プログラミングiii 自由課題

29119020 下村理雄

2018年12月20日

1　実装内容

本課題では、行列・ベクトルの演算を実装する。具体的に、以下を実装した。

- 行列同士の和・差・積。
- 行列の転置。
- ベクトル同士の和・差・内積。
- 行列とベクトルの積。
- 行列・ベクトルを用いた連立方程式を解く。

ここでは、5行5列行列と5次元ベクトルを考えている。また、連立方程式の解法はLU分解を用いることで実装する。

2　必要条件

制御構文を利用　行列・ベクトルの演算、行列・ベクトルの内容を表示するときに利用する。

関数の利用　行列・ベクトルを演算するための関数、行列・ベクトルを表示させる関数で利用する。

ポインタの利用　行列・ベクトルの内容を書き換えるときに利用する。

ファイル操作　行列・ベクトルの各要素をプログラムを実行するごとに手入力するのは大変である。そこで、その各要素の内容をあらかじめテキストファイルで書くことで、労力を軽減させる。プログラム中ではそのテキストファイルを読み込むためにファイル操作を利用する。

250 行以上（コメント含む）　290 行以上である。

3　ソースコード（コメント含む）

以下に「main.c」のソースコードを示す。

```
 1:#include<stdio.h>

 2://行列の構造体
 3:typedef struct matrix{
 4:  double m[5][5];
 5:}Matrix;

 6://ベクトルの構造体
 7:typedef struct vector{
 8:  double v[5];
 9:}Vector;

10: //行列の要素を取得
11: //getMatrix(行列アドレス,行の数,列の数)
12:double getMatrix(Matrix *matrix,int i,int j){
13:  if(0 <= i && i <= 5){
14:    if(0 <= j && j <= 5){
15:      return matrix -> m[i][j];
16:    }
17:  }
18:  return -1;
19:}

20://行列の要素をセット
21://setMatrix(行列アドレス,行の数,列の数,入力値)
22:void setMatrix(Matrix *matrix,int i,int j,double
   n){
23:  if(0 <= i && i <= 5){
24:    if(0 <= j && j <= 5){
25:      matrix -> m[i][j] = n;
26:    }
27:  }
28:}
```

```
29://行列の転置
30://trans(行列アドレス)
31:void trans(Matrix *matrix){
32:   Matrix temp = *matrix;
33:   int i,j;
34:   for(i = 0;i < 5;i++){
35:     for(j = 0;j < 5;j++){
36:       setMatrix(matrix,i,j,getMatrix(&temp,j,i));
37:     }
38:   }
39:}

40://行列と行列の和
41:Matrix addMat(Matrix *a,Matrix *b){
42:   Matrix result;
43:   int i,j;
44:   for(i = 0;i < 5;i++){
45:     for(j = 0;j < 5;j++){
46:       double t = getMatrix(a,i,j) + getMatrix(b,i
   ,j);
47:       setMatrix(&result,i,j,t);
48:     }
49:   }
50:   return result;
51:}

52://行列と行列の差
53:Matrix subMat(Matrix *a,Matrix *b){
54:   Matrix result;
55:   int i,j;
56:   for(i = 0;i < 5;i++){
57:     for(j = 0;j < 5;j++){
58:       double t = getMatrix(a,i,j) - getMatrix(b,i
   ,j);
59:       setMatrix(&result,i,j,t);
60:     }
61:   }
62:   return result;
```

```
63:}

64://行列と行列の積
65:Matrix mulMat(Matrix *a,Matrix *b){
66:  Matrix result;
67:  int i,j,k;
68:  for(i = 0;i < 5;i++){
69:    for(j = 0;j < 5;j++){
70:      double t = 0;
71:      for(k = 0;k < 5;k++){
72:        t += getMatrix(a,i,k) * getMatrix(b,k,j);
73:        setMatrix(&result,i,j,t);
74:      }
75:    }
76:  }
77:  return result;
78:}

79://行列の内容を表示
80://printMat(行列アドレス)
81:void printMat(Matrix *matrix){
82:  int i,j;
83:  for(i = 0;i < 5;i++){
84:    for(j = 0;j < 5;j++){
85:      printf("[%3.2f] ",getMatrix(matrix,i,j));
86:    }
87:    printf("\n");
88:  }
89:}

90:void inputMat(char *file,Matrix *matrix){
91:  FILE  *fp;
92:  if((fp = fopen(file,"r")) != NULL){
93:    int i = 0;
94:    double t[5] = {0};
95:    int j;
96:    while(fscanf(fp,"%lf %lf %lf %lf %lf",&t[0],&
   t[1],&t[2],&t[3],&t[4]) != EOF && i < 5){
97:      for(j = 0;j < 5;j++){
```

```
 98:         setMatrix(matrix,i,j,t[j]);
 99:       }
100:       i++;
101:     }
102:     fclose(fp);
103:   }
104:}

105://ベクトルの要素を取得
106://getVector(ベクトルアドレス,インデックス)
107:double getVector(Vector *vector,int i){
108:   if(0 <= i && i <= 5){
109:     return vector -> v[i];
110:   }
111:   return -1;
112:}

113://ベクトルの要素をセット
114://setVector(ベクトルアドレス,インデックス,入力値)
115:void setVector(Vector *vector,int i,double n){
116:   if(0 <= i && i <= 5){
117:     vector -> v[i] = n;
118:   }
119:}

120://ベクトルの和
121:Vector addVec(Vector *a,Vector *b){
122:   Vector result;
123:   int i;
124:   for(i = 0;i < 5;i++){
125:     double t = getVector(a,i) + getVector(b,i);
126:     setVector(&result,i,t);
127:   }
128:   return result;
129:}

130://ベクトルの差
131:Vector subVec(Vector *a,Vector *b){
132:   Vector result;
```

```
133:  int i;
134:  for(i = 0;i < 5;i++){
135:    double t = getVector(a,i) - getVector(b,i);
136:    setVector(&result,i,t);
137:  }
138:  return result;
139:}

140://ベクトルの内積
141:double mulVec(Vector *a,Vector *b){
142:  double t;
143:  int i;
144:  for(i = 0;i < 5;i++){
145:    t += getVector(a,i) * getVector(b,i);
146:  }
147:  return t;
148:}

149://行列とベクトルの積
150:Vector mul(Matrix *a,Vector *b){
151:  Vector result;
152:  int i,j,k;
153:  for(i = 0;i < 5;i++){
154:    for(j = 0;j < 5;j++){
155:      double t = 0;
156:      for(k = 0;k < 5;k++){
157:        t += getMatrix(a,i,k) * getVector(b,k);
158:        setVector(&result,i,t);
159:      }
160:    }
161:  }
162:  return result;
163:}

164://ベクトルの内容を表示
165://printVec(ベクトルアドレス)
166:void printVec(Vector *vector){
167:  int i;
168:  for(i = 0;i < 5;i++){
```

```
169:    printf("[%3.2f]\n",getVector(vector,i));
170:  }
171:}

172:void inputVec(char *file,Vector *vector){
173:  FILE *fp;
174:  if((fp = fopen(file,"r")) != NULL){
175:    int i = 0;
176:    int t = 0;
177:    while(fscanf(fp,"%d",&t) != EOF && i < 5){
178:      setVector(vector,i,(double)t);
179:      i++;
180:    }
181:    fclose(fp);
182:  }
183:}

184://LU分解
185://LUD(行列アドレス)
186:void LUD(Matrix *a){
187:  int n = 5;
188:  int k,i,j;
189:  for(k = 0; k < n - 1;k++){
190:    for(i =  k + 1;i < n;i++){
191:      for(j = k;j < n;j++){
192:        if(k == j){
193:          double t = getMatrix(a,i,j) / getMatrix
   (a,k,j);
194:          setMatrix(a,i,j,t);
195:        }else{
196:          double t = - getMatrix(a,i,k) *
   getMatrix(a,k,j) + getMatrix(a,i,j);
197:          setMatrix(a,i,j,t);
198:        }
199:      }
200:    }
201:  }
202:}
```

```
203://LU分解後、答えを求める
204://answer(行列アドレス,ベクトルアドレス)
205:Vector answer(Matrix *a,Vector *b){
206:  //ベクトルx,y(ベクトルxが解答)
207:  Vector x,y;
208:  int n = 5;
209:  //ベクトルyを求める
210:  int i;
211:  for(i = 0;i < n;i++){
212:    setVector(&y,i,getVector(b,i));
213:    double t = 0;
214:    int j;
215:    for(j = 1;j <= i;j++){
216:      t += - getMatrix(a,i,j - 1) * getVector(&y,
   j - 1);
217:      setVector(&y,i,t);
218:    }
219:  }
220:  //ベクトルxを求める
221:  for(i = n - 1;i >= 0;i--){
222:    setVector(&x,i,getVector(&y,i));
223:    double t = 0;
224:    int j;
225:    for(j = n - 1;j > i;j--){
226:      t += - getMatrix(a,i,j) * getVector(&x,j);
227:      setVector(&x,i,t);
228:    }
229:    double r = getVector(&x,i) / getMatrix(a,i,i
   );
230:    setVector(&x,i,r);
231:  }
232:  return x;
233:}

234:int main(int argc, char const *argv[]) {
235:  //行列
236:  Matrix mat1,mat2;
237:  //ベクトル
238:  Vector vec1,vec2;
```

```
239:  //行列1の読み込む
240:  char fileM1[] = "MatA.txt";
241:  inputMat(fileM1,&mat1);

242:  //行列2の読み込む
243:  char fileM2[] = "MatB.txt";
244:  inputMat(fileM2,&mat2);

245:  //行列内容表示
246:  printf("Mat A\n");
247:  printMat(&mat1);
248:  printf("Mat B\n");
249:  printMat(&mat2);

250:  // 行列の転置
251:  trans(&mat1);
252:  printf("A^T\n");
253:  printMat(&mat1);

254:  trans(&mat1);

255:  //行列と行列の和の確認
256:  mat2 = addMat(&mat1,&mat2);
257:  printf("Mat A + Mat B\n");
258:  printMat(&mat2);

259:  //行列と行列の差の確認
260:  mat2 = subMat(&mat2,&mat1);
261:  printf("Mat B - Mat A\n");
262:  printMat(&mat2);

263:  //行列と行列の積の確認
264:  mat2 = mulMat(&mat2,&mat1);
265:  printf("Mat A * Mat B\n");
266:  printMat(&mat2);

267:  //行列の転置を確認
268:  trans(&mat2);
```

```
269:  printf("Mat B t\n");
270:  printMat(&mat2);

271:  //ベクトル1を読み込む
272:  char fileV1[] = "VecA.txt";
273:  inputVec(fileV1,&vec1);

274:  //ベクトル2を読み込む
275:  char fileV2[] = "VecB.txt";
276:  inputVec(fileV2,&vec2);

277:  //ベクトル内容表示
278:  printf("Vec C\n");
279:  printVec(&vec1);
280:  printf("Vec D\n");
281:  printVec(&vec2);

282:  //ベクトルとベクトルの和の確認
283:  vec2 = addVec(&vec1,&vec2);
284:  printf("Vec C + Vec D\n");
285:  printVec(&vec2);

286:  //ベクトルとベクトルの差の確認
287:  vec2 = subVec(&vec2,&vec1);
288:  printf("Vec C - Vec D\n");
289:  printVec(&vec2);

290:  //ベクトルとベクトルの積の確認
291:  int res = mulVec(&vec1,&vec2);
292:  printf("Vec C * Vec D\n");
293:  printf("%d\n",res);

294:  //行列をLU分解
295:  LUD(&mat1);

296:  //LU分解後の内容を表示
297:  printf("LUD(Mat A)\n");
298:  printMat(&mat1);
```

```
299:  //連立方程式の解を表示
300:  printf("Mat AとVec Cの方程式の解\n");
301:  vec1 = answer(&mat1,&vec1);
302:  printVec(&vec1);
303:  return 0;
304:}
```

4 実行例

実行例は以下のようになる。

```
Mat A
[1.00] [-1.00] [1.00] [-1.00] [1.00]
[12.00] [-6.00] [2.00] [0.00] [0.00]
[1.00] [1.00] [1.00] [1.00] [1.00]
[12.00] [6.00] [2.00] [0.00] [0.00]
[4.00] [3.00] [2.00] [1.00] [0.00]
Mat B
[1.00] [2.00] [3.00] [4.00] [5.00]
[2.00] [3.00] [4.00] [5.00] [6.00]
[3.00] [4.00] [5.00] [6.00] [7.00]
[4.00] [5.00] [6.00] [7.00] [8.00]
[5.00] [6.00] [7.00] [8.00] [9.00]
A^T
[1.00] [12.00] [1.00] [12.00] [4.00]
[-1.00] [-6.00] [1.00] [6.00] [3.00]
[1.00] [2.00] [1.00] [2.00] [2.00]
[-1.00] [0.00] [1.00] [0.00] [1.00]
[1.00] [0.00] [1.00] [0.00] [0.00]
Mat A + Mat B
[2.00] [1.00] [4.00] [3.00] [6.00]
[14.00] [-3.00] [6.00] [5.00] [6.00]
[4.00] [5.00] [6.00] [7.00] [8.00]
[16.00] [11.00] [8.00] [7.00] [8.00]
[9.00] [9.00] [9.00] [9.00] [9.00]
Mat B - Mat A
[1.00] [2.00] [3.00] [4.00] [5.00]
[2.00] [3.00] [4.00] [5.00] [6.00]
[3.00] [4.00] [5.00] [6.00] [7.00]
```

```
[4.00] [5.00] [6.00] [7.00] [8.00]
[5.00] [6.00] [7.00] [8.00] [9.00]
Mat A * Mat B
[96.00] [29.00] [26.00] [7.00] [4.00]
[126.00] [32.00] [34.00] [8.00] [6.00]
[156.00] [35.00] [42.00] [9.00] [8.00]
[186.00] [38.00] [50.00] [10.00] [10.00]
[216.00] [41.00] [58.00] [11.00] [12.00]
Mat B t
[96.00] [126.00] [156.00] [186.00] [216.00]
[29.00] [32.00] [35.00] [38.00] [41.00]
[26.00] [34.00] [42.00] [50.00] [58.00]
[7.00] [8.00] [9.00] [10.00] [11.00]
[4.00] [6.00] [8.00] [10.00] [12.00]
Vec C
[1.00]
[0.00]
[8.00]
[0.00]
[1.00]
Vec D
[1.00]
[3.00]
[5.00]
[7.00]
[9.00]
Vec C + Vec D
[2.00]
[3.00]
[13.00]
[7.00]
[10.00]
Vec C - Vec D
[1.00]
[3.00]
[5.00]
[7.00]
[9.00]
Vec C * Vec D
```

```
50
LUD(Mat A)
[1.00] [-1.00] [1.00] [-1.00] [1.00]
[12.00] [6.00] [-10.00] [12.00] [-12.00]
[1.00] [0.33] [3.33] [-2.00] [4.00]
[12.00] [3.00] [6.00] [-12.00] [-0.00]
[4.00] [1.17] [2.90] [0.27] [-1.60]
Mat AとVec Cの方程式の解
[0.04]
[0.00]
[-0.23]
[-0.00]
[0.19]
```

なお、以上の実行例は現状あるテキストファイルを書き換えると違う結果になる。

5 プログラムの解説

構造体

- Matrix:2 次元配列をメンバーにもつ構造体
- Vector:1 次元配列をメンバーに持つ構造体

関数

- getMatrix:行列の各要素を取得
- setMatrix:行列の要素に値を設定
- trans:行列の転置
- addMat:行列同士の和
- subMat:行列同士の差
- mulMat:行列同士の積
- printMat:行列の各要素を表示
- inputMat:行列をファイルから読み込む
- getVector:ベクトルの各要素を取得

- setVector:ベクトルの要素に値を設定
- addVec:ベクトル同士の和
- subVec:ベクトル同士の差
- mulVec:ベクトル同士の内積
- mul:行列とベクトルの積
- printVec:ベクトルの内容を表示
- inputVec:ベクトルをファイルから読み込む
- LUD:LU 分解
- answer:連立方程式を解く

6 感想

- 自由課題の課題を決めるのが難しかった。
- この課題にした理由は、科学技術計算で学んだ C++の手法を C 言語でも扱えるようにしたいと考えたからである。
- プログラミン i&ii で java を学んでいたため、オブジェクト指向の考え方を使ってしまうことがあり、大変であった。