数値計算 講義資料
第1回 連立一次方程式

次のような連立一次方程式を数値計算で解く。

$$\left\{ \begin{array}{c} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{array} \right\} \tag{1}$$

この方程式は、n 行 n 列の正方行列 $\boldsymbol{A}$ とベクトル $\boldsymbol{x}, \boldsymbol{b}$ を用いて

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2}$$

と書くことができる。

1 Gauss-Jordan 法 掃き出し法

例えば、 以下の式

$$\left\{ \begin{array}{rcr} 2x - 2y + 3z & = & 1 \\ x + y - 6z & = & -1 \\ 3x - 2y + 4z & = & 4 \end{array} \right\} \tag{3}$$

から各係数と定数を抜き出し、次のように並べる

$$\left[\begin{array}{rrrr} 2 & -2 & 3 & 1 \\ 1 & 1 & -6 & -1 \\ 3 & -2 & 4 & 4 \end{array} \right] \tag{4}$$

左側に単位行列を作るように演算をおこなっていくと、右側の列に解が得られる。

1. 第 1 行目の両辺を 2 で割り第 1 行の先頭を 1 にする。

$$\left[\begin{array}{rrrr} 1 & -1 & 1.5 & 0.5 \\ 1 & 1 & -6 & -1 \\ 3 & -2 & 4 & 4 \end{array} \right] \tag{5}$$

2. 第 2 行- 第 1 行, 第 3 行-第 1 行 x3 の計算により、他の行の 1 列目をゼロにする。

$$\left[\begin{array}{rrrr} 1 & -1 & 1.5 & 0.5 \\ 0 & 2 & -7.5 & -1.5 \\ 0 & 1 & -0.5 & 2.5 \end{array} \right] \tag{6}$$

3.　第 2 行目の両辺を 2 で割り、第 2 行目 2 列の係数を 1 にする。

$$\begin{bmatrix} 1 & -1 & 1.5 & 0.5 \\ 0 & 1 & -3.75 & -0.75 \\ 0 & 1 & -0.5 & 2.5 \end{bmatrix} \tag{7}$$

4.　第 1 行- 第 2 行 x(-1), 第 3 行-第 1 行 の計算により、他の行の 2 列目をゼロにする。

$$\begin{bmatrix} 1 & 0 & -2.25 & 0.25 \\ 0 & 1 & -3.75 & -0.75 \\ 0 & 0 & 3.25 & 3.25 \end{bmatrix} \tag{8}$$

5.　第 3 行目の両辺を 3.25 で割り、第 3 行目 3 列の係数を 1 にする。

$$\begin{bmatrix} 1 & 0 & -2.25 & 0.25 \\ 0 & 1 & -3.75 & -0.75 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{9}$$

6.　第 1 行- 第 3 行 x(-2.25), 第 2 行-第 3 行 x(-3.75) の計算により、他の行の 3 列目をゼロにする。

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{10}$$

よって、解は、$x = 2, y = 3, z = 1$ である。それぞれのプログラムが完成したら、式 (3) の解を計算することにより、プログラムを確認してみよう。

この解法を一般化すると次のようになる。

1.　式 (1) の第 1 式の両辺を a_{11} で割る。

$$\left\{ \begin{array}{ccccc} x_1 & +(a_{12}/a_{11})x_2 & +\cdots+ & (a_{1n}/a_{11})x_n & = & b_1/a_{11} \\ a_{21}x_1 & +a_{22}x_2 & +\cdots+ & a_{2n}x_n & = & b_2 \\ \cdots & & & & & \\ a_{n1}x_1 & +a_{n2}x_2 & +\cdots+ & a_{nn}x_n & = & b_n \end{array} \right\} \tag{11}$$

2. 得られた第 1 式の両辺に a_{i1} をかけて、第 i 式 $(i = 2, \cdots, n)$ の x_1 の係数をゼロにする。

$$\left\{ \begin{array}{cccccc} x_1 & +(a_{12}/a_{11})x_2 & +\cdots+ & (a_{1n}/a_{11})x_n & = & b_1/a_{11} \\ & (a_{22}-a_{21}(a_{12}/a_{11}))x_2 & +\cdots+ & (a_{2n}-a_{21}(a_{1n}/a_{11}))x_n & = & b_2-a_{21}(b_1/a_{11}) \\ \cdots & & & & & \\ & (a_{n2}-a_{n1}(a_{12}/a_{11}))x_2 & +\cdots+ & (a_{nn}-a_{n1}(a_{1n}/a_{11}))x_n & = & b_n-a_{n1}(b_1/a_{11}) \end{array} \right\} \tag{12}$$

$a_{1j}^{(1)} = a_{1j}/a_{11}$, $b_1^{(1)} = b_1/a_{11}$ とすると、

$$\left\{ \begin{array}{cccccc} x_1 & +a_{12}^{(1)}x_2 & +\cdots+ & a_{1n}^{(1)}x_n & = & b_1^{(1)} \\ & (a_{22}-a_{21}a_{12}^{(1)})x_2 & +\cdots+ & (a_{2n}-a_{21}a_{1n}^{(1)})x_n & = & b_2-a_{21}b_1^{(1)} \\ \cdots & & & & & \\ & (a_{n2}-a_{n1}a_{n2}^{(1)})x_2 & +\cdots+ & (a_{nn}-a_{n1}a_{nn}^{(1)})x_n & = & b_n-a_{n1}b_1^{(1)} \end{array} \right\} \tag{13}$$

ここで、$a_{ij}^{(1)} = a_{ij}^{(0)} - a_{i1}a_{1j}^{(1)}$, $b_i^{(1)} = b_i^{(0)} - a_{i1}b_1^{(1)}$ とすると、式 (13) は、

$$\left\{ \begin{array}{llllll} x_1 & +a_{12}^{(1)}x_2 & +\cdots+ & a_{1n}^{(1)}x_n & = & b_1^{(1)} \\ & a_{22}^{(1)}x_2 & +\cdots+ & a_{2n}^{(1)}x_n & = & b_2^{(1)} \\ \cdots & & & & & \\ & a_{n2}^{(1)}x_2 & +\cdots+ & a_{nn}^{(1)}x_n & = & b_n^{(1)} \end{array} \right\} \tag{14}$$

3. 第 2 式の両辺を $a_{22}^{(1)}$ で割り、先ほどと同様に、第 i 式 $(i = 1, 3, 4, \cdots, n)$ の x_2 の係数をゼロとする。

$$\left\{ \begin{array}{lllllll} x_1 & & +a_{13}^{(2)}x_3 & +\cdots+ & a_{1n}^{(2)}x_n & = & b_1^{(2)} \\ & x_2 & +a_{23}^{(2)}x_3 & +\cdots+ & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ \cdots & & & & & & \\ & & a_{n3}^{(2)}x_3 & +\cdots+ & a_{nn}^{(2)}x_n & = & b_n^{(2)} \end{array} \right\} \tag{15}$$

4. 同様の操作を第 3 式から第 n 式まで繰り返して、消去を行うことにより、 x_1 から x_n までの解を求めることができる。

$$\left\{ \begin{array}{lllll} x_1 & & & = & b_1^{(n)} \\ & x_2 & & = & b_2^{(n)} \\ \cdots & & & & \\ & & x_n & = & b_n^{(n)} \end{array} \right\} \tag{16}$$

5. 一般に、第 k 段階のおける要素 $a_{ij}^{(k)}$ および、 $b_i^{(k)}$ は次の漸化式で与えられることになる。

   (a) 第 k 行のすべての要素を a_{kk} で割る。

$$\left\{ \begin{array}{l} a_{kj}^{(k)} = a_{kj}^{(k-1)}/a_{kk}^{(k-1)} \\ b_k^{(k)} = b_k^{(k-1)}/a_{kk}^{(k-1)} \end{array} \right\} (j = k+1, \cdots, n) \tag{17}$$

   (b) 第 k 行以外 $(i = 1, \cdots, k-1, k+1, \cdots, n)$ に対して、第 k 行を用いて、 k 列を 0 に。すでに 0 になっている部分は計算しないために、 $k+1$ 列から $(j = k+1, \cdots, n)$。

$$\left\{ \begin{array}{lll} a_{ij}^{(k)} & = & a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot a_{kj}^{(k)} \\ b_i^{(k)} & = & b_i^{(k-1)} - a_{ik} \cdot b_k^{(k)} \end{array} \right\} \left(\begin{array}{c} i = 1, \cdots, k-1, k+1, \cdots, n \\ j = k+1, \cdots, n \end{array} \right) \tag{18}$$

6. C 言語のプログラムでは、 配列は 0 から始まるため、以下のように書き直す。また、計算が同様なので、$\boldsymbol{A}$ とベクトル $\boldsymbol{x}, \boldsymbol{b}$ をあわせて一つの行列とすることができる。

$$\begin{bmatrix} a[0][0] & a[0][1] & \cdots & a[0][n-1] & a[0][n] \\ a[1][0] & a[1][1] & \cdots & a[1][n-1] & a[1][n] \\ \vdots & \vdots & & \vdots & \vdots \\ a[n-1][0] & a[n-1][1] & \cdots & a[n-1][n-1] & a[n-1][n] \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix} \tag{19}$$

7. 対角要素 a_{kk} が 0 になったり、他の要素に比べ非常に小さい場合には、オーバーフローがおこる。そのような場合には、行と行を入れ替えて計算を行い、計算終了後に入れ替えた行をもとに戻さなければならない。この処理をピボッティングと呼ぶが、今回は扱わない。

Gauss-Jordan 法のプログラム

gaussjordan_program.c

```c
#include <stdio.h>
#include <math.h>

void printmatrix(double □[11], int);  /*行列の出力*/

void gaussjordan(double □[11], int); /*Gauss-Jordan法*/

int main(void){

  double a[10][11];  /* n<=10まで*/
  int i, j, n;

  n, aの数値を入力する

  gaussjordan(a, n); /*Gauss-Jordan法*/
  printmatrix(a, n);

  return (0);
}


void printmatrix(double a[10][11], int n){
  int i, j;

  for (i=0; i<n; i++){
    for (j=0; j<n+1; j++){
      printf("%f ", a[i][j]);
    }
    printf("\n");
  }
  printf("\n");
}

void gaussjordan(double a[10][11], int n){
  int i, j, k;
  double w1, w2;

  for (k=0; k<n; k++){  /*cの配列は0から*/
    printmatrix(a, n);
    w1=a[k][k];
    for(j=k;j<=n; j++){
      a[k][j]=a[k][j]/w1;  /*式17の計算  第k行のすべての要素をa[k][k]で割る*/
    }                      /*j<kは０になっているので、j=kから*/
    for (i=0; i<=n; i++){
        w2=a[i][k];
        if (i != k){
          for(j=k;j<=n; j++){
            a[i][j]=a[i][j]-w2*a[k][j]; /*式18の計算 第k行以外に対して、第k行を用いてk列を０に*/
          }                             /*すでに０になっている部分は計算しない*/
        }
    }
  }

}
```

```
-u:--  gaussjordan_program.c   All L49    (C/l Abbrev)----12:42PM 1.15--------------------------------
(No changes need to be saved)
```

注意点: C 言語の場合、配列の要素の値は関数の中で書き換えられてしまう。

2　逆行列

Gauss-Jordan 法を用いて、逆行列を次のように計算することができる。

1. n 行 $2n$ 列の配列の左側に係数行列、右側に単位行列を作る

$$\left\{\begin{array}{cccccccc} a_{11} & a_{12} & \cdots & a_{1n} & 1 & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & a_{2n} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & 0 & 0 & \cdots & 1 \end{array}\right\} \tag{20}$$

2. この配列に対して、Gauss-Jordan 法を適用する。ただし、演算は行列全体に対して行う。　つまり、j についての演算のみ $< 2n$ まで行うように変更すると、右側に逆行列の成分が求められる。

3　Gauss の消去法

Gauss-Jordan の掃き出し法を改良し、計算回数を減らしたものである。

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix} \tag{21}$$

1. 前進部分
   掃き出し方法を用いた操作を行い、対角要素から下側の三角部分をゼロにする。
   (a) 第 2 行以降の第 1 列を 0 にする。
   $$\alpha_{i1} = -a_{i1}/a_{11}$$
   として、
   $$a_{ij}^{(1)} = a_{ij} + \alpha_{i1}a_{1j}, b_i^{(1)} = b_i + \alpha_{i1}b_1$$
   (b) 同様に作業を $k-1$ 回行う。
   (c) 第 $k+1$ 行以降の第 k 列を 0 にする。
   $$\alpha_{ik} = -a_{ik}^{(k-1)}/a_{kk}^{(k-1)}$$
   として、
   $$a_{ij}^{(k)} = a_{ij}^{k-1} + \alpha_{ik}a_{kj}^{(k-1)}, b_i^{(k)} = b_i^{k-1} + \alpha_{ik}b_k^{(k-1)}$$
   (d) $n-1$ 回目まで行うと、以下のような三角行列が完成される。
   $$\begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} & b'_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{nn} & b'_n \end{bmatrix} \tag{22}$$

2. 後退部分
   (a) 三角行列ができると、最後の式から、次のように x_n が求まる。
   $$x_n = b'_n/a'_{nn}$$

(b) x_n がわかれば、最後から 2 番目の式を用いて、次のように x_{n-1} が求まる。

$$x_{n-1} = (b'_{n-1} - a'_{n-1,n}x_n)/a'_{n-1,n-1}$$

(c) 同様に、順番に x_1 まで求めることができる。一般的には、次の式で計算できる。

$$x_i = (b'_i - \sum_{j=i+1}^{n} a'_{ij}x_j)/a'_{ii} \tag{23}$$

- 関数の始まり
  変数の宣言
- /*前進部分*/
  k=0 から n-2 まで繰り返す
  - i=k+1 から n-1 まで繰り返す
    - alpha=-a[i][k]/a[k][k]
    - j=k から n まで繰り返す
      a[i][j]=a[i][j]+alpha*a[k][j]
- /*後退部分の計算*/
  i=n-1 から 0 まで繰り返す
  式 23 を計算、配列が 0 から始まることに注意。
- 関数の終わり

前進部分で、 a[i][k] は 0 になることがわかっているので、計算する必要はないが、三角行列になっていることを確認できるように、計算を行っている。

4 Gauss-Seidel 法

適当な近似解を初期値として与え、反復処理により解に近づけていく手法。Gauss-Jordan 法や Gauss の消去法は、密な行列には威力を発揮するが、疎な行列の場合には、適した方法ではない。

$$\left\{ \begin{array}{c} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{array} \right\} \tag{24}$$

の次のように変形する。

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n)/a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n)/a_{22}$$

つまり、i 番目の式については、

$$x_i = (b_i - \sum_{j \neq i} a_{ij}x_j)/a_{ii} \tag{25}$$

このように、繰り返し x_1 から、x_n を収束するまで繰り返す。

収束条件としては、

$$e = \sum_{i=1}^{n} |(\text{前回の } x_i) - (\text{新しい } x_i)|$$

とし、e がある値より小さくなれば収束とする。または、５０回越えても収束しなければ、計算を終了する。初期値や行列によっては収束しないこともある。

Gauss-Seidel 法の関数、

1. 関数の始まり
2. 変数の宣言
3. 収束するまたは、50 回まで繰り返す
   - e=0
   - i=0 から n-1 まで繰り返す
     - w=b[i];
     - j=0 から n-1 まで繰り返す (ただし i=j は除く)
       - w から a[i][j]*x[j] を引く
     - w を a[i][i] で割る。
     - e に |x[i]-w| を加える
     - x[i]=w;

   計算途中の結果を出力する。