プログラミング演習 II レポート

課題名：課題３　モジュール設計

学籍番号	212273B
氏名	高木 壱哲
提出日	令和　　5 年　　6 月　28 日
書式修正版提出日	令和　　5 年　　月　　日

以下の項目が正しく記載されているかを確認し，「自己チェック」欄に○をつけること．

項目		自己チェック
(1) 表紙に学籍番号，氏名等の必要事項を記載したか		○
(2) 課題の目的を記載したか【1】		○
(3) ページ番号を記載したか		○
(4) 図は図の下に図番号付で図タイトルを記載し，表は表の上に表番号付で表タイトルを記載し，全ての図表を本文中で引用したか		○
必須課題１	(5) 課題の概要を記載したか【2.1】	○
	(6) LU 分解とその結果を用いた連立方程式の解法を数式，図，例などを用いて詳細に説明したか【2.2】	○
	(7) vector.h，vector.c，matrix.h，matrix.c，lu.h，lu.c，lu_test.c のプログラムリストを掲載し，それぞれの関数などを説明したか【2.3】	○
	(8) プログラムの出力結果を掲載し，結果を考察したか【2.4】	○
必須課題２	(9) 課題の概要を記載したか【3.1】	○
	(10) ２つの連立方程式それぞれの手計算で得られた解とプログラムで得られた解を掲載したか【3.2】	○
	(11) 得られた結果について，考察したか【3.3】	○
(12) 参考文献の書式は正確か		○

【】は対応する節

1. 目的

モジュール設計について学ぶ。デバッグを容易にするため・複数人で開発するため・コンパイル時間短縮のためなどで、複数のファイルに分割してプログラムを作成し、最終的に 1 つの実行形式にまとめる分割コンパイルについて学ぶ。

2. 必須課題 1

2.1. 概要

LU 分解モジュールの未実装関数（lu_solve()）を実装し、LU 分解を用いて連立方程式を解くアプリケーションを完成させる。

2.2. LU 分解の説明

LU 分解とは、N×N の対角成分より右上の要素が全て 0 である下三角行列 L と、N×N の対角成分が全て 1 で対角成分より左下の要素が全て 0 の上三角行列を L とするとき、N×N の正方行列 A を A = LU の形に分解することである。

以下の図 1 に $A = \begin{bmatrix} 8 & 16 & 24 & 32 \\ 2 & 7 & 12 & 17 \\ 6 & 17 & 32 & 59 \\ 7 & 22 & 46 & 105 \end{bmatrix}$ の場合についての例を示す。

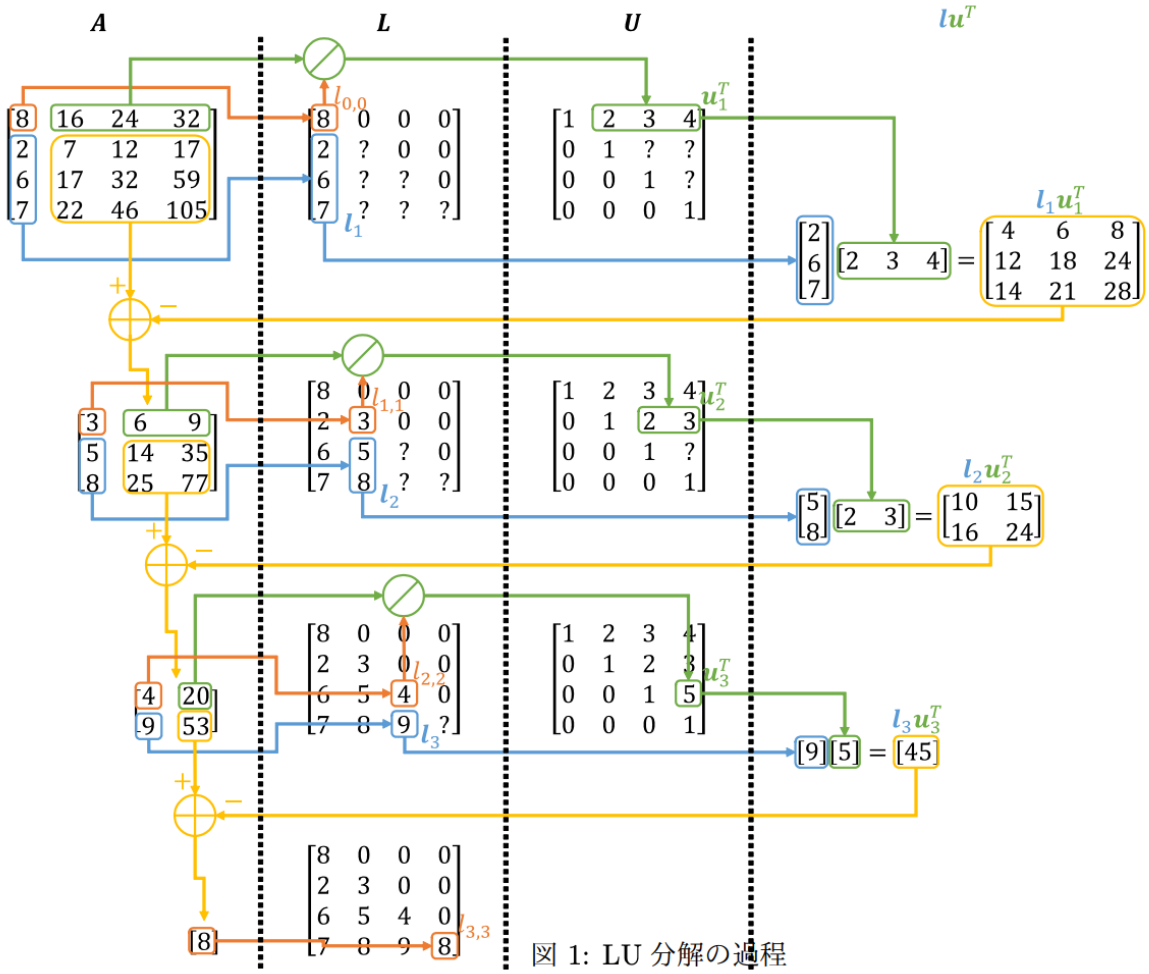

図 1: LU 分解の過程

2.3. プログラムリストの説明

vector.h のソースリストを以下のリスト 1 に、vector.c のソースリストを以下のリスト 2 に、matrix.h のソースリストを以下のリスト 3 に、matrix.c のソースリストを以下のリスト 4 に、lu.h のソースリストを以下のリスト 5 に、lu.c のソースリストを以下のリスト 6 に、lu_test.c のソースリストを以下のリスト 7 にそれぞれ示す。

リスト 1　vector.h のソースリスト

```
 1:    #ifndef VECTOR_H
 2:    #define VECTOR_H
 3:
 4:    #include <stdio.h>
 5:    #include <stdlib.h>
 6:
 7:    /**
 8:     * @struct vector
 9:     * @brief 1 次元配列構造体
10:     */
11:    typedef struct {
12:       int size; /**< サイズ */
13:       double* val; /**< 配列要素 */
14:    } vector;
15:
16:    /**
17:     * @brief サイズ n の 1 次元配列を動的に確保
18:     * @param[in] n サイズ
19:     * @param[out] vec 1 次元配列
20:     */
21:    void vector_new(int n, vector* vec);
22:
23:    /**
24:     * @brief 確保した 1 次元配列を開放
25:     * @param[in,out] vec 開放する 1 次元配列
26:     */
27:    void vector_delete(vector* vec);
28:
29:    /**
30:     * @brief 1 次元配列 vec2 を vec1 にコピー
31:     * @param[out] vec1 コピー先
32:     * @param[in] vec2 コピー元
33:     * @retval 1 成功
34:     * @retval 0 失敗（vec1 と vec2 のサイズ不整合）
35:     */
36:    int vector_copy(vector* vec1, const vector* vec2);
37:
38:    /**
39:     * @breif 1 次元配列 vec を format に従って標準出力に表示
40:     * @param[in] vec 出力する 1 次元配列
41:     * @param[in] format 出力形式（prinf の第 1 引数）
42:     */
43:    void vector_print(const vector* vec, const char* format);
44:
45:    #endif  /* VECTOR_H */
```

リスト 2　vector.c のソースリスト

```
 1:    #include "vector.h"
 2:
 3:    void vector_new(int n, vector* vec) {
```

```
 4:        if (n <= 0) {
 5:           fprintf(stderr, "allocation failure in vector_new()¥n");
 6:           exit(EXIT_FAILURE);
 7:        }
 8:        else {
 9:           vec->val = (double*)malloc(n * sizeof(double));
10:           if (vec->val == NULL) {
11:              fprintf(stderr, "allocation failure in vector_new()¥n");
12:              exit(EXIT_FAILURE);
13:           }
14:           vec->size = n;
15:        }
16:    }
17:
18:    void vector_delete(vector* vec) {
19:        free(vec->val);
20:    }
21:
22:    int vector_copy(vector* vec1, const vector* vec2) {
23:        int i;
24:
25:        if (vec1->size != vec2->size) {
26:           return 0;
27:        }
28:        for (i = 0; i < vec1->size; ++i) {
29:           vec1->val[i] = vec2->val[i];
30:        }
31:        return 1;
32:    }
33:
34:    void vector_print(const vector* vec, const char* format) {
35:        int i;
36:        printf("[");
37:        for (i = 0; i < vec->size; ++i) {
38:           printf(format, vec->val[i]);
39:           if (i < vec->size - 1) {
40:              printf(", ");
41:           }
42:        }
43:        printf("]¥n");
44:    }
```

リスト3　matrix.h のソースリスト

```
 1:    #ifndef MATRIX_H
 2:    #define MATRIX_H
 3:
 4:    #include <stdio.h>
 5:    #include <stdlib.h>
 6:
 7:    /**
 8:     * @struct matrix
 9:     * @brief 2 次元配列構造体
10:     */
11:    typedef struct {
12:        int size1; /**< 行数 */
13:        int size2; /**< 列数 */
14:        double** val; /**< 配列要素 */
15:    } matrix;
16:
17:    /**
18:     * @brief m×n の 2 次元配列を動的に確保
19:     * @param[in] m 行数
20:     * @param[in] n 列数
21:     * @param[out] mat 2 次元配列
22:     */
```

```
23:     void matrix_new(int m, int n, matrix* mat);
24:
25:     /**
26:      * @brief 確保した 2 次元配列を開放
27:      * @param[in,out] mat 開放する 2 次元配列
28:      */
29:     void matrix_delete(matrix* mat);
30:
31:     /**
32:      * @brief 2 次元配列 mat2 を mat1 にコピー
33:      * @param[out] mat1 コピー先
34:      * @param[in] mat2 コピー元
35:      * @retval 1 成功
36:      * @retval 0 失敗（mat1 と mat2 のサイズ不整合）
37:      */
38:     int matrix_copy(matrix* mat1, const matrix* mat2);
39:
40:     /**
41:      * @brief 2 次元配列 mat を format に従って標準出力に表示
42:      * @param[in] mat 出力する 2 次元配列
43:      * @param[in] format 出力形式（prinf の第 1 引数）
44:      */
45:     void matrix_print(const matrix* mat, const char* format);
46:
47:     #endif  /* MATRIX_H */
```

リスト 4　matrix.c のソースリスト

```
 1:     #include "matrix.h"
 2:
 3:     void matrix_new(int m, int n, matrix* mat) {
 4:         int i;
 5:
 6:         if ((m <= 0) || (n <= 0)) {
 7:             fprintf(stderr, "allocation failure in matrix_new()¥n");
 8:             exit(EXIT_FAILURE);
 9:         }
10:         mat->val = (double**)malloc(m * sizeof(double*));
11:         if (mat->val == NULL) { /* 領域割当に失敗したら */
12:             fprintf(stderr, "allocation failure in matrix_new()¥n");
13:             exit(EXIT_FAILURE);
14:         }
15:         for (i = 0; i < m; ++i) {
16:             mat->val[i] = (double*)malloc(n * sizeof(double));
17:             if (mat->val[i] == NULL) { /* 領域割当に失敗したら */
18:                 while (--i >= 0) {
19:                     free(mat->val[i]); /* 領域開放 */
20:                 }
21:                 free(mat->val);
22:                 fprintf(stderr, "allocation failure in matrix_new()¥n");
23:                 exit(EXIT_FAILURE);
24:             }
25:         }
26:         mat->size1 = m;
27:         mat->size2 = n;
28:     }
29:
30:     void matrix_delete(matrix* mat) {
31:         int i;
32:         for (i = 0; i < mat->size1; ++i) {
33:             free(mat->val[i]);
34:         }
35:         free(mat->val);
36:         mat->size1 = mat->size2 = 0;
37:     }
```

```
 38:
 39:    int matrix_copy(matrix* mat1, const matrix* mat2) {
 40:       int i, j;
 41:       if ((mat1->size1 != mat2->size1) || (mat2->size2 != mat2->size2)) {
 42:          return 0;
 43:       }
 44:       for (i = 0; i < mat1->size1; ++i) {
 45:          for (j = 0; j < mat1->size2; ++j) {
 46:             mat1->val[i][j] = mat2->val[i][j];
 47:          }
 48:       }
 49:       return 1;
 50:    }
 51:
 52:    void matrix_print(const matrix* mat, const char* format) {
 53:       int i, j;
 54:       printf("[");
 55:       for (i = 0; i < mat->size1; ++i) {
 56:          for (j = 0; j < mat->size2; ++j) {
 57:             printf(format, mat->val[i][j]);
 58:             if (j < mat->size2 - 1) {
 59:                printf(", ");
 60:             }
 61:          }
 62:          if (i == mat->size1 - 1) {
 63:             printf("]");
 64:          }
 65:          printf("¥n");
 66:       }
 67:    }
```

リスト 5　lu.h のソースリスト

```
  1:    #ifndef LU_H
  2:    #define LU_H
  3:
  4:    #include "vector.h"
  5:    #include "matrix.h"
  6:
  7:    /**
  8:     * @brief LU 分解（A=LU）
  9:     * @param[out] L 下三角行列
 10:     * @param[out] U 三角行列：対角成分は全て 1
 11:     * @param[in] A LU 分解対象行列
 12:     * @return 行列式
 13:     */
 14:    double lu_decomp(matrix* L, matrix* U, const matrix* A);
 15:
 16:    /**
 17:     * @brief LU 分解を用いて連立方程式を解く
 18:     * @param[out] x 解ベクトル
 19:     * @param[in] L 三角行列
 20:     * @param[in] U 上三角行列：対角成分は全て 1
 21:     * @param[in] b 右辺ベクトル
 22:     */
 23:    void lu_solve(vector* x, const matrix* L, const matrix* U, const vector*
b);
 24:
 25:    #endif  /* LU_H */
```

リスト 5　lu.c のソースリスト

```
  1:    #include "lu.h"
  2:
  3:    double lu_decomp(matrix* L, matrix* U, const matrix* A) {
```

```
 4:        int i, j, k;
 5:        int n = A->size1;
 6:        double det;
 7:
 8:        /* A を L にコピー */
 9:        matrix_copy(L, A);
10:        det = 1.0;
11:        for (i = 0; i < n; ++i) {
12:            det *= L->val[i][i];
13:            /* L の計算 */
14:            /* l_{j,i} = 0, (j=0,...,i-1) */
15:            for (j = 0; j < i; ++j) {
16:                L->val[j][i] = 0.0;
17:            }
18:            /* U の計算 */
19:            /* u_{i,j} = 0, (j=0,...,i-1) */
20:            for (j = 0; j < i; ++j) {
21:                U->val[i][j] = 0.0;
22:            }
23:            /* u_{i,i} = 1 */
24:            U->val[i][i] = 1.0;
25:            /* u_{i,j} = a_{i,j}/l_{i,i}, (j=i+1,...,n-1) */
26:            for (j = i + 1; j < n; ++j) {
27:                U->val[i][j] = L->val[i][j] / L->val[i][i];
28:            }
29:            /* A の再計算 */
30:            /* A-lu^T */
31:            for (j = i + 1; j < n; ++j) {
32:                for (k = i + 1; k < n; ++k) {
33:                    L->val[j][k] -= L->val[j][i] * U->val[i][k];
34:                }
35:            }
36:        }
37:        return det;
38:    }
39:
40:    void lu_solve(vector* x, const matrix* L, const matrix* U, const vector* b)
{
41:        /* ここを実装せよ */
42:        int i, j;
43:        int n = b->size;
44:        for (i = 0; i < n; i++) {
45:            x->val[i] = b->val[i];
46:            for (j = 0; j < i; j++) {
47:                x->val[i] -= L->val[i][j] * x->val[j];
48:            }
49:            x->val[i] /= L->val[i][i];
50:        }
51:        for (i = n - 1; i >= 0; i--) {
52:            for (j = i + 1; j < n; j++) {
53:                x->val[i] -= U->val[i][j] * x->val[j];
54:            }
55:        }
56:    }
```

リスト 5　lu.c のソースリスト

```
 1:    #include <stdlib.h>
 2:    #include "vector.h"
 3:    #include "matrix.h"
 4:    #include "lu.h"
 5:
 6:    /**
 7:     *    / 8 16 24  32 ¥
 8:     * A = | 2  7 12  17 |
 9:     *    | 6 17 32  59 |
10:     *    ¥ 7 22 46 105 /
```

```
11:      *
12:      *    / 160 ¥
13:      * b = |  70 |
14:      *    | 198 |
15:      *    ¥ 291 /
16:      * @param[out] A 係数行列
17:      * @param[out] b 右辺ベクトル
18:      */
19:     void set_equation1(matrix* A, vector* b) {
20:        /* 行列サイズ指定 */
21:        int N = 4;
22:        /* 領域確保 */
23:        matrix_new(N, N, A);
24:        vector_new(N, b);
25:        /* 係数行列要素指定 */
26:        A->val[0][0] = 8;
27:        A->val[0][1] = 16;
28:        A->val[0][2] = 24;
29:        A->val[0][3] = 32;
30:        A->val[1][0] = 2;
31:        A->val[1][1] = 7;
32:        A->val[1][2] = 12;
33:        A->val[1][3] = 17;
34:        A->val[2][0] = 6;
35:        A->val[2][1] = 17;
36:        A->val[2][2] = 32;
37:        A->val[2][3] = 59;
38:        A->val[3][0] = 7;
39:        A->val[3][1] = 22;
40:        A->val[3][2] = 46;
41:        A->val[3][3] = 105;
42:        /* 右辺ベクトル要素指定 */
43:        b->val[0] = 160;
44:        b->val[1] = 70;
45:        b->val[2] = 198;
46:        b->val[3] = 291;
47:     }
48:
49:     /**
50:      *    / 3 2 1 ¥
51:      * A = | 2 4 3 |
52:      *    ¥ 1 3 5 /
53:      *
54:      *    /  6   ¥
55:      * b = | 10   |
56:      *    ¥ 12.5 /
57:      * @param[out] A 係数行列
58:      * @param[out] b 右辺ベクトル
59:      */
60:     void set_equation2(matrix* A, vector* b) {
61:        /* 行列サイズ指定 */
62:        int N = 3;
63:        /* 領域確保 */
64:        matrix_new(N, N, A);
65:        vector_new(N, b);
66:        /* 係数行列要素指定 */
67:        A->val[0][0] = 3;
68:        A->val[0][1] = 2;
69:        A->val[0][2] = 1;
70:        A->val[1][0] = 2;
71:        A->val[1][1] = 4;
72:        A->val[1][2] = 3;
73:        A->val[2][0] = 1;
74:        A->val[2][1] = 3;
75:        A->val[2][2] = 5;
76:        /* 右辺ベクトル要素指定 */
```

```
 77:        b->val[0] = 6;
 78:        b->val[1] = 10;
 79:        b->val[2] = 12.5;
 80:    }
 81:
 82:
 83:    /**
 84:     *    / 1 2 2 ¥
 85:     * A = | 3 6 2 |
 86:     *    ¥ 1 1 1 /
 87:     *
 88:     *    / 1 ¥
 89:     * b = | 7 |
 90:     *    ¥ 5 /
 91:     * @param[out] A 係数行列
 92:     * @param[out] b 右辺ベクトル
 93:     */
 94:    void set_equation3(matrix* A, vector* b) {
 95:        /* 行列サイズ指定 */
 96:        int N = 3;
 97:        /* 領域確保 */
 98:        matrix_new(N, N, A);
 99:        vector_new(N, b);
100:        /* 係数行列要素指定 */
101:        A->val[0][0] = 1;
102:        A->val[0][1] = 2;
103:        A->val[0][2] = 2;
104:        A->val[1][0] = 3;
105:        A->val[1][1] = 6;
106:        A->val[1][2] = 2;
107:        A->val[2][0] = 1;
108:        A->val[2][1] = 1;
109:        A->val[2][2] = 1;
110:        /* 右辺ベクトル要素指定 */
111:        b->val[0] = 1;
112:        b->val[1] = 7;
113:        b->val[2] = 5;
114:    }
115:
116:    int main(void) {
117:        int N; /* 行列サイズ */
118:        double det; /* 行列式 */
119:        matrix A; /* 係数行列 */
120:        vector b; /* 右辺ベクトル */
121:        matrix L; /* L */
122:        matrix U; /* U */
123:        vector x; /* 解ベクトル */
124:        char sw[2]; /* 方程式切替スイッチ */
125:
126:        printf("input 1,2,3 ? ");
127:        fgets(sw, sizeof(sw), stdin);
128:        switch (atoi(sw)) {
129:        case 1:
130:            set_equation1(&A, &b);
131:            break;
132:        case 2:
133:            set_equation2(&A, &b);
134:            break;
135:        case 3:
136:            set_equation3(&A, &b);
137:            break;
138:        default:
139:            return EXIT_FAILURE;
140:        }
141:        /* 行列サイズ指定 */
142:        N = A.size1;
```

```
143:        /* 領域確保 */
144:        matrix_new(N, N, &L);
145:        matrix_new(N, N, &U);
146:        vector_new(N, &x);
147:        /* 係数行列表示 */
148:        printf("A =¥n");
149:        matrix_print(&A, "%f");
150:        /* 右辺ベクトル表示 */
151:        printf("b = ");
152:        vector_print(&b, "%f");
153:        /* LU 分解 */
154:        det = lu_decomp(&L, &U, &A);
155:        /* L と U を表示 */
156:        printf("L =¥n");
157:        matrix_print(&L, "%f");
158:        printf("U =¥n");
159:        matrix_print(&U, "%f");
160:        /* 行列式表示 */
161:        printf("det = %3.0f¥n", det);
162:        /* LU 分解を用いて連立方程式を解く */
163:        lu_solve(&x, &L, &U, &b);
164:        /* 解ベクトル表示 */
165:        printf("x = ");
166:        vector_print(&x, "%f");
167:        /* 領域解放 */
168:        matrix_delete(&A);
169:        matrix_delete(&L);
170:        matrix_delete(&U);
171:        vector_delete(&b);
172:        vector_delete(&x);
173:
174:        return EXIT_SUCCESS;
175:    }
```

このプログラムに含まれる構造体・関数をそれぞれ以下の表 1・2 にまとめて示す。

表 1　プログラムに含まれる構造体一覧

ファイル名	構造体	説明
matrix.h	matrix{ int size1; int size2; double **val;}	2 次元配列構造体 size1: 行数 size2: 列数 **val: 配列要素
vector.h	vector{ int size; double *val;}	1 次元配列構造体 size: サイズ *val: 配列要素

表 2　プログラムに含まれる関数一覧

ファイル名	関数	説明
matrix.h	void matrix_new(int m, int n, matrix *mat);	m×n の 2 次元配列を動的に確保 m: 行数 n: 列数 *mat: 行列

	void matrix_delete(matrix *mat);	確保した 2 次元配列を開放 mat: 開放する 2 次元配列
	int matrix_copy(matrix *mat1, const matrix *mat2);	2 次元配列 mat2 を mat1 にコピー mat1: コピー先 mat2: コピー元
	void matrix_print(const matrix *mat, const char *format);	2 次元配列 mat を format に従って標準出力に表示 mat: 出力する 2 次元配列 format: format 出力形式（prinf の第 1 引数）
vector.h	void vector_new(int n, vector* vec);	サイズ n の 1 次元配列を動的に確保 n: サイズ vec: 1 次元配列
	void vector_delete(vector* vec);	確保した 1 次元配列を開放 mat: 開放する 1 次元配列
	int vector_copy(vector* vec1, const vector* vec2);	1 次元配列 mat2 を mat1 にコピー mat1: コピー先 mat2: コピー元
	void vector_print(const vector* vec, const char* format);	1 次元配列 mat を format に従って標準出力に表示 mat: 出力する 1 次元配列 format: format 出力形式（prinf の第 1 引数）
lu.h	double lu_decomp(matrix* L, matrix* U, const matrix* A);	LU 分解（A=LU） L: 下三角行列 U: 三角行列<対角成分は全て 1> A: LU 分解対象行列
	void lu_solve(vector* x, const matrix* L, const matrix* U, const vector* b);	LU 分解を用いて連立方程式を解く x: 解ベクトル L: 三角行列 U: 上三角行列<対角成分は全て 1> b: 右辺ベクトル
lu-test.c	void set_equation1(matrix* A, vector* b);	テストドライバ 1 A: 係数行列 b: 右辺ベクトル
	void set_equation2(matrix* A,	テストドライバ 2

	vector* b);	A: 係数行列 b: 右辺ベクトル
	void set_equation3(matrix* A, vector* b);	テストドライバ 3 A: 係数行列 b: 右辺ベクトル

2.4. 結果と考察

実行結果を以下の図 2 に示す。

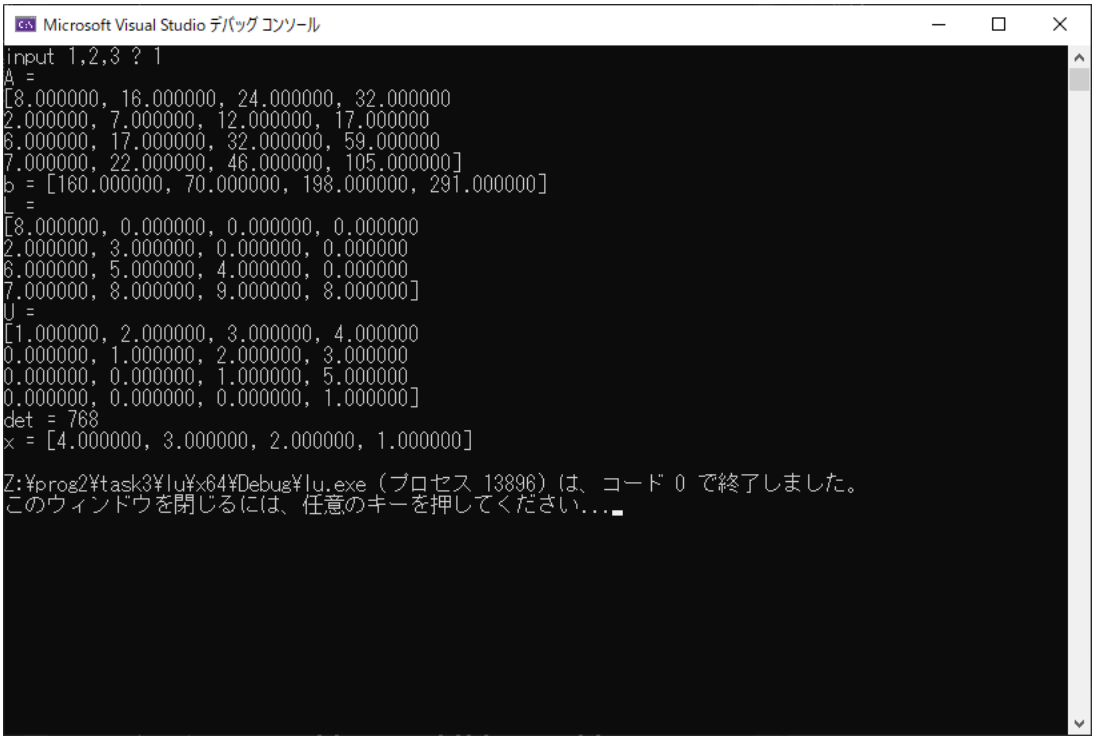

図 2　課題 1 の実行結果

図 1 より解は[4, 3, 2, 1]となっており、これは 3 日目の授業ページの通りであり、正しいことがわかる。

3. 必須課題 2

3.1. 概要

必須課題 1 のプログラムを利用し、スイッチ 2・3 の 2 つについて解の出力結果を求める。これを手計算で解いた場合と比較し、考察する。

3.2. 連立方程式とその解

スイッチ 2（$A = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 3 \\ 1 & 3 & 5 \end{bmatrix}$, $b = \begin{bmatrix} 6 \\ 10 \\ 12.5 \end{bmatrix}$ ）の場合について考える。

手計算を用いて LU 分解を行うと、$L = \begin{bmatrix} 3 & 0 & 0 \\ 2 & \frac{8}{3} & 0 \\ 1 & \frac{7}{3} & \frac{21}{8} \end{bmatrix} \fallingdotseq \begin{bmatrix} 3 & 0 & 0 \\ 2 & 2.666\ldots & 0 \\ 1 & 2.333\ldots & 2.625 \end{bmatrix}$,

$U = \begin{bmatrix} 1 & \frac{2}{3} & \frac{1}{3} \\ 0 & 1 & \frac{7}{8} \\ 0 & 0 & 1 \end{bmatrix} \fallingdotseq \begin{bmatrix} 1 & 0.666\ldots & 0.333\ldots \\ 0 & 1 & 0.875\ldots \\ 0 & 0 & 1 \end{bmatrix}$ となる。$\begin{cases} \boldsymbol{Ly} = \boldsymbol{b} \ \ldots(1) \\ \boldsymbol{Ux} = \boldsymbol{y} \ \ldots(2) \end{cases}$より、(1) を解くと

$$\begin{bmatrix} 3 & 0 & 0 \\ 2 & \frac{8}{3} & 0 \\ 1 & \frac{7}{3} & \frac{21}{8} \end{bmatrix} \cdot \begin{bmatrix} y0 \\ y1 \\ y2 \end{bmatrix} = \begin{bmatrix} 6 \\ 10 \\ 12.5 \end{bmatrix}$$

$$\begin{bmatrix} y0 \\ y1 \\ y2 \end{bmatrix} = \begin{bmatrix} \frac{6}{3} \\ \frac{10 - 2 \cdot y0}{\frac{8}{3}} \\ \frac{12.5 - y0 - \frac{7}{3} \cdot y1}{\frac{21}{8}} \end{bmatrix} = \begin{bmatrix} 2 \\ \frac{9}{4} \\ 2 \end{bmatrix}$$

であり、これを用いて (2) を解くと

$$\begin{bmatrix} 1 & \frac{2}{3} & \frac{1}{3} \\ 0 & 1 & \frac{7}{8} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x0 \\ x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 2 \\ \frac{9}{4} \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} x0 \\ x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 2 - \frac{2}{3} \cdot x1 - \frac{1}{3} \cdot x2 \\ \frac{9}{4} - \frac{7}{8} \cdot x2 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \\ 2 \end{bmatrix}$$

となる。プログラムを実行した結果が以下の図 3 である。

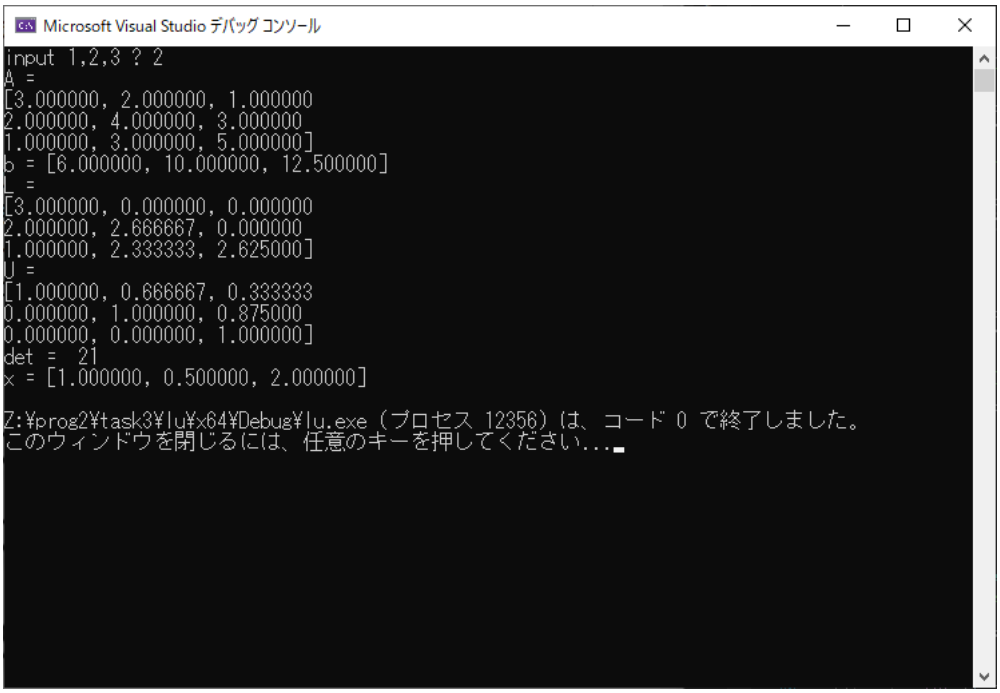

図 3　課題 2 スイッチ 2 についての実行結果

L, U, x について手計算で解いた値と同じ値が表示されている。よってこのプログラムは正常に動作していると考えられる。

スイッチ 3（$A = \begin{bmatrix} 1 & 2 & 2 \\ 3 & 6 & 2 \\ 1 & 1 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 7 \\ 5 \end{bmatrix}$ ）の場合について考える。

手計算を用いて LU 分解を行うと、$L = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 0 & 0 \\ 1 & -1 & \blacksquare \end{bmatrix}$, $U = \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & \blacksquare \\ 0 & 0 & 1 \end{bmatrix}$ となる。■ で表されているのは 0 での除算を含む部分であり、この解法では答えを導けない。このため、解を求めることはできないと考えられる。プログラムを実行した結果を以下の図 4 に示す。

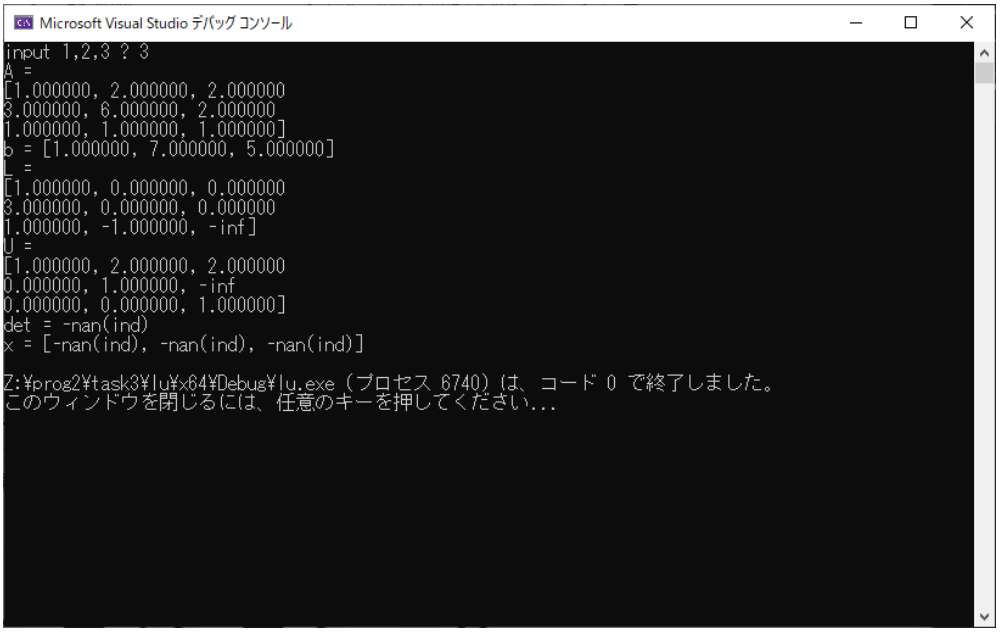

図 4　課題 2 スイッチ 3 についての実行結果

これより、手計算で解いた場合と同じく L,U に-inf という文字が表示されている。このため解を求めることができず、x の値も-nan(ind)となっている。

3.3. 考察

スイッチ 2 の場合については手計算の場合と L, U, x について同じ値が出力された。スイッチ 3 の場合については手計算の時点で 0 での除算が含まれていたため最後まで計算できなかったが、プログラムを実行した場合についても-inf, -nan(ind)という文字が表示され、計算結果を見ることができなかった。0 での除算はできないため、この様な実行結果になったと考えられる。しかし、計算サイト[1]で計算してみたところ、以下の図 5 のように解を求めることができていた。計算方法を変えれば結果を求められると考えられる。

$$Ax = LUx = b$$
$$x = A^{-1}b = U^{-1}L^{-1}b$$

図 5　計算サイトによるスイッチ 3 の実行結果

4. オプション課題 1

未着手

5. オプション課題 2

未着手

参考文献

[1] 連立方程式 - 高精度計算サイト - Keisan,
https://keisan.casio.jp/exec/system/1278931746, 2023/06/29 閲覧