計算機プログラミングI 講義ノート #9

担当：井上 純一 (情報科学研究科棟 8-13), 赤間 清 (情報基盤センター)
URL：http://chaosweb.complex.eng.hokudai.ac.jp/~j_inoue/PROG2007/PROG2007.html

平成 19 年 6 月 22 日

目 次

※ 注意：今回は練習問題のプログラムと結果を紙に印刷したものを期日までに提出していただきます.

19 数値計算 II：連立 1 次方程式の解法 — Gauss-Jordan 法 —

ここでは配列を用いた数値計算の一例として連立方程式の数値的な解法を取り上げます. 具体的には

$$\begin{aligned} a_{11}x + a_{12}y + a_{13}z &= b_1 \\ a_{21}x + a_{22}y + a_{23}z &= b_2 \\ a_{31}x + a_{32}y + a_{33}z &= b_3 \end{aligned}$$

のような方程式を解くことを考えます. ここで, 上記の方程式は次のように行列を用いて書き直せることに注意しましょう.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{17}$$

ここで, 正方行列

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{18}$$

と 1×3 行列 (ベクトル)

$$\boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{19}$$

を合わせた次の**拡大係数行列** $(\boldsymbol{A} : \boldsymbol{b})$:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{pmatrix} \tag{20}$$

に対して基本変形を行い, 正方行列 $\boldsymbol{A}$ の部分を単位行列にすることができれば, そのときの $\boldsymbol{b}$ がその答えを与えるというのがこの解法 — **Gauss-Jordan 法 (掃き出し法)** — での基本的な指針となります.

例えば, 上の拡大係数行列 $(\boldsymbol{A} : \boldsymbol{b})$ に対し, 第 1 行を a_{11} で割ると (以下の操作で各行の成分を割る係数 a_{11} 等を**ピポット**と呼びます)

$$\begin{pmatrix} 1 & a_{12}/a_{11} & a_{13}/a_{11} & b_1/a_{11} \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{pmatrix} \tag{21}$$

が得られます. 次にこの行列 (21) の第 1 行を a_{21} 倍したものを第 2 行から引くことで

$$\begin{pmatrix} 1 & a_{12}/a_{11} & a_{13}/a_{11} & b_1/a_{11} \\ 0 & a_{22} - (a_{12}a_{21}/a_{11}) & a_{23} - (a_{13}a_{21}/a_{11}) & b_2 - (b_1a_{21}/a_{11}) \\ a_{31} & a_{32} & a_{33} & b_3 \end{pmatrix} \tag{22}$$

となります. 最後にこの行列 (22) の第 1 行を a_{31} 倍したものを第 3 行から引くことで

$$\begin{pmatrix} 1 & a_{12}/a_{11} & a_{13}/a_{11} & b_1/a_{11} \\ 0 & a_{22} - (a_{12}a_{21}/a_{11}) & a_{23} - (a_{13}a_{21}/a_{11}) & b_2 - (b_1a_{21}/a_{11}) \\ 0 & a_{32} - (a_{12}a_{31}/a_{11}) & a_{33} - (a_{13}a_{31}/a_{11}) & b_3 - (b_1a_{31}/a_{11}) \end{pmatrix} \tag{23}$$

が得られることになり, 無事に第 1 列に $1, 0, 0$ が現れることになるわけです.

従って, 正方行列 $\boldsymbol{A}$ を単位行列にするためには, 行列 (23) においてピポットを $\{a_{22} - (a_{12}a_{21}/a_{11})\}$ に選び, これで第 2 行を割り, ... という操作を繰り返していけばよいことになります.

この操作によって, 上記の拡大係数行列の右端に現れるベクトル $\boldsymbol{b}$ は $\boldsymbol{b} \to \boldsymbol{b}' \to \cdots$ のように変化していきますが, 正方行列 $\boldsymbol{A}$ が単位行列になった時点での $\boldsymbol{b}$ の値がこの連立方程式の解を与えることになるわけです.

練習問題 19.1

教科書に与えられた連立方程式:

$$\begin{aligned} 2x + y - z &= 5 \\ -3x + 3y + 2z &= 1 \\ x - 2y - 2z &= -1 \end{aligned}$$

に対し, 上記に説明した操作を繰り返して解を求めよ. (※ これはプログラミングをするのではなく, 紙に鉛筆で計算すること).

練習問題 19.2

次のプログラムをファイルに書き込み, コンパイル・実行することで動作を確かめよ. (※ 教科書 100 ページの図 6.7 に載っている PAD も合わせて参照すること).

```
#include<stdio.h>
#define IMAX 3
#define JMAX 4

double array[IMAX][JMAX]={
  { 2,1,-1,5 },
  { -3,3,2,1 },
  { 1,-2,-2,-1 },
};

void PrintMat(void){
  int i,j;
  for(i=0; i<IMAX; i++){
    for(j=0; j<JMAX; j++){
      printf("%5.2f ",array[i][j]);
    }
    putchar('\n');
  }
  putchar('\n');
}

void SweepOut(void){
  int i,j,axis;
  double pivot,aik;

 for(axis=0; axis<IMAX; axis++){
    pivot=array[axis][axis];

    for(j=axis;j<JMAX;j++){
      array[axis][j]/=pivot;
    }

    for(i=0;i<IMAX;i++){
      if(i!=axis){
aik=array[i][axis];
for(j=0; j<JMAX;j++){
  array[i][j]-=array[axis][j]*aik;
```

```
}
      }
    }
  }
}

main()
{
  PrintMat();
  SweepOut();
  PrintMat();
}
```

20　リダイレクション

Linux 上で動作する C 言語プログラミングでは, プログラミング中に現れた scanf 関数, printf 関数に対し, リダイレクションという操作を行うことで, キーボード入力をファイルからの入力へ, ディスプレイへの出力をファイルへの出力として与えることができます. 例えば, input.dat というファイルに書かれた数字の列をプログラム test.c に現れる scanf 関数の入力として与えたい場合には, test.c をコンパイルして得られる実行形式 a.out に対して

```
./a.out < input.dat
```

とすればよいことになります.

逆に test.c の中に現れる printf 関数の出力を output.dat に出力したい場合には

```
./a.out > output.dat
```

とします. この両者を組み合わせる, すなわち, input.dat というファイルに書かれた数字の列をプログラム test.c に現れる scanf 関数の入力として与え, かつ, 逆に test.c の中に現れる printf 関数の出力を output.dat に出力したい場合には

```
./a.out < input.dat > output.dat
```

とすることになります. 以上をふまえて次の練習問題をやっていただきます.

練習問題 20.1

1 から 9 までの数字を書き込んだ input.dat というファイルを予め用意しておき, これら数字の和を計算し,

```
sum=*****
```

という形式の出力を output.dat に出力させるようなプログラムを作成し, 実行結果を確かめよ.

21　ファイル操作

前節でリダイレクションによるファイルからのデータの読み込みとファイルへの書き込みを学びましたが, ここでは, より柔軟なファイルの操作法を学びます. また, データの関連性を視覚化するものとして, gnuplot と呼ばれるグラフ作成ツールの使い方についても学習します.

21.1 ファイル操作の基本

以下に ASCII ファイルの**読み込み**と**書き込み**の典型的なプログラムを載せておきます. 教科書の第 8 章を合わせて確認してください.

```
/* ファイルの読み込みと書き込みの典型例 */
#include<stdio.h>
#include<stdlib.h>

main()
{
  FILE *pt,*pt2;
  double data, sum=0;
  int n;
        /* データを ASCII ファイル ascii.txt から読み込む */
       if((pt=fopen("ascii.txt", "rt")) != NULL){
     /* データを一つ一つ読み込み，その総和を算出 */
  while(fscanf(pt,"%lf", &data) != EOF){
       sum += data;
          n++;
  }
fclose(pt);
/* ascii.txt のデータ数，平均値を算出し，mean.txt に書き込む */
if((pt2=fopen("mean.txt", "wt")) != NULL){
fprintf(pt2, "n = %d \n sum = %lf", n, sum);
           }
fclose(pt2);
}
```

21.2 gnuplot によるグラフの作成

数値データが数値計算により求まったら, 今度はそのデータをグラフとして見てみたいと思うのが人情というものです. ここでは gnuplot というフリーソフトを用いて, データをグラフ化 (可視化) する方法を簡単に見て行くことにします.

21.2.1 gnuplot の起動とデータのプロット

コマンドラインから

```
iecsv{your_account}2% gnuplot [リターン]
```

としてみましょう. すると gnuplot が立ち上がり

```
gnuplot >
```

という入力待ちの状態になります. そこで例えば

```
0.100000 1.000000
0.500000 1.000000
1.000000 0.999282
1.500000 0.986609
.................
.................
```

という内容のデータファイル test.dat に対して, 第 1 列を x 軸に, 第 2 列を y 軸として描きたいのであれば

gnuplot > plot “test.dat” [リターン]

とすれば, test.dat が描写されます. 「線」で描きたい場合には

gnuplot > plot “test.dat” with line [リターン]

とすればよく, gnuplot を終了したい場合には

gnuplot > quit [リターン]

とします. データが

```
0.100000 1.000000 -4.000000
0.500000 1.000000 -4.000000
1.000000 0.999282 -3.994238
1.500000 0.986609 -3.894168
.................
.................
```

のように 3 行に渡っていて, この第 1 行を x 軸, 第 3 行を y 軸に描きたいのであれば

gnuplot > plot “test.dat” using 1:3 with line [リターン]

とすれば OK です.

21.2.2　gnuplot で描いたグラフを印刷形式で保存する方法

gnuplot で描いた図をレポートや論文として印刷できる形式 (ポストスクリプト形式, PS ファイル) に変換するにはコマンドラインから

```
iecsv{your_account}3%  gnuplot [リターン]
```

として gnuplot に入り, そのあと

gnuplot > set term postscript [リターン]

gnuplot > set output “out.ps” [リターン]
gnuolot > plot “test.dat” [リターン]

とすれば, ここでは何も表示されないですが

gnuolot > quit [リターン]

として gnuplot を抜けて

```
iecsv{your_account}4% ls  [リターン]
```

で見てみると, 確に out.ps ができています. この out.ps をディスプレイ上で見るには ghostview を用います.

```
iecsv{your_account}5% ghostview out.ps [リターン]
```

また, プリントアウトしたければ

```
iecsv{your_account}6% lpr out.ps  [リターン]
```

とすれば OK です.

21.3　写像力学系とカオス

ここでは物理学, 数学, 生物学や経済学などの社会科学, 工学など非常に広範な分野で研究されているカオスについて, その性質や基礎的概念に簡単に触れる目的で写像力学系を用いた数値実験を行なうことにしてみましょう. そして gnuplot による「データの可視化」を実際に行なってみましょう.

21.3.1　写像力学系

カオスが他の運動と異なるものであるという意味で, カオスとは

比較的簡単な規則に支配された不規則な運動

とであると言われます. こう書いてしまうと, 何のことだかさっぱりわからないかもしれません. そこで, 実際にそのような運動を生じさせる規則 (力学系) をとりあげてこの言葉の意味を理解することから始めてみましょう. ここで, 力学系というと, 微分方程式としてのニュートンの運動方程式を連想されるかもしれませんが, ここでは簡単のため, 写像 (マップ) と呼ばれる次の繰り返し演算を扱うことにします.

$$x_{n+1} \quad = \quad L(x_n) \quad (n = 1, 2 \cdots) \tag{24}$$

これは初期値を x_1 とした数列 $x_1, x_2, \cdots, x_n, \cdots$ が規則 (24) に従って生成されるということを意味します. この n を「時間」と考えれば, この写像 (24) を一種の時間発展とみなすこともできます. もちろん, 自然界の現象の中で, このような単純な写像で記述されるものはほとんどないですが, その現象のある側面のみ

を注目して観察し, かつ運が良ければこの写像も, その現象の, その注目する側面に関しては良い近似を与える場合があります. 従って, 写像を勉強すること自体は決して無意味なことではないわけです.

さて, この写像を特徴つけるのは関数 $L(x)$ です. これは有限区間で定義されるものが望ましいでしょう. また, できることならば, 何らかの可変なパラメータを有し (例えば a としょう), そのパラメータ a を調節することにより, 単なる固定点への漸近挙動だけでなく, 周期運動, そしてカオスとそこから生まれる運動にバラエティを持たせられるものがよいわけです. そこで, まずは次のような関数を使った写像力学系を調べて行くことにします.

$$x_{n+1} = L(x_n; a) = ax_n(1 - x_n) \tag{25}$$

ここで, $a = 4$ の場合は**ロジスティック写像**と呼ばれる有名な写像です. さて, 実際に (25) がどのような数列を生成するかを確認するために, 次のような課題を行なってもらうことにしょう.

今週の提出課題

写像力学系 (25) より初期値を $x_1 = 0.1$ として数列 $x_1, x_2, \cdots, x_n$ (n は 100 程度でよい.) を求めるプログラムを作成し, 横軸を n 縦軸を x_n として, gnuplot でプロットし, その挙動を考察せよ. a の値は $a = 1.0, 3.5, 4.0$ の 3 つの場合について行なうこと.

(注)
得られた結果をグラフとして見る場合, 得られるデータをファイルに

```
1.000000 0.004670
1.000000 0.004649
1.000000 0.004627
1.000000 0.004606
1.000000 0.004584
1.000000 0.004563
1.000000 0.004542
1.000000 0.004522
1.000000 0.004501
1.000000 0.004481
.................
.........
```

のような形式で格納する必要がある.

※ 提出するのは得られたグラフをプリントアウトしたものと作成しプログラムです.
なお, プログラムファイル (テキストファイル) のプリンアウト方法は

```
iecsv{your_account}16%  a2ps prog.c | lpr [リターン]
```

ポストスクリプト・ファイルのプリントアウトは

```
iecsv{your_account}17% lpr sample.ps  [リターン]
```

です.