数値解析レポート No.2

32 番 平田 蓮

1　LU 分解を用いて逆行列を求める

$C = LU$ と LU 分解をしたとき、$C^{-1} = U^{-1}L^{-1}P$ となる。上三角行列である U、下三角行列である L の逆行列を求めることができれば元の行列の逆行列を求めることができる。

1.0.1　上三角行列について

U は上三角行列であるので、U^{-1} も上三角行列である。この二つの行列を以下のように定める。

$$U = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix},\ U^{-1} = \begin{pmatrix} a'_{11} & a'_{12} & a'_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a'_{33} \end{pmatrix}$$

$UU^{-1} = E$ であるので、

$$UU^{-1} = \begin{pmatrix} a_{11}a'_{11} & a_{11}a'_{12} + a_{12}a'_{22} & a_{11}a'_{13} + a_{12}a'_{23} + a_{13}a'_{33} \\ 0 & a_{22}a'_{22} & a_{22}a'_{23} + a_{23}a'_{33} \\ 0 & 0 & a_{33}a'_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

となる。これを解くと、

$$U^{-1} = \begin{pmatrix} a'_{11} & -a_{12}a'_{11}a'_{22} & -a_{12}a'_{11}a'_{23} - a_{13}a'_{11}a'_{33} \\ 0 & a'_{22} & -a_{23}a'_{22}a'_{33} \\ 0 & 0 & a'_{33} \end{pmatrix}$$

$$\begin{cases} a'_{11} = \dfrac{1}{a_{11}} \\ a'_{22} = \dfrac{1}{a_{22}} \\ a'_{33} = \dfrac{1}{a_{33}} \end{cases}$$

を得れる。

1.0.2　下三角行列について

下三角行列と同様に以下のように定める。

$$L = \begin{pmatrix} b_{11} & 0 & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{pmatrix},\ L^{-1} = \begin{pmatrix} b'_{11} & 0 & 0 \\ b'_{21} & b'_{22} & 0 \\ b'_{31} & b'_{32} & b'_{33} \end{pmatrix}$$

$LL^{-1} = E$ であるので、

$$LL^{-1} = \begin{pmatrix} b_{11}b'_{11} & 0 & 0 \\ b_{21}b'_{11} + b_{22}b'_{21} & b_{22}b'_{22} & 0 \\ b_{31}b'_{11} + b_{32}b'_{21} + b_{33}b'_{31} & b_{32}b'_{22} + b_{33}b'_{32} & a_{33}a'_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

となる。これを解くと、

$$L^{-1} = \begin{pmatrix} b'_{11} & 0 & 0 \\ -b_{21}b'_{11}b'_{22} & b'_{22} & 0 \\ -b_{31}b'_{11}b'_{33} - b_{32}b'_{21}b'_{33} & -b_{32}b'_{22}b'_{33} & b'_{33} \end{pmatrix}$$

$$\begin{cases} b'_{11} = \dfrac{1}{b_{11}} \\ b'_{22} = \dfrac{1}{b_{22}} \\ b'_{33} = \dfrac{1}{b_{33}} \end{cases}$$

を得れる。

これらの結果を用いて L, U の逆行列を求める関数 `LUinv()` をリスト 1 に示す。

リスト 1　LUinv

```
1  void LUinv(double **L, double **Li, double **U, double **Ui, int row) {
2      int i, j, k;
3      for (i = 0; i < row; i++) {
4          Ui[i][i] = 1.0 / U[i][i];
5          Li[i][i] = 1.0 / L[i][i];
6      }
7
8      for (i = 1; i < row; i++) {
9          for (j = 0; j < row - i; j++) {
10             for (k = 0; k < i; k++) {
11                 Ui[j][j + i] -= U[j][j + k + 1] * Ui[j][j] * Ui[j + k + 1][j + i];
12                 Li[j + i][j] -= L[j + k + 1][j] * Li[j][j] * Li[j + i][j + k + 1];
13             }
14         }
15     }
16 }
```

このプログラムは、3 次元だけでなく任意の大きさの行列に対して計算が行えるようにしてある。

このプログラムを使用して C の逆行列を求めると、

$$C^{-1} = \begin{pmatrix} 0.030303 & -0.068182 & 0.083333 \\ -0.030303 & 0.318182 & 0.166667 \\ 0.181818 & 0.090909 & 0 \end{pmatrix}$$

となった。計算すると、正しいことがわかる。

2　はさみうち法

はさみうち法は、二分法を改良したものである。改良点として、二分法は次の端点を二端点の中点とするが、はさみうち法はに端点を結んだ直線が x 軸と交わる点とする。

はさみうち法を行う関数 false_pos() をリスト 2 に示す。

リスト 2　false_pos

```
1  void false_pos(
2      double a1,
3      double a2,
4      double (*f)(double x)
5  ){
6      int i;
7      double next;
8
9      puts("はさみうち法");
10
11     for (i = 1; EPOCHS >= i; ++i) {
12         next = (a1 * f(a2) - a2 * f(a1)) / (f(a2) - f(a1));
13
14         printf("Epoch:␣%2d,␣x=%lf\n", i, next);
15
16         if (0 > f(a1) * f(next)) {
17             a2 = next;
18         } else {
19             a1 = next;
20         }
21     }
22 }
23
```

このプログラムを使用して課題 3 の二分法、ニュートン法と比較してみる。

二分法

```
Epoch:  1, x=2.000000
Epoch:  2, x=1.500000
Epoch:  3, x=1.750000
Epoch:  4, x=1.875000
Epoch:  5, x=1.812500
Epoch:  6, x=1.781250
Epoch:  7, x=1.765625
Epoch:  8, x=1.757812
Epoch:  9, x=1.753906
Epoch: 10, x=1.751953
```

はさみうち法

```
Epoch:  1, x=1.181818
Epoch:  2, x=1.330162
Epoch:  3, x=1.447304
Epoch:  4, x=1.536569
Epoch:  5, x=1.602391
Epoch:  6, x=1.649613
Epoch:  7, x=1.682775
Epoch:  8, x=1.705697
Epoch:  9, x=1.721363
Epoch: 10, x=1.731984
```

ニュートン法

```
Epoch:  1, x=1.800000
Epoch:  2, x=1.755637
Epoch:  3, x=1.753667
Epoch:  4, x=1.753664
Epoch:  5, x=1.753664
Epoch:  6, x=1.753664
Epoch:  7, x=1.753664
Epoch:  8, x=1.753664
Epoch:  9, x=1.753664
Epoch: 10, x=1.753664
```

真の値は $x \approx 1.753664$ である。はさみうち法の方が収束が遅い。

3　ヤコビ法、ガウス・サイデル法が収束しない条件

二つの方法の収束条件として、係数行列の全ての行列について対角成分の絶対値が対角成分以外の絶対値の和より大きいというものがある。これを満たしていない方程式、

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 12 \\ 18 \end{pmatrix}$$

を考える。これの解は、

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

である。まず初期値を

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

としてヤコビ法によって解く。解の遷移は、

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \to \begin{pmatrix} 4 \\ 4 \\ 4 \end{pmatrix} \to \begin{pmatrix} -2 \\ -2 \\ -2 \end{pmatrix} \to \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix} \to \begin{pmatrix} -14 \\ -14 \\ -14 \end{pmatrix}$$

となり、絶対値が発散してしまい、収束しない。同様に、ガウス・サイデル法を用いても収束しない。

参考文献

[1] はさみうち法 (非線形方程式の数値解法), Qiita, @omu58n,
”https://qiita.com/omu58n/items/98886614ff92c00e5f05”

[2] 反復法, 東京大学, ”http://nkl.cc.u-tokyo.ac.jp/13n/SolverIterative.pdf”