LU分解を用いた連立一次方程式

LU分解

- 連立一次方程式を解く手段
- 通常，連立方程式を解く場合は変数を減らしていくという方針で解いていくが，これもその1つ
- Gaussの消去法とLU分解は別物
  - 計算機で実装した場合Gaussの消去法はLU分解に比べ計算速度がかなり「遅い」
  - 逆行列を求めるやり方も「遅い」

連立一次方程式の行列表現

- 一般に連立一次方程式は

$$A_{0,0}x_0+A_{0,1}x_1\cdots A_{0,n-1}x_{n-1} = b_0$$
$$A_{1,0}x_0+A_{1,1}x_1\cdots A_{1,n-1}x_{n-1} = b_1$$

$$A_{n-1,0}x_0+A_{n-1,1}x_1\cdots A_{n-1,n-1}x_{n-1} = b_{n-1} \quad (1)$$

- これを行列表現すると,

$$\begin{pmatrix} A_{0,0} & A_{0,1} & \cdots & A_{0,n-1} \\ A_{1,0} & A_{1,1} & \cdots & A_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ A_{n-1,0} & A_{n-1,1} & \cdots & A_{n-1,n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix}$$

$$Ax=b \quad (2)$$

上三角行列，下三角行列

- LU分解では，行列Aを下三角行列Lと上三角行列Uの二つに分解する
- 三角行列であれば解を求めることが容易

$$
A = \begin{pmatrix}
A_{0,0} & A_{0,1} & \cdots & A_{0,n-1} \\
\vdots & \vdots & \ddots & \vdots \\
A_{1,0} & A_{1,1} & \cdots & A_{1,n-1} \\
\vdots & \vdots & \ddots & \vdots \\
A_{n-1,0} & A_{n-1,1} & \cdots & A_{n-1,n-1}
\end{pmatrix}
$$

$$
= \begin{pmatrix}
1 & 0 & \cdots & 0 & 0 \\
l_{1,0} & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & & \vdots & \vdots \\
l_{n-2,0} & l_{n-2,1} & \cdots & 1 & 0 \\
l_{n-1,0} & l_{n-1,1} & \cdots & l_{n-1,n-2} & 1
\end{pmatrix}
\begin{pmatrix}
u_{0,0} & u_{0,1} & \cdots & u_{0,n-2} & u_{0,n-1} \\
0 & u_{1,1} & \cdots & u_{1,n-2} & u_{1,n-1} \\
\vdots & \vdots & & \vdots & \vdots \\
0 & 0 & \cdots & u_{n-2,n-2} & u_{n-2,n-1} \\
0 & 0 & \cdots & 0 & u_{n-1,n-1}
\end{pmatrix}
$$

$$
= LU \qquad (3)
$$

LU分解で連立一次方程式を解く(1)

- 一次方程式 $Ax = b$ を解く
- 行列Aを下三角行列Lと上三角行列Uの二つにLU分解する

  $A = LU$
- 行列AがLU分解されると求める方程式は以下のようになる

  $Ax = LUx = b$

LU分解で連立一次方程式を解く(2)

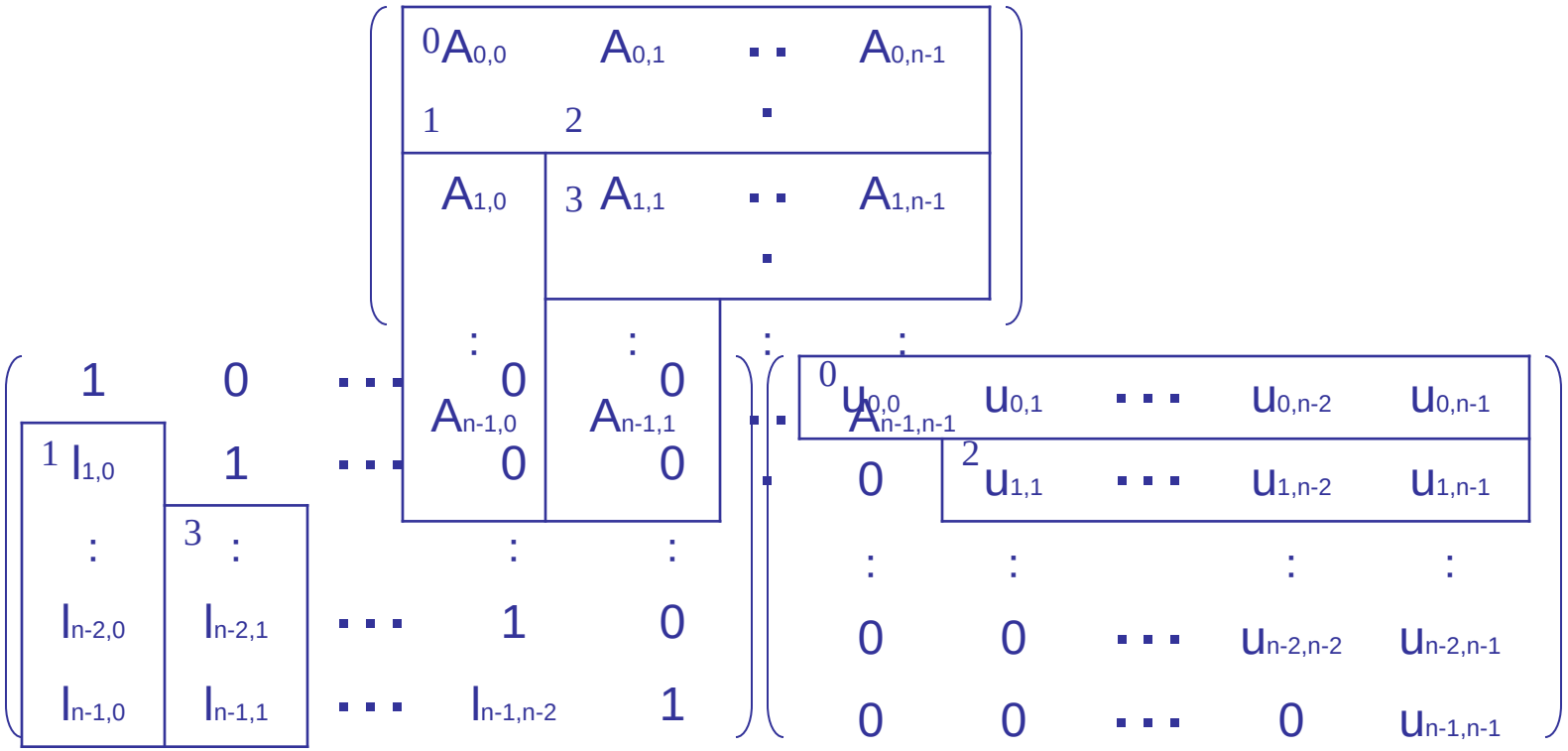
- $y = Ux$とおいて,まず

  $Ly = b$

  を解く. (三角行列を係数とする方程式は容易に解くことができる)
- yを求めたら

  $Ux = y$

  を解くことで,xを求めることができる



$$u_{0,k} = A_{0,k} \quad \text{where}(0 \leqq k \leqq n-1)$$
$$l_{i,0} = A_{i,0}/u_{0,0} \quad \text{where}(1 \leqq i \leqq n-1)$$
$$u_{1,k} = A_{1,k} - u_{0,k} \cdot l_{1,0} \quad \text{where}(1 \leqq k \leqq n-1) \qquad (4)$$

ここで

$$LUx = b \qquad (5)$$
$$Lc = b \qquad (6)$$

となるcを求める.

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ l_{1,0} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ l_{n-2,0} & l_{n-2,1} & \cdots & 1 & 0 \\ l_{n-1,0} & l_{n-1,1} & \cdots & l_{n-1,n-2} & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix} \quad (7)$$

これをcについて解くと

$$c_0 = b_0$$
$$c_1 = b_1 - l_{1,0} \cdot c_0$$
$$c_2 = b_2 - l_{2,0} \cdot c_0 - l_{2,1} \cdot c_1$$
$$\vdots$$
$$c_k = \sum_{i=0}^{k-1} l_{k,i} \cdot c_i \qquad \text{where}(0 \leqq k \leqq n-1) \quad (8)$$

最後にUx = cをxについて解くと

$$\begin{pmatrix} u_{0,0} & u_{0,1} & \cdots & u_{0,n-2} & u_{0,n-1} \\ 0 & u_{1,1} & \cdots & u_{1,n-2} & u_{1,n-1} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & u_{n-2,n-2} & u_{n-2,n-1} \\ 0 & 0 & \cdots & 0 & u_{n-1,n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} \quad (9)$$

より

$$x_{n-1} = c_{n-1}/u_{n-1,n-1}$$

$$x_{n-2} = (c_{n-2} - u_{n-2,n-1} \cdot x_{n-1})/u_{n-2,n-2}$$

$$x_{n-3} = (c_{n-3} - u_{n-3,n-2} \cdot x_{n-2} - u_{n-3,n-1} \cdot x_{n-1})/u_{n-3,n-3}$$

$$\vdots$$

$$x_{n-k} = (c_{n-k} - \sum_{i=1}^{k-1} u_{n-k,n-i} \cdot x_{n-i})/u_{n-k,n-k} \quad \text{where}(1 \leqq k \leqq n) \qquad (10)$$

プログラム例

```
#include <stdio.h>
#define M 4

main(int argc, char *argv[])
{
  double a[M][M];
  double b[M];
  double c[M];
  double l[M][M];
  double u[M][M];
  double x[M];
  int i,j,k;
  FILE *fp;

  fp = fopen(argv[1], "r");          /* ファイルより入力行列を読み込む */
  for(i = 0; i < M; i++){
    for(j = 0; j < M; j++){
      fscanf(fp,"%lf",&a[i][j]);
    }
    fscanf(fp,"%lf",&b[i]);
  }
  fclose(fp);
```

```
for(i = 0; i < M; i++){              /* L行列、U行列を1と0で初期化 */
  for(j = 0; j < M; j++){
    u[i][j] = 0;
    if(i == j)
              l[i][j] = 1;
    else
              l[i][j] = 0;
  }
}

for(i = 0; i < M; i++){
  for(j = i; j < M; j++){            /* U行列の生成 */
    u[i][j] = a[i][j];
    for(k = 0; k < i; k++){
              u[i][j] -= u[k][j] * l[i][k];
    }
  }
  for(j = i+1; j < M; j++){          /* L行列の生成 */
    l[j][i] = a[j][i];
    for(k = 0; k < i; k++){
              l[j][i] -= u[k][i] * l[j][k];
    }
    l[j][i] /= u[i][i];
  }
}
```

```
for(i = 0; i < M; i++){              /* c行列の生成 */
  c[i] = b[i];
  for(j = 0; j < i; j++){
    c[i] -= l[i][j] * c[j];
  }
}

for(i = M-1; i >= 0; i--){           /* x行列の生成 */
  x[i] = c[i];
  for(j = M-1; j > i; j--){
    x[i] -= u[i][j] * x[j];
  }
  x[i] /= u[i][i];
}
printf("入力行列¥n");                /* 入力行列の出力 */
for(i = 0; i < M; i++){
  for(j = 0; j < M; j++){
    printf("%10.5lf ",a[i][j]);
  }
  printf("%10.5lf¥n",b[i]);
}
```

```
  printf("¥nL行列¥n");              /* L行列の出力 */
  for(i = 0; i < M; i++){
    for(j = 0; j < M; j++){
      printf("%10.5lf ",l[i][j]);
    }
    printf("¥n");
  }
  printf("¥nU行列¥n");              /* U行列の出力 */
  for(i = 0; i < M; i++){
    for(j = 0; j < M; j++){
      printf("%10.5lf ",u[i][j]);
    }
    printf("¥n");
  }
  printf("¥n");
  for(i = 0; i < M; i++){           /* 解の出力 */
    printf("x%d = %10.5lf¥n",i,x[i]);
  }
}
```

実行結果

入力行列として次の行列を与える

```
3  2  6  1  17
2  4  1  6  23
5  4  1  3  23
3  2  5  6  26
```

```
入力行列
   3.00000   2.00000   6.00000   1.00000  17.00000
   2.00000   4.00000   1.00000   6.00000  23.00000
   5.00000   4.00000   1.00000   3.00000  23.00000
   3.00000   2.00000   5.00000   6.00000  26.00000

L行列
   1.00000   0.00000   0.00000   0.00000
   0.66667   1.00000   0.00000   0.00000
   1.66667   0.25000   1.00000   0.00000
   1.00000   0.00000   0.12121   1.00000

U行列
   3.00000   2.00000   6.00000   1.00000
   0.00000   2.66667  -3.00000   5.33333
   0.00000   0.00000  -8.25000   0.00000
   0.00000   0.00000   0.00000   5.00000

x0 =   2.00000
x1 =   1.50000
x2 =   1.00000
x3 =   2.00000
```

ピボッティング

- もし$A_{i,i}$や$u_{k,k}$等の対角上の要素が0であれば，前式より，0徐算となり解が求まらない
- これを避けるために，0である要素を含む行と0でない要素を含む行を入れ替える（ピボッティング）
  - しかし、0でなくても，0に近い小さな値で割ったときは丸め誤差が大きくなる．この誤差を小さくするために，0でなくても，最も絶対値の大きな値を含む行と最も絶対値の小さな値を含む行 を入れ替えるのがよい

課題

- ピボッティングを行なうプログラムを作りなさい
- 実際に，対角要素の一部が0である行列を読み込ませて，動作を確認しなさい