=21      -1.000004e+00   9.999991e-01  -9.999966e-01   9.999970e-01
k=22      -1.000002e+00   9.999995e-01  -9.999982e-01   9.999983e-01

Solution:             -1              1       -0.999999        0.999999
------------------------おしまい-----------------------
```

実行例３）SOR 法：　収束加速パラメータ $\omega$=1.2 の場合

```
------------------------実行開始-----------------------
Solve Linear Equation System: Ax=b
* Matrix size: n=4        Tolerance: eps=1e-06

* A'=[A|b]
  -5.000000e+00   1.000000e+00  -2.000000e+00   1.000000e+00   9.000000e+00
  -1.000000e+00   5.000000e+00   1.000000e+00   1.000000e+00   6.000000e+00
   1.000000e+00   2.000000e+00   5.000000e+00   2.000000e+00  -2.000000e+00
   2.000000e+00   1.000000e+00  -1.000000e+00  -4.000000e+00  -4.000000e+00

Iteration by SOR
k=0        0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
k=1       -2.160000e+00   9.216000e-01  -4.039680e-01   3.016704e-01
k=2       -1.240510e+00   9.825090e-01  -7.178900e-01   9.054794e-01
k=3       -1.114194e+00   9.310703e-01  -9.505594e-01   9.148769e-01
k=4       -1.037866e+00   1.013262e+00  -9.663071e-01   9.881760e-01
k=5       -1.008254e+00   9.901180e-01  -9.943387e-01   9.927492e-01
k=6       -1.005178e+00   1.001115e+00  -9.969443e-01   9.977609e-01
k=7       -1.000701e+00   9.994128e-01  -9.990863e-01   9.995770e-01
k=8       -1.000541e+00   9.998699e-01  -9.997874e-01   9.996573e-01
k=9       -1.000107e+00   1.000031e+00  -9.998674e-01   9.999738e-01
k=10      -1.000041e+00   9.999583e-01  -9.999841e-01   9.999635e-01
k=11      -1.000018e+00   1.000009e+00  -9.999856e-01   9.999947e-01
k=12      -1.000002e+00   9.999955e-01  -9.999976e-01   9.999976e-01
k=13      -1.000002e+00   1.000000e+00  -9.999989e-01   9.999989e-01

Solution:             -1              1             -1              1
------------------------おしまい-----------------------
```

## 参考文献

高倉葉子：数値計算の基礎―解法と誤差―，コロナ社（2007）
森口繁一，伊理正夫，武市正人 編：Ｃによる算法痛論，東京大学出版会（2000）
Heath, Michael T.: Scientific Computing, An Introductory Survey, McGraw-Hill (2002)