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

# 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。

# 必須課題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。

## 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 の場合についての例を示す。

A picture containing text, diagram, plan, schematic

Description automatically generated

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

リスト2　vector.cのソースリスト

|  |
| --- |
| 1. #include "vector.h" 2. void vector\_new(int n, vector\* vec) { 3. if (n <= 0) { 4. fprintf(stderr, "allocation failure in vector\_new()\n"); 5. exit(EXIT\_FAILURE); 6. } 7. else { 8. vec->val = (double\*)malloc(n \* sizeof(double)); 9. if (vec->val == NULL) { 10. fprintf(stderr, "allocation failure in vector\_new()\n"); 11. exit(EXIT\_FAILURE); 12. } 13. vec->size = n; 14. } 15. } 16. void vector\_delete(vector\* vec) { 17. free(vec->val); 18. } 19. int vector\_copy(vector\* vec1, const vector\* vec2) { 20. int i; 21. if (vec1->size != vec2->size) { 22. return 0; 23. } 24. for (i = 0; i < vec1->size; ++i) { 25. vec1->val[i] = vec2->val[i]; 26. } 27. return 1; 28. } 29. void vector\_print(const vector\* vec, const char\* format) { 30. int i; 31. printf("["); 32. for (i = 0; i < vec->size; ++i) { 33. printf(format, vec->val[i]); 34. if (i < vec->size - 1) { 35. printf(", "); 36. } 37. } 38. printf("]\n"); 39. } |

リスト3　matrix.hのソースリスト

|  |
| --- |
| 1. #ifndef MATRIX\_H 2. #define MATRIX\_H 3. #include <stdio.h> 4. #include <stdlib.h> 5. /\*\* 6. \* @struct matrix 7. \* @brief 2次元配列構造体 8. \*/ 9. typedef struct { 10. int size1; /\*\*< 行数 \*/ 11. int size2; /\*\*< 列数 \*/ 12. double\*\* val; /\*\*< 配列要素 \*/ 13. } matrix; 14. /\*\* 15. \* @brief m×nの2次元配列を動的に確保 16. \* @param[in] m 行数 17. \* @param[in] n 列数 18. \* @param[out] mat 2次元配列 19. \*/ 20. void matrix\_new(int m, int n, matrix\* mat); 21. /\*\* 22. \* @brief 確保した2次元配列を開放 23. \* @param[in,out] mat 開放する2次元配列 24. \*/ 25. void matrix\_delete(matrix\* mat); 26. /\*\* 27. \* @brief 2次元配列mat2をmat1にコピー 28. \* @param[out] mat1 コピー先 29. \* @param[in] mat2 コピー元 30. \* @retval 1 成功 31. \* @retval 0 失敗（mat1とmat2のサイズ不整合） 32. \*/ 33. int matrix\_copy(matrix\* mat1, const matrix\* mat2); 34. /\*\* 35. \* @brief 2次元配列matをformatに従って標準出力に表示 36. \* @param[in] mat 出力する2次元配列 37. \* @param[in] format出力形式（prinfの第1引数） 38. \*/ 39. void matrix\_print(const matrix\* mat, const char\* format); 40. #endif /\* MATRIX\_H \*/ |

リスト4　matrix.cのソースリスト

|  |
| --- |
| 1. #include "matrix.h" 2. void matrix\_new(int m, int n, matrix\* mat) { 3. int i; 4. if ((m <= 0) || (n <= 0)) { 5. fprintf(stderr, "allocation failure in matrix\_new()\n"); 6. exit(EXIT\_FAILURE); 7. } 8. mat->val = (double\*\*)malloc(m \* sizeof(double\*)); 9. if (mat->val == NULL) { /\* 領域割当に失敗したら \*/ 10. fprintf(stderr, "allocation failure in matrix\_new()\n"); 11. exit(EXIT\_FAILURE); 12. } 13. for (i = 0; i < m; ++i) { 14. mat->val[i] = (double\*)malloc(n \* sizeof(double)); 15. if (mat->val[i] == NULL) { /\* 領域割当に失敗したら \*/ 16. while (--i >= 0) { 17. free(mat->val[i]); /\* 領域開放 \*/ 18. } 19. free(mat->val); 20. fprintf(stderr, "allocation failure in matrix\_new()\n"); 21. exit(EXIT\_FAILURE); 22. } 23. } 24. mat->size1 = m; 25. mat->size2 = n; 26. } 27. void matrix\_delete(matrix\* mat) { 28. int i; 29. for (i = 0; i < mat->size1; ++i) { 30. free(mat->val[i]); 31. } 32. free(mat->val); 33. mat->size1 = mat->size2 = 0; 34. } 35. int matrix\_copy(matrix\* mat1, const matrix\* mat2) { 36. int i, j; 37. if ((mat1->size1 != mat2->size1) || (mat2->size2 != mat2->size2)) { 38. return 0; 39. } 40. for (i = 0; i < mat1->size1; ++i) { 41. for (j = 0; j < mat1->size2; ++j) { 42. mat1->val[i][j] = mat2->val[i][j]; 43. } 44. } 45. return 1; 46. } 47. void matrix\_print(const matrix\* mat, const char\* format) { 48. int i, j; 49. printf("["); 50. for (i = 0; i < mat->size1; ++i) { 51. for (j = 0; j < mat->size2; ++j) { 52. printf(format, mat->val[i][j]); 53. if (j < mat->size2 - 1) { 54. printf(", "); 55. } 56. } 57. if (i == mat->size1 - 1) { 58. printf("]"); 59. } 60. printf("\n"); 61. } 62. } |

リスト5　lu.hのソースリスト

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

リスト5　lu.cのソースリスト

|  |
| --- |
| 1. #include "lu.h" 2. double lu\_decomp(matrix\* L, matrix\* U, const matrix\* A) { 3. int i, j, k; 4. int n = A->size1; 5. double det; 6. /\* AをLにコピー \*/ 7. matrix\_copy(L, A); 8. det = 1.0; 9. for (i = 0; i < n; ++i) { 10. det \*= L->val[i][i]; 11. /\* Lの計算 \*/ 12. /\* l\_{j,i} = 0, (j=0,...,i-1) \*/ 13. for (j = 0; j < i; ++j) { 14. L->val[j][i] = 0.0; 15. } 16. /\* Uの計算 \*/ 17. /\* u\_{i,j} = 0, (j=0,...,i-1) \*/ 18. for (j = 0; j < i; ++j) { 19. U->val[i][j] = 0.0; 20. } 21. /\* u\_{i,i} = 1 \*/ 22. U->val[i][i] = 1.0; 23. /\* u\_{i,j} = a\_{i,j}/l\_{i,i}, (j=i+1,...,n-1) \*/ 24. for (j = i + 1; j < n; ++j) { 25. U->val[i][j] = L->val[i][j] / L->val[i][i]; 26. } 27. /\* Aの再計算 \*/ 28. /\* A-lu^T \*/ 29. for (j = i + 1; j < n; ++j) { 30. for (k = i + 1; k < n; ++k) { 31. L->val[j][k] -= L->val[j][i] \* U->val[i][k]; 32. } 33. } 34. } 35. return det; 36. } 37. void lu\_solve(vector\* x, const matrix\* L, const matrix\* U, const vector\* b) { 38. /\* ここを実装せよ \*/ 39. int i, j; 40. int n = b->size; 41. for (i = 0; i < n; i++) { 42. x->val[i] = b->val[i]; 43. for (j = 0; j < i; j++) { 44. x->val[i] -= L->val[i][j] \* x->val[j]; 45. } 46. x->val[i] /= L->val[i][i]; 47. } 48. for (i = n - 1; i >= 0; i--) { 49. for (j = i + 1; j < n; j++) { 50. x->val[i] -= U->val[i][j] \* x->val[j]; 51. } 52. } 53. } |

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. \* / 8 16 24 32 \ 7. \* A = | 2 7 12 17 | 8. \* | 6 17 32 59 | 9. \* \ 7 22 46 105 / 10. \* 11. \* / 160 \ 12. \* b = | 70 | 13. \* | 198 | 14. \* \ 291 / 15. \* @param[out] A 係数行列 16. \* @param[out] b 右辺ベクトル 17. \*/ 18. void set\_equation1(matrix\* A, vector\* b) { 19. /\* 行列サイズ指定 \*/ 20. int N = 4; 21. /\* 領域確保 \*/ 22. matrix\_new(N, N, A); 23. vector\_new(N, b); 24. /\* 係数行列要素指定 \*/ 25. A->val[0][0] = 8; 26. A->val[0][1] = 16; 27. A->val[0][2] = 24; 28. A->val[0][3] = 32; 29. A->val[1][0] = 2; 30. A->val[1][1] = 7; 31. A->val[1][2] = 12; 32. A->val[1][3] = 17; 33. A->val[2][0] = 6; 34. A->val[2][1] = 17; 35. A->val[2][2] = 32; 36. A->val[2][3] = 59; 37. A->val[3][0] = 7; 38. A->val[3][1] = 22; 39. A->val[3][2] = 46; 40. A->val[3][3] = 105; 41. /\* 右辺ベクトル要素指定 \*/ 42. b->val[0] = 160; 43. b->val[1] = 70; 44. b->val[2] = 198; 45. b->val[3] = 291; 46. } 47. /\*\* 48. \* / 3 2 1 \ 49. \* A = | 2 4 3 | 50. \* \ 1 3 5 / 51. \* 52. \* / 6 \ 53. \* b = | 10 | 54. \* \ 12.5 / 55. \* @param[out] A 係数行列 56. \* @param[out] b 右辺ベクトル 57. \*/ 58. void set\_equation2(matrix\* A, vector\* b) { 59. /\* 行列サイズ指定 \*/ 60. int N = 3; 61. /\* 領域確保 \*/ 62. matrix\_new(N, N, A); 63. vector\_new(N, b); 64. /\* 係数行列要素指定 \*/ 65. A->val[0][0] = 3; 66. A->val[0][1] = 2; 67. A->val[0][2] = 1; 68. A->val[1][0] = 2; 69. A->val[1][1] = 4; 70. A->val[1][2] = 3; 71. A->val[2][0] = 1; 72. A->val[2][1] = 3; 73. A->val[2][2] = 5; 74. /\* 右辺ベクトル要素指定 \*/ 75. b->val[0] = 6; 76. b->val[1] = 10; 77. b->val[2] = 12.5; 78. } 79. /\*\* 80. \* / 1 2 2 \ 81. \* A = | 3 6 2 | 82. \* \ 1 1 1 / 83. \* 84. \* / 1 \ 85. \* b = | 7 | 86. \* \ 5 / 87. \* @param[out] A 係数行列 88. \* @param[out] b 右辺ベクトル 89. \*/ 90. void set\_equation3(matrix\* A, vector\* b) { 91. /\* 行列サイズ指定 \*/ 92. int N = 3; 93. /\* 領域確保 \*/ 94. matrix\_new(N, N, A); 95. vector\_new(N, b); 96. /\* 係数行列要素指定 \*/ 97. A->val[0][0] = 1; 98. A->val[0][1] = 2; 99. A->val[0][2] = 2; 100. A->val[1][0] = 3; 101. A->val[1][1] = 6; 102. A->val[1][2] = 2; 103. A->val[2][0] = 1; 104. A->val[2][1] = 1; 105. A->val[2][2] = 1; 106. /\* 右辺ベクトル要素指定 \*/ 107. b->val[0] = 1; 108. b->val[1] = 7; 109. b->val[2] = 5; 110. } 111. int main(void) { 112. int N; /\* 行列サイズ \*/ 113. double det; /\* 行列式 \*/ 114. matrix A; /\* 係数行列 \*/ 115. vector b; /\* 右辺ベクトル \*/ 116. matrix L; /\* L \*/ 117. matrix U; /\* U \*/ 118. vector x; /\* 解ベクトル \*/ 119. char sw[2]; /\* 方程式切替スイッチ \*/ 120. printf("input 1,2,3 ? "); 121. fgets(sw, sizeof(sw), stdin); 122. switch (atoi(sw)) { 123. case 1: 124. set\_equation1(&A, &b); 125. break; 126. case 2: 127. set\_equation2(&A, &b); 128. break; 129. case 3: 130. set\_equation3(&A, &b); 131. break; 132. default: 133. return EXIT\_FAILURE; 134. } 135. /\* 行列サイズ指定 \*/ 136. N = A.size1; 137. /\* 領域確保 \*/ 138. matrix\_new(N, N, &L); 139. matrix\_new(N, N, &U); 140. vector\_new(N, &x); 141. /\* 係数行列表示 \*/ 142. printf("A =\n"); 143. matrix\_print(&A, "%f"); 144. /\* 右辺ベクトル表示 \*/ 145. printf("b = "); 146. vector\_print(&b, "%f"); 147. /\* LU分解 \*/ 148. det = lu\_decomp(&L, &U, &A); 149. /\* LとUを表示 \*/ 150. printf("L =\n"); 151. matrix\_print(&L, "%f"); 152. printf("U =\n"); 153. matrix\_print(&U, "%f"); 154. /\* 行列式表示 \*/ 155. printf("det = %3.0f\n", det); 156. /\* LU分解を用いて連立方程式を解く \*/ 157. lu\_solve(&x, &L, &U, &b); 158. /\* 解ベクトル表示 \*/ 159. printf("x = "); 160. vector\_print(&x, "%f"); 161. /\* 領域解放 \*/ 162. matrix\_delete(&A); 163. matrix\_delete(&L); 164. matrix\_delete(&U); 165. vector\_delete(&b); 166. vector\_delete(&x); 167. return EXIT\_SUCCESS; 168. } |

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, vector\* b); | テストドライバ2  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 |

## 結果と考察

実行結果を以下の図2に示す。

A screenshot of a computer

Description automatically generated  
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。

# 必須課題２

## 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。

## 連立方程式とその解

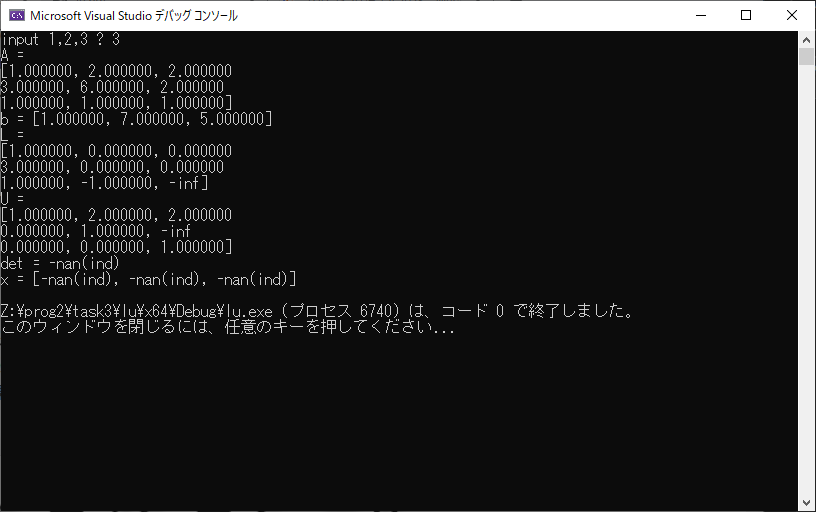
スイッチ2（ , ）の場合について考える。  
手計算を用いてLU分解を行うと、 ,   
 となる。より、（1）を解くと  
であり、これを用いて（2）を解くと  
となる。プログラムを実行した結果が以下の図3である。

A picture containing text, screenshot, computer, software

Description automatically generated  
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（ , ）の場合について考える。  
手計算を用いてLU分解を行うと、 , 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。

## 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。

A screenshot of a computer

Description automatically generated with medium confidence  
図5　計算サイトによるスイッチ3の実行結果

# オプション課題１

未着手

# 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