数値解析C （塩田）

2007年1月10日のレポート課題

- 提出期限 2007年1月24日 (水) 17:00
- 提出先 情報科学棟 512 号室 (塩田研究室)

課題 6次対称行列 A を

$$A = (a_{ij}) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 1 & 1/2 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1 & 1/2 & 1/3 & 1/4 & 1/4 & 1/4 \\ 1 & 1/2 & 1/3 & 1/4 & 1/5 & 1/5 \\ 1 & 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \end{pmatrix}$$

と定める。($a_{ij} = 1/\min(i,j)$ である。)
累乗法を用いて、A の絶対値最大の固有値 λ と、λ に対する固有ベクトル v を求めよ。(検算も怠らないこと。)

発展課題 デフレーションを行って、全ての固有値・固有ベクトルを求めよ。

参考 以下に3次行列の累乗法のサンプルプログラムを付けておきます。
(http://lupus.is.kochi-u.ac.jp/~shiota/na06/na06.html からダウンロードできます。)

```
/* power.c  数値解析 C
累乗法による最大固有値の計算

gcc power.c -lm
*/

/* 行列のサイズを設定 */
#define SIZE 3

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

/* 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);

/* ベクトル b, c の内積 (b,c) を返す関数 */
double innprod(VECTOR b, VECTOR c);

/* ベクトル b の長さ（ユークリッドノルム）を返す関数 */
double vecnorm(VECTOR b);

/* ベクトル b の c 倍（スカラー倍） x = cb を返す関数 */
VECTOR scavecmul(double c, VECTOR b);

/* ベクトル b, c の和 x = b+c を返す関数 */
VECTOR vecadd(VECTOR b, VECTOR c);

/* ベクトル b, c の差 x = b-c を返す関数 */
VECTOR vecsub(VECTOR b, VECTOR c);

main()
{
      MATRIX a;
      VECTOR v,v0,w;
      int i,j;
      double n,e1,e2,err=1.0e-10;
      double lambda;

      /* ランダムな行列 A の生成 */
      srand(time(NULL));
      for(i=0;i<SIZE;i++) for(j=0;j<SIZE;j++) a.ent[i][j]=(10.0*rand())/RAND_MAX-5;
      printf("A :\n"); matprint(a); printf("\n");

      /* ランダムな単位ベクトル v （初期値）の生成 */
      for(i=0;i<SIZE;i++) v.ent[i]=(5.0*rand())/RAND_MAX;
      n=vecnorm(v);
      v=scavecmul(1.0/n,v);

      /* 新 v <- ( Av 方向の単位ベクトル ) の反復 */
      do{
            v0=v; // 直前の v を v0 として保存
            v=matvecmul(a,v0);
            n=vecnorm(v);
            v=scavecmul(1.0/n,v);
            // 途中経過を出力する場合は次のコメントアウトをはずす
            // printf("v :\n"); vecprint(v); printf("\n");
            e1=vecnorm(vecsub(v,v0));  // v と  v0 の差を測定
            e2=vecnorm(vecadd(v,v0));  // v と -v0 の差を測定
      }while(e1>err && e2>err);        // 方向が安定したら終了

      /* 固有値の計算 */
      w=matvecmul(a,v);
      lambda=innprod(w,v)/innprod(v,v);

      /* 出力と検算 */
      printf("最大固有値 λ = %lf\n\n",lambda);
```

```
	printf("λに対する固有ベクトル v  :\n"); vecprint(v); printf("\n");
	printf("----- 検算 -----\n\n");
	printf("Av :\n"); vecprint(w); printf("\n");
	printf("λ v :\n"); vecprint(scavecmul(lambda,v)); printf("\n");
}

/*  実行例

A :
  -0.5686498  -3.1283745   3.9268336
  -3.5646055  -2.1349595  -4.2297066
   3.3722438   4.8335979   0.9993793

最大固有値 λ = -4.045492

λに対する固有ベクトル v  :
   0.7821787
   0.0967060
  -0.6155034

----- 検算 -----

Av :
  -3.1642976
  -0.3912232
   2.4900138

λ v :
  -3.1642976
  -0.3912232
   2.4900138

*/

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;
}

/* ベクトル b, c の内積 (b,c) を返す関数 */
double innprod(VECTOR b, VECTOR c)
{
      int i;
      double x=0.0;
      for(i=0;i<SIZE;i++) x+=b.ent[i]*c.ent[i];
      return x;
}

/* ベクトル b の長さ（ユークリッドノルム）を返す関数 */
double vecnorm(VECTOR b)
{
      return sqrt(innprod(b,b));
}

/* ベクトル b の c 倍（スカラー倍） x = cb を返す関数 */
VECTOR scavecmul(double c, VECTOR b)
{
      int i;
      VECTOR x;
      for(i=0;i<SIZE;i++) x.ent[i]=c*b.ent[i];
      return x;
}

/* ベクトル b, c の和 x = b+c を返す関数 */
VECTOR vecadd(VECTOR b, VECTOR c)
{
      int i;
      VECTOR x;
      for(i=0;i<SIZE;i++) x.ent[i]=b.ent[i]+c.ent[i];
      return x;
}

/* ベクトル b, c の差 x = b-c を返す関数 */
VECTOR vecsub(VECTOR b, VECTOR c)
{
      int i;
      VECTOR x;
      for(i=0;i<SIZE;i++) x.ent[i]=b.ent[i]-c.ent[i];
      return x;
}
```