数値計算法・数値解析

連立一次方程式：LU 分解

宮崎大学 工学部

第 7 回

連立一次方程式：LU 分解（利点）

連立方程式 $Ax = b$ で b を何種類も変えた時の結果が欲しい

- LU 分解して解く $\sim n^2$
  LU 分解を求めるのに $\sim n^3/3$

利点：少し早く解ける

比較：

- 毎回ガウス-ジョルダン法で解く $\sim n^3$
- 逆行列 A^{-1} をかける $\sim n^2$
  逆行列を求めるのに $\sim n^3$

LU 分解（利点）

LU 分解から 逆行列 A^{-1} を求める～ $2n^3/3$
※直接，逆行列を求めるのと同じ計算量

行列 A の行列式 $|A|$ を求める
※ LU 分解した行列の対角要素の積で計算可能

LU 分解はよく使われる！

LU分解

【定義】正方行列 A が与えられた時，

$$A = LU$$

となる下三角行列 L および上三角行列 U に分解する．

$$L = \begin{pmatrix} l_{00} & 0 & 0 & \dots & \\ l_{10} & l_{11} & 0 & \dots & \\ l_{20} & l_{21} & l_{22} & \dots & \\ \dots & & & & \\ l_{(n-1)0} & l_{(n-1)1} & \dots & l_{(n-1)(n-1)} & \end{pmatrix}, U = \begin{pmatrix} 1 & u_{01} & \dots & u_{0(n-1)} \\ 0 & 1 & u_{12} & \dots \\ 0 & 0 & 1 & \dots \\ \dots & & & \\ 0 & \dots & & 1 \end{pmatrix}$$

※ L の対角要素を 1 に，U の対角要素を u_{kk} にする場合もある

LU分解（原理）

1. ガウス-ジョルダン法と同様に，
   方程式のα倍と加算で上三角行列を作る
2. その時の α を記憶していくことで
   下三角行列が得られる

※ガウス-ジョルダン法で，各段階の列を記憶しておくと
下三角行列Lになる

例：以下の行列のLU分解

$$\begin{pmatrix} 2 & -4 & 6 \\ -1 & 3 & -4 \\ 1 & 1 & -2 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 3 & -2 \end{pmatrix} \begin{pmatrix} 1 & -2 & 3 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

例題：LU 分解

以下の行列を LU 分解せよ

$$\begin{pmatrix} 2 & 4 & -10 \\ 1 & 6 & 7 \\ 3 & 5 & -13 \end{pmatrix}$$

連立一次方程式：LU 分解による解法

$$
\begin{aligned}
Ax &= b \\
LUx &= b
\end{aligned}
$$

1. $Ly = b$ を y について解く（前進代入）
2. $Ux = y$ を x について解く（後退代入）

LU 分解：前進代入

$$Ly = \begin{pmatrix} l_{00} & 0 & 0 & \dots & \\ l_{10} & l_{11} & 0 & \dots & \\ l_{20} & l_{21} & l_{22} & \dots & \\ \dots & & & & \\ l_{(n-1)0} & l_{(n-1)1} & \dots & l_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \dots \\ y_{(n-1)} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \dots \\ b_{(n-1)} \end{pmatrix}$$

$$y_j = (b_j - \sum_{i=0}^{j-1} l_{ji} y_i)/l_{jj}$$

LU 分解：後退代入

$$
Ux = \begin{pmatrix} 1 & u_{01} & \dots & u_{0(n-1)} \\ \dots & & & \\ \dots & 1 & \dots & u_{(n-3)(n-1)} \\ \dots & 0 & 1 & u_{(n-2)(n-1)} \\ 0 & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_{(n-1)} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \dots \\ y_{(n-1)} \end{pmatrix}
$$

$$
x_j = (y_j - \sum_{i=j+1}^{n-1} u_{ji} x_i)
$$

例題：前進代入，後退代入

以下の連立一次方程式を前進代入，後退代入により解け

$$
\begin{pmatrix} 2 & -4 & 6 \\ -1 & 3 & -4 \\ 1 & 1 & -2 \end{pmatrix}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}
=
\begin{pmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 3 & -2 \end{pmatrix}
\begin{pmatrix} 1 & -2 & 3 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}
=
\begin{pmatrix} 5 \\ -3 \\ 2 \end{pmatrix}
$$

LU分解のプログラム(1/4)

```
#define N  3
int main(int argc, char *argv[])
{
  double      a[][N] = {{2, -4, 6},
                        {-1, 3, -4},
                        {1,  1, -2}};
  double      b[] = {5, -3, 2};

  lu_decomp(a, N);
  print_array(a, N);

  lu_solve(a, b, N);
  print_vec(b, N);

  return 0;
}
```

LU分解のプログラム(2/4)

```
void print_array(double a[][N], int size) {
  for(int j = 0; j < size; j++) {
    for(int i = 0; i < size; i++) {
      printf("%f␣", a[j][i]);
    }
    printf("\n");
  }
  printf("\n");
}

void print_vec(double v[], int size) {
  for(int i = 0; i < size; i++) {
    printf("%f␣", v[i]);
  }
  printf("\n");
}
```

LU 分解のプログラム (3/4)

```
void lu_decomp(double a[][N], int size)
{ // 前進消去
  for(int k = 0; k < size; k++) {
    double pivot = a[k][k];
    for(int i = k+1; i < size; i++) {
      a[k][i] /= pivot;
    }
    for(int j = k+1; j < size; j++) {

      double s = a[j][k];
      for(int i = k+1; i < size; i++) {
        a[j][i] = a[j][i] - s * a[k][i];
    } }
  }
  return;
}
```

LU分解のプログラム(4/4)

```
void lu_solve(double lu[][N], double *b, int size
) {
  double      y[size];
  // 前進代入
  for(int j = 0; j < size; j++) {
    y[j] = b[j];
    for(int i = 0; i < j; i++) {
      y[j] -= lu[j][i] * y[i];
    }
    y[j] /= lu[j][j];
  }

  // 後退代入
  // ！！プログラム課題で作成！！
  return;
}
```