レポート課題 No.2

川口廣伊智
学籍番号:051715223

2017/07/06

0　レポートについての注意

0.1　各課題の解答の構成について

まず課題の解釈をし、解答するためのプログラムを記載した。次に得られた結果を記載した。考察すべき内容があった場合は簡単な考察も付けた。すべての課題に考察がついているわけではない。

0.2　プログラムについて

各課題についてその計算を行うためのプログラムを全掲した。長ったらしいが、ソースコードの後に特に工夫した部分がどこかを記した。

1　基本課題 EX3-1

1.1　課題概要

LU 分解を用いて行列の行列式を計算するプログラムを作成した。

ソースコード 1　LU 分解を用いて行列の行列式を計算するプログラム

```
1  #include "matrix_util.h"
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* http://www.netlib.org/lapack/explore-html/d3/d6a/dgetrf_8f.html */
6  extern void dgetrf_(int *M, int *N, double *A, int *LDA, int*IPIV, int *INFO);
7
8  /* http://www.netlib.org/lapack/explore-html/d6/d49/dgetrs_8f.html */
9  extern void dgetrs_(char *TRANS, int *N, int *NRHS, double *A, int *LDA, int *IPIV,
10                      double *B, int *LDB, int *INFO);
11
12 int main(int argc, char** argv) {
13   char* filename;
14   FILE *fp;
15
16   int i, m, n;
```

```
17  double **a;
18  double *b;
19  double determinant = 1.0;
20
21
22  int *ipiv;
23  int info;
24  char trans = 'T';
25  int nrhs = 1;
26
27  if (argc < 2) {
28    fprintf(stderr, "Usage: %s inputfile\n", argv[0]);
29    exit(1);
30  }
31  filename = argv[1];
32
33  /* read matrix A and vector B from a file */
34  fp = fopen(filename, "r");
35  if (fp == NULL) {
36    fprintf(stderr, "Error: file can not open\n");
37    exit(1);
38  }
39  read_dmatrix(fp, &m, &n, &a);
40  if (m != n) {
41    fprintf(stderr, "Error: inconsistent number of equations\n");
42    exit(1);
43  }
44  read_dvector(fp, &n, &b);
45  if (m != n) {
46    fprintf(stderr, "Error: inconsistent number of equations\n");
47    exit(1);
48  }
49  printf("Matrix A:\n");
50  fprint_dmatrix(stdout, n, n, a);
51
52  /* perform LU decomposition */
53  ipiv = alloc_ivector(n);
54  dgetrf_(&n, &n, &a[0][0], &n, &ipiv[0], &info);
55  if (info != 0) {
56    fprintf(stderr, "Error: LAPACK::dgetrf failed\n");
57    exit(1);
58  }
59  printf("Result of LU decomposition:\n");
60  fprint_dmatrix(stdout, n, n, a);
```

```
61    printf("Pivot for LU decomposition:\n");
62    fprint_ivector(stdout, n, ipiv);
63
64
65    /* calculate the determinant of given a[][] */
66    for(i=0;i<n;i++){
67      determinant = determinant * a[i][i];
68    }
69
70    /* output the value of determinant */
71    printf("determinant is %lf\n", determinant);
72
73
74
75  }
```

このプログラムを用いて Vandermonde 行列の行列式を計算した。今回は 5 行 5 列で

$$(x_1, x_2, x_3, x_4, x_5) = (1, 2, 3, 4, 5)$$

の Vandermonde 行列を用いた。5 行 5 列程度の行列なら手打ちで成分を書きだすこともできなくはないが、もっと大きなサイズの Vandermonde 行列を作るとすると手打ちでは苦しい。なので $(x_1, x_2, \cdots, x_n)$ を指定して Vandermonde 行列を生成するプログラムも作成した。

ソースコード 2　LU 分解を用いて行列の行列式を計算するプログラム

```
1   /* output vandermonde matrix */
2
3   #include<stdio.h>
4   #include<math.h>
5
6   int main(void){
7
8     double x[5] = {1.0, 2.0, 3.0, 4.0, 5.0}; //choose numbers you like
9     double a[5][5];  // depends on the size of x[]
10    int i, j, n;
11    n = 5; // depends on the size of x[]
12
13
14    printf("%d %d\n", n, n); //output the size of matrix
15
16    for (i = 0;i < n;i++){
17      for (j = 0;j < n;j++){
18        a[i][j] = pow(x[j], i);
19        if(j < n - 1){
20          printf("%lf ", a[i][j]);
21        }else{
```

```
        printf("%lf\n", a[i][j]);
      }
    }
  }

  return 0;
}
```

行列のサイズが大きくなると Vandermonde 行列の行列式も手計算では一苦労なのでこれも $(x_1, x_2, \cdots, x_n)$ を指定して自動で計算するプログラムを作成した。

ソースコード 3　LU 分解を用いて行列の行列式を計算するプログラム

```
/* calculate theoretical value of vandermonde determinant */

#include<stdio.h>

int main(void){

  int i, j, n;

  double determinant;
  determinant = 1.0;

  double x[] = {1.9, 2.0, 3.1, 4.2, 5.4}; //choose numbers you like

  n = sizeof(x) / 8; //n is the size of x[]

  for(i=0;i<n;i++){
    for(j=i+1;j<n;j++){

      determinant *= x[j] - x[i]; //calculate the determinant following useful formula

    }
  }

  printf("%lf\n", determinant); //output the value of the determinant

  return 0;
}
```

1.2 結果

1.3 考察

2 基本課題 EX3-2

3 基本課題 EX3-3

4 応用課題 EX3-1

4.1 実験概要

pointer.c のソースコードを見て出力される結果を予想し、実際にコンパイルして得た出力と比較した。
まず、ベクトルの方について予想される出力を考え、結果と比較する。行列の方についても同様にする。

4.2 実験結果

pointer.c のソースコードはこのようであった。

ソースコード 4　LU 分解を用いて行列の行列式を計算するプログラム

```
1  #include "matrix_util.h"
2  #include <stdio.h>
3
4  int main() {
5    int n, i, j;
6    double *v;
7    double **m;
8    n = 10;
9
10   /* test for vector */
11   v = alloc_dvector(n);
12   for (i = 0; i < n; ++i) v[i] = i;
13   fprint_dvector(stdout, n, v);
14
15   printf("v     = %lu\n", (long)v);
16   printf("&v[0] = %lu\n", (long)&v[0]);
17
18   printf("(v+2) = %lu\n", (long)(v+2));
19   printf("&v[2] = %lu\n", (long)&v[2]);
20
21   printf("*v       = %10.5f\n", *v);
22   printf("v[0]     = %10.5f\n", v[0]);
23
24   printf("*(v+2)   = %10.5f\n", *(v+2));
25   printf("v[2]     = %10.5f\n", v[2]);
26
```

```
27    printf("(v+2)[3] = %10.5f\n", (v+2)[3]);
28    printf("*(v+2+3) = %10.5f\n", *(v+2+3));
29
30    free_dvector(v);
31
32    /* test for matrix */
33    m = alloc_dmatrix(n, n);
34    for (i = 0; i < n; ++i)
35      for (j = 0; j < n; ++j)
36        m[i][j] = 100 * i + j;
37    fprint_dmatrix(stdout, n, n, m);
38
39    printf("m        = %lu\n", (long)m);
40    printf("&m[0]    = %lu\n", (long)&m[0]);
41
42    printf("m[0]     = %lu\n", (long)m[0]);
43    printf("&m[0][0] = %lu\n", (long)&m[0][0]);
44
45    printf("m[2]     = %lu\n", (long)m[2]);
46    printf("&m[2][0] = %lu\n", (long)&m[2][0]);
47
48    printf("m+2      = %lu\n", (long)(m+2));
49    printf("&m[2]    = %lu\n", (long)&m[2]);
50
51    printf("(*(m+2))[3] = %10.5f\n", (*(m+2))[3]);
52    printf("*(*(m+2)+3) = %10.5f\n", *(*(m+2)+3));
53    printf("m[2][3]     = %10.5f\n", m[2][3]);
54
55    printf("*(m+2)[3]   = %10.5f\n", *(m+2)[3]);
56    printf("*((m+2)[3]) = %10.5f\n", *((m+2)[3]));
57    printf("*(m[5])     = %10.5f\n", *(m[5]));
58    printf("m[5][0]     = %10.5f\n", m[5][0]);
59
60    free_dmatrix(m);
61  }
```

6 行目で v がポインタ変数として定義されている。

15 行目のプリント関数は v を出力している。v は v[0] のアドレスを表すので出力されるのは v[0] のアドレス (具体的にはわからない) であるはずだ。

16 行目は&v[0] を出力している。&v[0] は v[0] が格納されているアドレスを表すので出力されるのは v[0] のアドレス (具体的にはわからない) であるはずだ。

18 行目は v+2 を出力している。v+2 は v[2] のアドレスを表すので出力されるのは v[2] のアドレスである。*v は double 型で定義されていたので v[0] のアドレスに 16 を足したものが出力される。

19 行目は&v[2] を出力している。&v[2] は v[2] が格納されているアドレスを表すので出力されるのは v[2] のアドレスである。

21 行目は*v を出力している。*v は v のアドレスに格納されている数値を表すので出力されるのは v[0] の値、すなわち 0.00000 である。

22 行目は v[0] を出力している。これは取りも直さず v[0] の値を表すので出力されるのは 0.00000 である。

5　応用課題 EX3-2

6　基本課題 EX3-3

7　基本課題 EX4-1

参考文献

[1] 増原英彦 + 東京大学情報教育連絡会『情報科学入門 Ruby を使って学ぶ』(東京大学出版会, 2010)

[2] http://www.cp.cmc.osaka-u.ac.jp/~kikuchi/texts/conservation.pdf