数値解析C （塩田）

2006年12月13日のレポート課題

- 提出期限 2006年12月27日 (水) 17:00
- 提出先 情報科学棟 512 号室 (塩田研究室)

課題 6次行列 A を

$$A = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 \end{pmatrix},$$

6次ベクトル b を

$$b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$$

と定めるとき、LU分解法を用いて方程式

$$Ax = b$$

の解 x を計算せよ。ただしレポートには、解 x だけでなく L, U も明記し、検算も怠らないこと。（ピボット選択は使わなくて良い。）

発展課題 行列やベクトルの成分、行列のサイズを変えてプログラムを実行し、自分の作ったプログラムの限界を探ってみよう。

参考 以下に2次行列限定のLU分解法プログラムのサンプルを付けておきます。(一昨年作ったので名前が 2004 になっていますが。)

- `http://lupus.is.kochi-u.ac.jp/~shiota/na06/na06.html` からダウンロードできます。
- 高知大学のホームページからも辿れます：

  学部・大学院 → 理学部 数理情報科学科 → 情報科学教室
  → 教員組織・研究の情報 → 教員組織
  → 塩田の「個人のホームページ」 → オンライン教材他

- 行列やベクトルを表す型を定義して関数を見易くしたつもりです。
- ポインタの使い方もこれを参考に。

```
/* lu2004.c  数値解析 C
LU 分解法, 2次元限定 version

gcc lu2004.c ; a.out
*/

#include<stdio.h>
#include<stdlib.h>

/* 行列のサイズを設定 */
#define SIZE 2

/* MATRIX, VECTOR という型を定義 */
typedef struct { double ent[SIZE][SIZE]; } MATRIX;
typedef struct { double ent[SIZE]; } VECTOR;

/* MATRIX a;
   と宣言すると a.ent[i][j] が a の (i,j)-成分を表す。
   VECTOR b;
   と宣言すると b.ent[i] が b の 第 i 成分を表す。 */

/* 行列を表示する関数 */
void matprint(MATRIX a);

/* ベクトルを表示する関数 */
void vecprint(VECTOR b);

/* 行列 A, B の積 C = A B を返す関数 */
MATRIX matmul(MATRIX a, MATRIX b);

/* 行列 A とベクトル b の積 c = A b を返す関数 */
VECTOR matvecmul(MATRIX a, VECTOR b);

/* 正方行列 A の LU 分解を計算する関数
   2つの行列を返すために、L と U はポインタで受け渡しする。 */
void lu(MATRIX a, MATRIX *l, MATRIX *u);

/* LU 分解法の前進代入 : L y = b の解 y を返す関数 */
VECTOR step2(MATRIX l, VECTOR b);

/* LU 分解法の後退代入 : U x = y の解 x を求める関数 */
VECTOR step3(MATRIX u, VECTOR y);

main()
{
        MATRIX a,l,u,c;
        VECTOR b,x,y,z;
        int i,j,n,k;

        /* ランダムな行列 A, ベクトル b の生成 */
        srand(time(NULL));
        for(i=0;i<SIZE;i++) for(j=0;j<SIZE;j++) a.ent[i][j]=(5.0*rand())/RAND_MAX;
        for(i=0;i<SIZE;i++) b.ent[i]=(5.0*rand())/RAND_MAX;
        printf("A :\n"); matprint(a); printf("\n");

        /* LU 分解の計算と検算 */
        lu(a,&l,&u);
        printf("L :\n"); matprint(l); printf("\n");
```

```
        printf("U :\n"); matprint(u); printf("\n");
        c=matmul(l,u);
        printf("検算 LU :\n"); matprint(c); printf("\n");

        /* 解 x の計算と検算 */
        y=step2(l,b);
        x=step3(u,y);
        printf("b :\n"); vecprint(b); printf("\n");
        printf("Ax = b の解 :\n"); vecprint(x); printf("\n");
        z=matvecmul(a,x);
        printf("検算 Ax :\n"); vecprint(z); printf("\n");
}

/*  実行例

A :
   1.5277566   1.7621387
   1.3235878   2.0830409

L :
   1.5277566   0.0000000
   1.3235878   0.5563937

U :
   1.0000000   1.1534159
   0.0000000   1.0000000

検算 LU :
   1.5277566   1.7621387
   1.3235878   2.0830409

b :
   4.9147008
   4.8965423

Ax = b の解 :
   1.8930249
   1.1478209

検算 Ax :
   4.9147008
   4.8965423

*/

void matprint(MATRIX a)
{
        int i,j;
        for(i=0;i<SIZE;i++){
                for(j=0;j<SIZE;j++) printf("%12.7lf",a.ent[i][j]);
                printf("\n");
        }
}

void vecprint(VECTOR b)
{
        int i;
        for(i=0;i<SIZE;i++) printf("%12.7lf\n",b.ent[i]);
}

/* 行列 A, B の積 C = A B を返す関数 */
MATRIX matmul(MATRIX a, MATRIX b)
{
```

```
        MATRIX c;
        int i,j,k;
        double x;

        for(i=0;i<SIZE;i++){
                for(j=0;j<SIZE;j++){
                        x=0.0;
                        for(k=0;k<SIZE;k++) x+=a.ent[i][k]*b.ent[k][j];
                        c.ent[i][j]=x;
                }
        }
        return c;
}

/* 行列 A とベクトル b の積 c = A b を返す関数 */
VECTOR matvecmul(MATRIX a, VECTOR b)
{
        VECTOR c;
        int i,k;
        double x;

        for(i=0;i<SIZE;i++){
                x=0.0;
                for(k=0;k<SIZE;k++) x+=a.ent[i][k]*b.ent[k];
                c.ent[i]=x;
        }
        return c;
}

/* 以下は2次行列限定 */

/* 正方行列 A の LU 分解を計算する関数 */
void lu(MATRIX a, MATRIX *l, MATRIX *u)
{
        l->ent[0][0]=a.ent[0][0];
        u->ent[0][1]=a.ent[0][1]/l->ent[0][0];
        l->ent[1][0]=a.ent[1][0];
        l->ent[1][1]=a.ent[1][1]-l->ent[1][0]*u->ent[0][1];

        l->ent[0][1]=0.0;
        u->ent[0][0]=1.0;
        u->ent[1][0]=0.0;
        u->ent[1][1]=1.0;
}
/* （注：pivot 選択は使っていない） */

/* LU 分解法の前進代入 : L y = b の解 y を返す関数 */
VECTOR step2(MATRIX l, VECTOR b)
{
        VECTOR y;
        y.ent[0]=b.ent[0]/l.ent[0][0];
        y.ent[1]=(b.ent[1]-l.ent[1][0]*y.ent[0])/l.ent[1][1];
        return y;
}

/* LU 分解法の後退代入 : U x = y の解 x を求める関数 */
VECTOR step3(MATRIX u, VECTOR y)
{
        VECTOR x;
        x.ent[1]=y.ent[1];
        x.ent[0]=y.ent[0]-u.ent[0][1]*x.ent[1];
        return x;
}
```