数値解析レポートNo.2

Ec4-39 湯嶋 皓騎

実施環境

- CPU: Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
- OS: Arch Linux (WSL2 5.15.123.1-microsoft-standard-WSL2)
- コンパイラ: GNU Compiler Collection (GCC) 13.2.1 20230801
- Cライブラリ: GNU C Library (GNU libc) stable release version 2.38.

1.1. $A\mathbf{x} = \mathbf{b}$ の解 $\mathbf{x}$ を求める

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & -1 & 5 \\ 1 & 2 & -4 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 2 \\ -2 \\ 4 \end{pmatrix}$$

の解を求める.

1.1.1. 解析解の導出

連立方程式の解が存在するならば，$\mathrm{rank}A = \mathrm{rank}(A, \mathbf{b})$ である.
また，一意の解が存在するならば，$\mathrm{rank}A = \mathrm{rank}(A, \mathbf{b}) = n$ (ただし n は行列 A の列数) である.

$$A \Rightarrow \begin{pmatrix} 1 & -1 & 5 \\ 1 & 2 & -4 \\ 2 & 1 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & -1 & 5 \\ 0 & 3 & -9 \\ 0 & 3 & -9 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & -1 & 5 \\ 0 & 1 & -3 \\ 0 & 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -3 \\ 0 & 0 & 0 \end{pmatrix}$$

$$(A, \mathbf{b}) \Rightarrow \begin{pmatrix} 1 & -1 & 5 & -2 \\ 1 & 2 & -4 & 4 \\ 2 & 1 & 1 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & -1 & 5 & -2 \\ 0 & 3 & -9 & 6 \\ 0 & 3 & -9 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & -3 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

今回 $n = 3$ だから，$\mathrm{rank}A = \mathrm{rank}(A, \mathbf{b}) = 2, 2 \neq 3$である.
つまり，この方程式は$3 - 2 = 1$ 個の任意変数を有する非自明解を持つ.

ここで，任意変数 $t \in \mathbb{R}$ を用いて

$$A = \begin{pmatrix} 1 & 0 & 2t \\ 0 & 1 & -3t \\ 0 & 0 & 0 \end{pmatrix}$$

とおくと，この方程式の解 $\mathbf{x}$ は

$$\mathbf{x} = \begin{pmatrix} -2t \\ 3t + 2 \\ t \end{pmatrix}$$

である.

1.1.2. 通常通りのプログラムでは求められない理由

課題4から課題6で作成したプログラムの動作を考える.

課題4は前進消去・後退代入を用いて解を求めている．式 (2) からわかるように，後退代入で
係数行列の3行3列成分で割る動作を行うことになり，これはゼロ除算だから未定義動作を引き起こし，
正しい値にならない．

課題5はLU分解を用いて解を求めている．これも同様に後退代入を行うときにゼロ除算を行うため，未定義動作
を引き起こし，正しく値を求めることができない．

課題6は反復計算を用いて解を求めている．この方程式は一つの解に収束しないので，初期値とループ回数によって出力が変化する．
今回の場合ヤコビ法では解は発散し，ガウス・サイデル法では収束した．

初期値を 0 に設定したときは $\mathbf{x} = \begin{pmatrix} -4 \\ 8 \\ 2 \end{pmatrix}$ に，1 に設定したときは $\mathbf{x} = \begin{pmatrix} -10 \\ 17 \\ 5 \end{pmatrix}$ に収束した．これは正しい値であるが，$\mathbf{x}$ が一意の解を持つように見えてしまうので不適切である．

以上のことから，課題で作成したプログラムでは非自明な解を持つ方程式を解くことができないことがわかる．

1.1.3. 対策方法

課題4のプログラムについて対策方法を考える．
前進消去が完了した後に，成分が全て0の行が存在しないか確認する．もし存在したら一番下へ移動させる．
そして，零成分の行の解を1.0とすると，$t = 1.0$としたときの解が求まる．

コードを示す．

```
#define EPS 1e-10 // 十分に小さい数

int i, j;
double **matrix;
double *vector, *result;

// 前進消去を行う

for (i=0; i<row-1; i++) {
  for (j=0; i<col; j++) {
    if (matrix[i][j] > EPS) continue;

    // 行の成分が全て0
    swap_matrix(matrix, i, row-1);
    swap_vector(vector, i, row-1);
  }
}

for (i = row - 1; i >= 0; i--) {
  double row_value = vector[i];

  for (j = col - 1; j > i; j--) {
    row_value -= matrix[i][j] * result[j];
  }
```

```
    if (matrix[i][i] < EPS)
      result[i] = 1.0;
    else
      result[i] = row_value / matrix[i][i];
  }
```

result が $t = 1.0$ としたときの解になる.

1.2. 逆行列を求める

行列 $C = \begin{pmatrix} 2 & -1 & 5 \\ -4 & 2 & 1 \\ 8 & 2 & -1 \end{pmatrix}$ の逆行列を求める.

行列 C に対して LU 分解を行って得られた行列 L, U, P を用いて C^{-1} を求める.
$C^{-1} = U^{-1}L^{-1}P$ であり, $LL^{-1} = I, UU^{-1} = I$ (ただし I は単位行列)だから,
L^{-1} と U^{-1} を求めればよい.

ここで,

$$
\begin{aligned}
U &= (u_{ij}), \qquad & U^{-1} &= (u'_{ij}) \\
L &= (l_{ij}), \qquad & L^{-1} &= (l'_{ij})
\end{aligned}
$$

とおく.

1.2.1. U 行列の逆行列

$UU^{-1} = I$ を解く.

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

だから,

$$
\begin{pmatrix} u_{11}u'_{11} & u_{11}u'_{12} + u_{12}u'_{22} & u_{11}u'_{13} + u_{12}u'_{23} + u_{13}u'_{33} \\ 0 & u_{22}u'_{22} & u_{22}u'_{23} + u_{23}u'_{33} \\ 0 & 0 & u_{33}u'_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
$$

である. 対角成分に注目すると

$$
u'_{ii} = \frac{1}{u_{ii}}
$$

だとわかる. これを踏まえて対角成分以外の値を考えると, u'_{12} は,

$$
\begin{aligned}
u_{11}u'_{12} + u_{12}u'_{22} &= 0 \\
u'_{12} &= -\frac{u_{12}}{u_{11}u_{22}} = -u_{12}u'_{11}u'_{22}
\end{aligned}
$$

同様に，

$$
\begin{aligned}
u'_{23} &= -u_{23}u'_{22}u'_{33} \\
u'_{13} &= -u'_{11}(u_{12}u'_{23} + u_{13}u'_{33}) = u'_{11}u'_{33}(u_{12}u_{23}u'_{22} - u_{13})
\end{aligned}
$$

である．したがって，逆行列 U^{-1} は

$$
U^{-1} = \begin{pmatrix}
u'_{11} & -u_{12}u'_{11}u'_{22} & u'_{11}u'_{33}(u_{12}u_{23}u'_{22} - u_{13}) \\
0 & u'_{22} & -u_{23}u'_{22}u'_{33} \\
0 & 0 & u'_{33}
\end{pmatrix}
$$

ただし $u'_{ii} = u_{ii}^{-1}$ である．

1.2.2. L 行列の逆行列

同様に $LL^{-1} = I$ を解く．

$$
LL^{-1} = \begin{pmatrix}
l_{11}l'_{11} & 0 & 0 \\
l_{21}l'_{11} + l_{22}l'_{21} & l_{22}l'_{22} & 0 \\
l_{31}l'_{11} + l_{32}l'_{21} + l_{33}l'_{31} & l_{32}l'_{22} + l_{33}l'_{32} & l_{33}l'_{33}
\end{pmatrix} = I
$$

1.2.1. と同じように成分を求めると，

$$
L^{-1} = \begin{pmatrix}
l'_{11} & 0 & 0 \\
-l_{21}l'_{11}l'_{22} & l'_{22} & 0 \\
l'_{11}l'_{33}(l_{21}l_{32}l'_{22} - l_{31}) & -l_{32}l'_{22}l'_{33} & l'_{33}
\end{pmatrix}
$$

ただし $l'_{ii} = l_{ii}^{-1}$ である．

1.2.3. C の逆行列

$C^{-1} = U^{-1}L^{-1}P$ を計算する．課題5のプログラムを用いて LU 分解を行った結果，

$$
L = \begin{pmatrix}
1 & 0 & 0 \\
-0.5 & 1 & 0 \\
0.25 & -0.5 & 1
\end{pmatrix}
\qquad
U = \begin{pmatrix}
8 & 2 & -1 \\
0 & 3 & 0.5 \\
0 & 0 & 5.5
\end{pmatrix}
\qquad
P = \begin{pmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{pmatrix}
$$

であった．ここで U^{-1}, L^{-1} は

$$
U^{-1} = \begin{pmatrix}
0.1250 & -0.0833 & 0.0303 \\
0 & 0.3333 & -0.0303 \\
0 & 0 & 0.1818
\end{pmatrix},
\qquad
L^{-1} = \begin{pmatrix}
1 & 0 & 0 \\
0.5 & 1 & 0 \\
0 & 0.5 & 1
\end{pmatrix}
$$

と計算できる．これより，

$$C^{-1} = \begin{pmatrix} 0.030303 & -0.0681818 & 0.0833333 \\ -0.030303 & 0.318182 & 0.166667 \\ 0.181818 & 0.0909091 & 0 \end{pmatrix}$$

と求まった．

2. はさみうち法

2.1. はさみうち法とは

はさみうち法は関数の根を調べるアルゴリズムのひとつで，二分法と似たものである．

二分法では2点の中点を考えていたが，はさみうち法は2点を通る直線の値が0になる点を考えるものである．
二分法は1ステップごとに幅が半分になっていく一方，はさみうち法は関数の接点に近い直線を引くことができれば
少ないステップ数で根を求めることができる．

2.2. はさみうち法の評価

課題3を用いて評価を行う．課題3は関数 $f(x)$ の範囲 $(1:3)$ にある根を求める問題である．関数 $f(x)$の定義は以下の通りである．

$$f(x) = x^4 - 3x^2 + 5x - 9$$

はさみうち法で根を求める関数を示す．ただし，`FUNC_D2D` は `double` 型の変数を1つ引数に持ち，`double` 型の値を返す関数ポインタである． `SQUEEZE_METHOD` は列挙型のメンバである．

```
void squeeze_method(FUNC_D2D f, double x_min, double x_max) {
    int loop_count;

    // The loop starts at 1 and runs LOOP_COUNT_MAX times.
    for (loop_count = 1; (loop_count <= LOOP_COUNT_MAX); loop_count++) {
        double poi =
            (x_min * f(x_max) - x_max * f(x_min)) / (f(x_max) - f(x_min));
        print_current_result(SQUEEZE_METHOD, loop_count, poi);

        if (f(poi) > 0)
            x_max = poi;
        else
            x_min = poi;
    }
}
```

```
remain loop_count: 9, bisection method: 2.000000000, squeeze method: 1.181818182
remain loop_count: 8, bisection method: 1.500000000, squeeze method: 1.330162049
remain loop_count: 7, bisection method: 1.750000000, squeeze method: 1.447303924
remain loop_count: 6, bisection method: 1.875000000, squeeze method: 1.536568968
remain loop_count: 5, bisection method: 1.812500000, squeeze method: 1.602390729
remain loop_count: 4, bisection method: 1.781250000, squeeze method: 1.649612800
```

```
remain loop_count: 3, bisection method: 1.765625000, squeeze method: 1.682775076
remain loop_count: 2, bisection method: 1.757812500, squeeze method: 1.705697489
remain loop_count: 1, bisection method: 1.753906250, squeeze method: 1.721362780
remain loop_count: 0, bisection method: 1.751953125, squeeze method: 1.731983564
```

今回のケースでははさみうち法の方が二分法よりも収束が遅かった.

3. ヤコビ法やガウス・サイデル法が収束しない条件

線形方程式 $A\mathbf{x} = \mathbf{b}$ を反復的解法で解く場合を考える.
ここで，$A = D + L + U$ を満たすような対角行列 D と下三角行列 L，上三角行列 Uを用いると， $k+1$ ステップ目でのヤコビ法の漸化式は以下のように書き下せる．ただし $\mathbf{x}^{(n)}$ は n ステップ目の $\mathbf{x}$ の値を意味する.

$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - (L+U)\mathbf{x}^{(k)})$$

ここで $H = -D^{-1}(L+U), \mathbf{c} = D^{-1}\mathbf{b}$ とおくと,

$$\mathbf{x}^{(k+1)} = H\mathbf{x}^{(k)} + \mathbf{c}$$

と書き下せる．解が $k = N$ で収束するならば $\mathbf{x}^{(N+1)} = \mathbf{x}^{(N)}$ とみなせるから,

$$\begin{aligned}\mathbf{x}^{(k+1)} - H\mathbf{x}^{(k)} &= \mathbf{x}^{(N)} - H\mathbf{x}^{(N)} \\ \mathbf{x}^{(k)} - \mathbf{x}^{(k+1)} &= H(\mathbf{x}^{(k)} - \mathbf{x}^{(N)})\end{aligned}$$

である．この式は $k+1$ ステップでの誤差は k ステップの誤差を H 倍したものであることを意味している．つまり，$||H|| < 1$ ならば $\mathbf{x}$ は収束する．これは固有値の絶対値の最大値が1未満であることを意味する.

しかしながら，一般の行列の最大固有値を解析的に求めることは困難である．そこで，A が対角優位行列ならば固有値の絶対値は1よりも小さいことを利用すると，ヤコビ法は A が対角優位行列ならば収束することがわかる．これを数式で表わすと，行列の大きさを n としたとき，全ての $i = 1, 2, 3, \cdots n$ に対して

$$a_{ii} > \sum_{j=1,\, j\neq i}^{n} a_{ij}$$

を満たすならば収束すると言える.

ガウス・サイデル法の漸化式は

$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - L\mathbf{x}^{(k+1)} - U\mathbf{x}^{(k)})$$

と書き下され，これはヤコビ法の収束条件，または A が対称かつ正定値行列ならば収束する.
2つ目の条件を数式で書き表わすと，任意のベクトル $\mathbf{v}$ に対して

$$\mathbf{v}^T A\mathbf{v} > 0$$

が成立するならば収束する.

どちらも収束しない例として,

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

がある．この行列は $1 < 2 + 3$ だから対角優位ではない．また対称行列でもないので収束しないことが予想される．
実際にプログラムを動かすと発散することが確認できる．

ただし，ここで挙げた収束条件は十分条件であり，1.1.2. のようにこれを満たさなくとも収束する場合が存在する．

3. 工夫した点

はさみうち法を試すプログラムを作成する上で，関数ポインタや列挙型を用いてあらかじめインターフェースを整備しておくことで処理の本質的な部分のみを分かりやすく記述できた．

4. 要望

特にありません．