レポート課題 2

奥 屋 直 己

2018 年 7 月 31 日

1 課題 1

作成したガウス消去法を行うプログラムをソースコード 1 に示す。

ソースコード 1 ガウス消去法

```
#include<stdio.h>

#define N 10
#define M 100
#define L 100

//ガウス消去法の関数
void s_gauss(int n ,double x[] ,double a[][N] ,double b[]){
  int i,j,k;
  double mik;

  //ガウスの前進消去
  for(k=0 ;k<n-1 ;k++){
    for(i=k+1 ;i<n ;i++){
      mik=a[i][k]/a[k][k];
      for(j=k+1 ;j<n ;j++){
        a[i][j]-=mik*a[k][j];
      }
      b[i]-=mik*b[k];
    }
  }

  //ガウスの後退代入
  for(k=n-1 ;k>=0 ;k--){
    x[k]=b[k];
    for(i=k+1 ;i<n ;i++){
      x[k]-=a[k][i]*x[i];
    }
    x[k]/=a[k][k];
  }
}

int main(int argc ,char *argv[]){
  int i,j,k;
  int x_n=0;
  int sum;
  double a_b[N][N];
```

```
double a[N][N];
double x[N];
double b[N];
char f_name[M];
char temp[L]={};
FILE *fr;

//引数から取ったファイル名をf_name に代入
//ファイルには行列A と b を複合した物を書き込んでおく
sprintf(f_name,"%s",argv[1]);

if((fr=fopen(f_name,"r")) == NULL)
  printf("ファイルが開けません\n");
else{
  fgets(temp,L,fr); //ファイルの 1行目だけ読み込み、数字の数を数える
  for(i=0 ;i<L ;i++){
    if(temp[i]>=’0’ && temp[i]<=’9’){
      x_n++;
    }
  }

  rewind(fr); //もう一度 1行目から読み込む

  i=0;

  //ファイルから値を読み込みそれぞれを配列a_b に代入
  while(!feof(fr)){
    for(j=0 ;j<x_n ;j++){
      if(j < x_n-1)
        fscanf(fr,"%lf",&a_b[i][j]);
      else
        fscanf(fr,"%lf\n",&a_b[i][j]);
    }
    i++;
  }
  fclose(fr);
}

//i の値は行数となるのでそれを変数 sum に代入
sum=i;

//ファイルから読み込んだ値を行列A と b に分け、それぞれ配列 a,b に代入
for(i=0 ;i<sum ;i++){
  for(j=0 ;j<x_n ;j++){
    if(j < x_n-1){
      a[i][j]=a_b[i][j];
    }
    else
      b[i]=a_b[i][j];
  }
}

printf("A=\n");
```

```
  for(i=0 ;i<sum ;i++){
    for(j=0 ;j<x_n-1 ;j++){
      printf(" %5.2f ",a[i][j]);
    }
    printf("\n");
  }

  printf("\nb=\n");
  for(i=0 ;i<sum ;i++){
    printf(" %5.2f\n",b[i]);
  }

  s_gauss(sum,x,a,b);

  printf("\nx=\n");
  for(i=0 ;i<sum ;i++){
    printf(" %5.2f\n",x[i]);
  }

  return 0;
}
```

今回、行列の入力を容易にするため、あらかじめ入力すべき行列を保存したファイルを用意し、そのファイル読み込む形式とした。

1.1 実行例

以下にガウス消去法の実行例をソースコード 2 に示す。

ソースコード 2 実行結果

```
lesser [java] $ cat 1-1.dat
1 1 1 2
2 3 4 4
2 1 1 3
lesser [java] $ ./gauss 1-1.dat
A=
  1.00 1.00 1.00
  2.00 3.00 4.00
  2.00 1.00 1.00

b=
  2.00
  4.00
  3.00

x=
  1.00
  2.00
 -1.00
lesser [java] $ ./gauss 1-2.dat
A=
  1.00 2.00 1.00 2.00
 -1.00 -1.00 0.00 -1.00
 -2.00 -1.00 2.00 0.00
```

```
25   2.00 6.00 1.00 4.00
26
27 b=
28   0.00
29  -1.00
30  -6.00
31   5.00
32
33 x=
34   1.00
35   2.00
36  -1.00
37  -2.00
38 lesser [java] $
```

1.2 解説

まず、main 関数ないで作成しておいたデータが格納されたファイルを読み込む。ファイル名は引数でしていする。ファイル内は行列 **A** と行列 **b** を複合した値を保存しておく。これはソースコード 2 の 1 行目で、cat 関数で表示している。ソースコード 1 の 47 行目で char 型の変数 f_name に引数で入力したファイル名を代入し、49 行目でファイルを開く。ファイルが開けた場合、まず、ファイルの 1 行目のみ読み込み、何列で構成されているかを数字の数を読み込むことにより判定し、変数 x_n に代入する。59 行目の rewind 関数を使いもう一度 1 行目から読み込むようにリセットし、64 行目からデータを読み込む。fscanf 関数でそれぞれ読み込むが、1 行目の終端を読み込む時のみ、改行 \n も読み込む。この読み込んだ値は行列 **A** と **b** が混ざっているので、これらを 2 次元配列 a_b に格納する。80 行目より、2 次元配列 a_b を 2 次元配列 a と配列 b に分ける。次にガウス消去法の関数を解説する。このプログラムではガウスの消去法は 8 行目からの関数で行っている。引数として、行数 n、答えを格納する配列 x、行列 **A** を格納する配列 a、右辺の行列 **b** を格納する行列 b をとる。まず 12 行目からガウスの前進消去を行う。1 回目のループで 1 行目以降の 1 列目を 0 にするように計算する。2 行目が 1 行目の何倍となるかを計算した値を変数 mik に代入する。これを使い mik 倍した 1 行目引く 2 行目をし、1 列目を 0 にする。これを右辺の行列 b にも同じことをする。3 行目も同様に行う。2 回目のループで 2 行目以降の 2 列目を 0 にするように計算する。これを繰り返すことで、以下のような形にする。

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

次に 23 行目からの交代代入について説明する。後退代入では n 行目から走査を始める。n 行目では $a_{nn}x_n = b_n$ となっているので、$x_n = \frac{b_n}{a_{nn}}$ が答えとなる。n-1 行の時、$a_{n-1n-1}x_{n-1} = b_{n-1} - a_n n - 1 x_n$ となる。これを関数内では x[n-1] に b[n-1] を代入し、それから a[n][n-1]*x[n] を引いた値を x[n-1] に代入。最後に x[n-1] を a[n-1][n-1] で割った値を x[n-1] に代入し、求める。これを繰り返すことで行列 **x** を求める。

2 LU 分解

LU 分解を行うプログラムをソースコード 3 に示す。

ソースコード 3　LU 分解

```
#include<stdio.h>

#define N 10
```

```
#define M 100
#define L 100

void intlz_lu(int n ,double a[][N] ,double l[][N] ,double u[][N]){
  int i,j;

  for(i=0 ;i<n ;i++){
    for(j=0 ;j<n ;j++){
      u[i][j]=a[i][j];
    }
  }

  for(i=0 ;i<n ;i++){
    for(j=0 ;j<n ;j++){
      if(i == j)
        l[i][j]=1.0;
      else
        l[i][j]=0.0;
    }
  }
}

void lu(int n ,double l[][N] ,double u[][N]){
  int i,j,k;
  double mik;

  for(k=0 ;k<n-1 ;k++){
    for(i=k+1 ;i<n ;i++){
      mik=u[i][k]/u[k][k];
      l[i][k]=mik;
      u[i][k]=0.0;
      for(j=k+1 ;j<n ;j++){
        u[i][j]-=mik*u[k][j];
      }
    }
  }
}

int main(int argc ,char *argv[]){
  int i,j;
  int sum=0;
  int x_n=0;
  double a[N][N];
  double l[N][N],u[N][N];
  char f_name[M];
  char temp[L]={};
  FILE *fr;

  sprintf(f_name,"%s",argv[1]);

  if((fr=fopen(f_name,"r")) == NULL)
    printf("ファイルが開けません\n");
  else{
```

```
    fgets(temp,L,fr);
    for(i=0 ;i<L ;i++){
      if(temp[i]>='0' && temp[i]<='9'){
        x_n++;
      }
    }

    rewind(fr);

    i=0;

    while(!feof(fr)){
      for(j=0 ;j<x_n ;j++){
        if(j < x_n-1)
          fscanf(fr,"%lf",&a[i][j]);
        else
          fscanf(fr,"%lf\n",&a[i][j]);
      }
      i++;
    }
    fclose(fr);
  }

  sum=i;
  printf("A=\n");
  for(i=0 ;i<sum ;i++){
    for(j=0 ;j<x_n ;j++){
      printf(" %5.2f ",a[i][j]);
    }
    printf("\n");
  }

  intlz_lu(sum,a,l,u);
  lu(sum,l,u);

  printf("\nL=\n");
  for(i=0 ;i<sum ;i++){
    for(j=0 ;j<x_n ;j++){
      printf(" %5.2f ",l[i][j]);
    }
    printf("\n");
  }

  printf("\nU=\n");
  for(i=0 ;i<sum ;i++){
    for(j=0 ;j<x_n ;j++){
      printf(" %5.2f ",u[i][j]);
    }
    printf("\n");
  }

  return 0;
}
```

このプログラムもガウス消去法と同様にファイルからデータを読み込んでいる。ただし、今回は行列 A の部分だけとなっている。

2.1 実行例

以下に実行例をソースコード 4 に示す。

ソースコード 4　LU 分解の実行結果

```
lesser [java] $ cat 2-1.dat
1 1 1
2 3 4
2 1 1
lesser [java] $ ./LU 2-1.dat
A=
  1.00 1.00 1.00
  2.00 3.00 4.00
  2.00 1.00 1.00

L=
  1.00 0.00 0.00
  2.00 1.00 0.00
  2.00 -1.00 1.00

U=
  1.00 1.00 1.00
  0.00 1.00 2.00
  0.00 0.00 1.00
lesser [java] $ cat 2-2.dat
1 2 1 2
-1 -1 0 -1
-2 -1 2 0
2 6 1 4
lesser [java] $ ./LU 2-2.dat
A=
  1.00 2.00 1.00 2.00
 -1.00 -1.00 0.00 -1.00
 -2.00 -1.00 2.00 0.00
  2.00 6.00 1.00 4.00

L=
  1.00 0.00 0.00 0.00
 -1.00 1.00 0.00 0.00
 -2.00 3.00 1.00 0.00
  2.00 2.00 -3.00 1.00

U=
  1.00 2.00 1.00 2.00
  0.00 1.00 1.00 1.00
  0.00 0.00 1.00 1.00
  0.00 0.00 0.00 1.00
lesser [java] $
```

2.2 解説

main 関数はガウス消去法とほぼ同じとなっている。この問題では配列 l と u をそれぞれ初期化する関数と実際に LU 分解を行う関数に分けた。 7 行目の intlz_lu 関数では配列 l と u をそれぞれ初期化している。引数として、配列の行数 n と行列 A、L、U をそれぞれ格納した配列 a、l、u となっている。配列 u は配列 a をそのまま代入する。配列 l は単位行列で初期化する。すなわち、配列の行番号と列番号が等しい時のみ 1.0 を代入し、それ以外は 0.0 を代入する。次に 26 行目で LU 分解を行う lu 関数を定義する。引数として、配列の行数 n と配列 l、u を取っている。LU 分解の計算方法は 3×3 の時だと

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{23} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{23} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

となればいいので、行列 U の 1 行目は

$$u_{11} = a_{11}$$

$$u_{12} = a_{12}$$

$$u_{13} = a_{13}$$

となる、次に L の 1 列目を計算する

$$l_{21} = \frac{a_{21}}{u_{11}}$$

$$l_{31} = \frac{a_{31}}{u_{11}}$$

となる。さらに U の 2 行目を計算すると

$$u_{22} = a_{22} - l_{21}u_{12}$$

$$u_{23} = a_{23} - l_{21}u_{13}$$

同様に 3 行目も行うことで計算できる。ソースコード 3 の 26 行目はこれと同様のことを行っている。配列 u はすでに配列 a で初期化しているため。u の 1 行目は計算されている。上記の計算では $\frac{a_{21}}{u_{11}}$ だが、配列 u は配列 a で初期化しているので、u[2][1]/u[1][1] で行う。この値を変数 mik に代入し、これを l[1][0] に代入し、同じ行と列の u[1][0] に 0 を代入する。このループ内のまま次の行の配列 u を求める。これを繰り返し LU 分解を行う。

3 ドゥーリトル法

作成したプログラムをソースコード 5 に示す。

ソースコード 5 ドゥーリトル法

```
//LU 分解から x を求める

#include<stdio.h>

#define N 10
#define M 100
#define L 100

//LU 分解をするの関数
void lu(int n ,double a[][N] ,double l[][N] ,double u[][N]){
  int i,j,k;
  double mik;

```

```
  //U 行列に A 行列の値を代入
  for(i=0 ;i<n ;i++){
    for(j=0 ;j<n ;j++){
      u[i][j]=a[i][j];
    }
  }

  //L 行列を単位行列で初期化
  for(i=0 ;i<n ;i++){
    for(j=0 ;j<n ;j++){
      if(i == j)
        l[i][j]=1.0;
      else
        l[i][j]=0.0;
    }
  }

  //LU 分解
  for(k=0 ;k<n-1 ;k++){
    for(i=k+1 ;i<n ;i++){
      mik=u[i][k]/u[k][k];
      l[i][k]=mik;
      u[i][k]=0.0;
      for(j=k+1 ;j<n ;j++){
        u[i][j]-=mik*u[k][j];
      }
    }
  }
}

//LU 分解後の L と行列 b から行列 y を求める
void f_gauss(int n ,double y[] ,double a[][N] ,double b[]){
  int i,j;

  //ガウスの前進代入
  for(i=0 ;i<n ;i++){
    y[i]=b[i];
    for(j=0 ;j<i ;j++){
      y[i]-=a[i][j]*y[j];
    }
    y[i]/=a[i][i];
  }
}

//LU 分解後の行列 U と行列 y から答えの行列 x をもとめる
void r_gauss(int n ,double x[] ,double a[][N] ,double b[]){
  int i,j;

  //ガウスの後退代入
  for(i=n-1 ;i>=0 ;i--){
    x[i]=b[i];
    for(j=i+1 ;j<n ;j++){
      x[i]-=a[i][j]*x[j];
```

```c
    }
    x[i]/=a[i][i];
  }
}

int main(int argc ,char *argv[]){
  int i,j;
  int sum=0;
  int x_n=0;
  double a_b[N][N];
  double a[N][N],b[N];
  double l[N][N],u[N][N];
  double x[N],y[N];
  char f_name[M];
  char temp[L]={};
  FILE *fr;

  sprintf(f_name,"%s",argv[1]);

  if((fr=fopen(f_name,"r")) == NULL)
    printf("ファイルが開けません\n");
  else{
    fgets(temp,L,fr); //一列目だけ読み込み、数字の数を数える
    for(i=0 ;i<L ;i++){
      if(temp[i]>='0' && temp[i]<='9'){
        x_n++; //行列A と b を並べた横の数字の数
      }
    }

    rewind(fr); //ファイルをもう一度最初から読み込む

    i=0;

    while(!feof(fr)){
      for(j=0 ;j<x_n ;j++){
        if(j < x_n-1) //一列目の最後の値を読み込むときだけ改行も読み込む
          fscanf(fr,"%lf",&a_b[i][j]);
        else
          fscanf(fr,"%lf\n",&a_b[i][j]);
      }
      i++;
    }
    fclose(fr);
  }

  sum=i; //列数を格納

  //読み込んだ行列A と b が複合された数列を a と b に分ける
  for(i=0 ;i<sum ;i++){
    for(j=0 ;j<x_n ;j++){
      if(j < x_n-1){
        a[i][j]=a_b[i][j];
      }
```

```
      else
        b[i]=a_b[i][j];
    }
  }

  printf("A=\n");
  for(i=0 ;i<sum ;i++){
    for(j=0 ;j<x_n-1 ;j++){
      printf(" %5.2f ",a[i][j]);
    }
    printf("\n");
  }

  printf("\nb=\n");
  for(i=0 ;i<sum ;i++){
    printf(" %5.2f \n",b[i]);
  }

  lu(sum,a,l,u);

  f_gauss(sum,y,l,b);

  r_gauss(sum,x,u,y);

  printf("\nx=\n");
  for(i=0 ;i<sum ;i++){
    printf(" %5.2f \n",x[i]);
  }

  return 0;
}
```

3.1 実行結果

以下に実行例をソースコード 6 に示す。

ソースコード 6　ドゥーリトル法の実行結果

```
lesser [java] $ ./LU-2 1-1.dat
A=
  1.00 1.00 1.00
  2.00 3.00 4.00
  2.00 1.00 1.00

b=
  2.00
  4.00
  3.00

x=
  1.00
  2.00
 -1.00
lesser [java] $ ./LU-2 1-2.dat
```

```
A=
  1.00 2.00 1.00 2.00
 -1.00 -1.00 0.00 -1.00
 -2.00 -1.00 2.00 0.00
  2.00 6.00 1.00 4.00

b=
  0.00
 -1.00
 -6.00
  5.00

x=
  1.00
  2.00
 -1.00
 -2.00
lesser [java] $
```

実行結果よりガウス消去法と値が一致していることがわかる。

3.2 解説

このプログラムの main 関数はガウス消去法とほぼ同じとなっている。また、10 行目の lu 関数も LU 分解で説明したものとなっている。このプログラムは LU 分解したものから答えとなる行列 x を求めるものとなっている。LU 分解を行なった結果をそれぞれ配列 l,u に格納している。これらとガウス消去法で行なった、後退代入を使い、答えを求める。LU 分解を行なったあと、以下のように式を書き換えれる。

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{LUx} = \mathbf{b}$$

$$\mathbf{Ux} = \mathbf{y}$$

とおくと、

$$\mathbf{Ly} = \mathbf{b}$$

となる。これより、$\mathbf{Ly} = \mathbf{b}$ を使い $\mathbf{y}$ を求め、$\mathbf{Ux} = \mathbf{y}$ を使い $\mathbf{x}$ を求める。ソースコード 3 の 45 行目より、行列 y を求める。行列 L の形はガウスの後退代入を行うときの形の逆なのでループする向きを逆にすれば良い。また、58 行目で x を求める時は、ガウスの後退代入がそのまま適応できるので、これから x を求めるとこで、解を求めた。

4 逆行列のプログラム

行列 A の逆行列は

$$A^{-1} = L^{-1}U^{-1}$$

で求めることができる。3×3 の場合、L^{-1} と U^{-1} は以下のように求めることができる。

$$\begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{23} & 1 \end{pmatrix} \begin{pmatrix} p_{11} & 0 & 0 \\ p_{21} & p_{22} & 0 \\ p_{31} & p_{23} & p_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ 0 & p_{22} & p_{23} \\ 0 & 0 & p_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

これらはそれぞれ

ソースコード 7　L^{-1}

```
for(i=0 ;i<n ;i++){
  for(j=0 ;j<i-1 ;i++){
    for(k=j ;k<i-1 ;k++){
      p[i][j]+=l[i][k]*p[k][j];
    }
    p[i][j]/=-l[i][i];
  }
 }
```

ソースコード 8　U^{-1}

```
for(i=n-1 ;i>=0 ;i--){
  p[i][i]=1/u[i][i];
  for(j=i+1 ;j<n ;j++){
    for(k=i+1 ;k<j k++){
      p[i][j]+=u[i][k]p[k][j];
    }
    p[i][j]/=-u[i][i];
  }
 }
```

で求めることができる。これより行列 A の逆数を LU を使い求めることができる。

5　行列式の計算

行列 A の行列式は以下のようにして求めることができる。

$$|A| = |L||U|$$

また、L と U の行列式はそれぞれ

$$|L| = l_{11} * l_{22} * \ldots * l_{nn}$$

$$|U| = u_{11} * u_{22} * \ldots * u_{nn}$$

あとはこの二つをかけ合わせると行列 A の行列式が求まる。