Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа программной инженерии

**Отчёт по лабораторной работе №2**

1. по дисциплине «Вычислительная математика»
2. Выполнил студент гр. 3530904/90002 Мэн Цзянин

Преподаватель

С.П.Воскобойников

1. Санкт-Петербург
2. 2021

Оглавление

[Постановка задачи 3](#_Toc65077170)

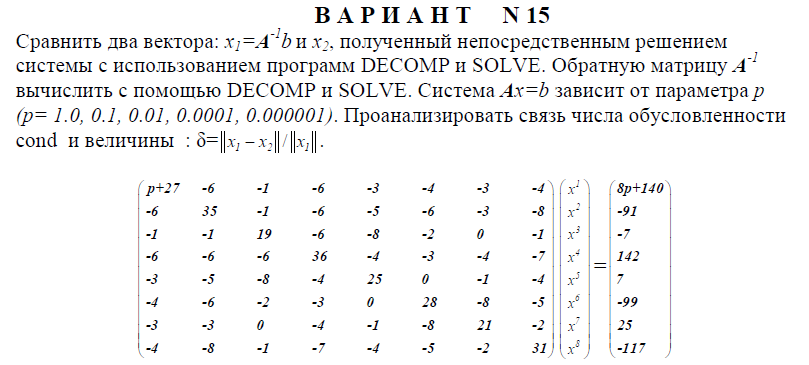
[Ход работы 3](#_Toc65077171)

[Результат работы программы 4](#_Toc65077172)

[Код программы 10](#_Toc65077173)

[Вывод 18](#_Toc65077174)

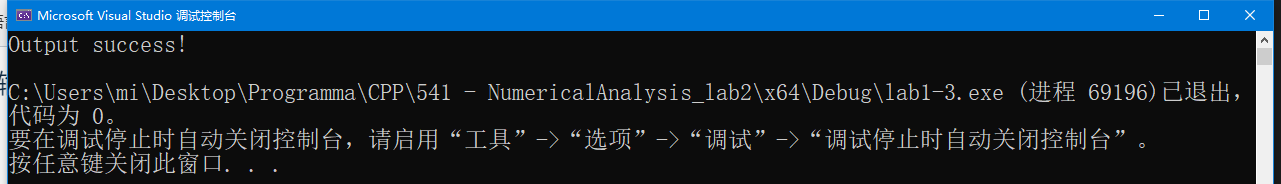
# Постановка задачи



# Ход работы

1. Построены матрицы A и векторы b в зависимости от изменения параметра p.
2. Найдена обратная матрийа A^-1 с помощью программы DECOMP и SOLVE, получино число обратной матрицы cond(A^-1)
3. Получино решение x1 = A^-1\*b
4. Для матрицы P(P==A) найдено её LU разложение с помощью программы DECOMP, а затем необходимое число раз (а именно N = 8) решены системы с помощью программы SOLVE. Получино решение x2
5. Вычислено вектор невязки |x1-x2|
6. Вычислена величина 
7. Получено число обусловленности
8. Исследована связь числа обусловленности и delta.

# Результат работы программы



result-lab2-table.xlsx

связь числа обусловленности cond и величины delta=||x1-x2||/||x1||

талица 1

|  |  |  |
| --- | --- | --- |
| **p** | **cond(A^-1)** | **delta** |
| 1 | 5.93E+02 | 1.75E-14 |
| 0.1 | 5777 | 1.22E-13 |
| 0.01 | 5.76E+04 | 1.27E-12 |
| 0.0001 | 5.76E+06 | 1.47E-10 |
| 0.000001 | 5.76E+08 | 1.14E-08 |











# Код программы

1. #include <math.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <fstream>
5. #include <iostream>
6. #include "cmath.h"
7. #include <iomanip>
8. #include <vector>
9. const int DIM\_8 = 8;
10. double cond;
11. static int flag = -1;
12. int pivot[DIM\_8];
13. std::ofstream out("lab2-result.txt");
14. double\* minus(double\* x1, double\* x2)
15. {
16. double\* result = new double[DIM\_8];
17. for (int j = 0; j < 8; j++)
18. {
19. result[j] = fabs(x1[j] - x2[j]);
20. }
21. return result;
22. }
23. double get\_relative\_norm(double\* x1, double\* x2)
24. {
25. double minus\_norm = 0;
26. double norm\_x1 = 0;
27. for (int j = 0; j < 8; j++)
28. {
29. minus\_norm += fabs(x1[j] - x2[j]);
30. norm\_x1 += fabs(x1[j]);
31. }
32. return (minus\_norm / norm\_x1);
33. }
34. double\* set\_matrix\_A(double p)
35. {
36. double\* retP = new double[64];
37. retP[0] = p + 27; retP[1] = -6;  retP[2] = -1;  retP[3] = -6;  retP[4] = -3;  retP[5] = -4;  retP[6] = -3;  retP[7] = -4;
38. retP[8] = -6;     retP[9] = 35;  retP[10] = -1; retP[11] = -6; retP[12] = -5; retP[13] = -6; retP[14] = -3; retP[15] = -8;
39. retP[16] = -1;    retP[17] = -1; retP[18] = 19; retP[19] = -6; retP[20] = -8; retP[21] = -2; retP[22] = 0;  retP[23] = -1;
40. retP[24] = -6;    retP[25] = -6; retP[26] = -6; retP[27] = 36; retP[28] = -4; retP[29] = -3; retP[30] = -4; retP[31] = -7;
41. retP[32] = -3;    retP[33] = -5; retP[34] = -8; retP[35] = -4; retP[36] = 25; retP[37] = 0;  retP[38] = -1; retP[39] = -4;
42. retP[40] = -4;    retP[41] = -6; retP[42] = -2; retP[43] = -3; retP[44] = 0;  retP[45] = 28; retP[46] = -8; retP[47] = -5;
43. retP[48] = -3;    retP[49] = -3; retP[50] = 0;  retP[51] = -4; retP[52] = -1; retP[53] = -8; retP[54] = 21; retP[55] = -2;
44. retP[56] = -4;    retP[57] = -8; retP[58] = -1; retP[59] = -7; retP[60] = -4; retP[61] = -5; retP[62] = -2; retP[63] = 31;
45. return retP;
46. }
47. double\* set\_vector\_b(double p)
48. {
49. double\* ret\_b = new double[DIM\_8];
50. ret\_b[0] = 8 \* p + 140;
51. ret\_b[1] = -91;
52. ret\_b[2] = -7;
53. ret\_b[3] = 142;
54. ret\_b[4] = 7;
55. ret\_b[5] = -99;
56. ret\_b[6] = 25;
57. ret\_b[7] = -117;
58. return ret\_b;
59. }
60. void show\_matrix(double\* arr)
61. {
62. for (int i = 0; i < 8; i++) {
63. for (int j = 0; j < 8; j++)
64. {
65. out << std::scientific << std::setprecision(3) << setiosflags(std::ios::left);
66. *//out << std::setw(15) << arr[i \* 8 + j];*
67. out << arr[i \* 8 + j] << '\t';
68. }
69. out << std::endl;
70. }
71. out << std::endl;
72. }
73. void show\_2d\_array(double\*\* arr)
74. {
75. out << " before decomp, P1=\n";
76. for (int i = 0; i < 8; i++) {
77. for (int j = 0; j < 8; j++)
78. {
79. out << std::scientific << std::setprecision(3) << setiosflags(std::ios::left);
80. out << std::setw(15) << arr[i][j];
81. }
82. out << std::endl;
83. }
84. out << std::endl;
85. }
86. void show\_vector(double\* v)
87. {
88. for (int i = 0; i < 8; i++)
89. {
90. out << std::scientific << std::setprecision(3) << setiosflags(std::ios::left);
91. *// out << std::setw(15) << v[i];*
92. out << v[i] << '\n';
93. }
94. out << std::endl;
95. }
96. double\* inv(double\* arr)
97. {
98. static int flag;
99. int pivot[DIM\_8];
100. double\* a\_inv = new double[64];
101. double b[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };
102. decomp(DIM\_8, DIM\_8, arr, &cond, pivot, &flag);
103. if (flag != 0)
104. {
105. out << "Fail to decompose!! flag =" << flag << "\n";
106. return NULL;
107. }
108. for (int i = 0; i < DIM\_8; i++)
109. {
110. b[i] = 1;
111. solve(8, 8, arr, b, pivot);
112. for (int j = 0; j < DIM\_8; j++)
113. {
114. a\_inv[j \* 8 + i] = b[j];
115. b[j] = 0;
116. }
117. }
118. out << "A^-1:\n";
119. show\_matrix(a\_inv);
120. out << "cond(A^-1)=" << cond << "\n\n";
121. return a\_inv;
122. }
123. *// matrix multiply by vector*
124. double\* multiply\_by\_vector(double\* matrix, double\* vec)
125. {
126. double\* result\_vector = new double[DIM\_8];
127. double temp\_element = 0;
128. for (int i = 0; i < DIM\_8; ++i)
129. {
130. temp\_element = 0;
131. for (int j = 0; j < DIM\_8; ++j)
132. {
133. temp\_element += matrix[i \* DIM\_8 + j] \* vec[j];
134. }
135. result\_vector[i] = temp\_element;
136. }
137. return result\_vector;
138. }
139. double\* decomp\_solve\_2d(double matrix\_A[][8], double\* vec\_b)
140. {
141. decomp(DIM\_8, DIM\_8, \*matrix\_A, &cond, pivot, &flag);
142. if (flag != 0)
143. {
144. out << "fail to decompose! flag=" << flag << '\n';
145. }
146. solve(8, 8, \*matrix\_A, vec\_b, pivot);
147. out << "x2(solution with decomp and solve)=\n";
148. show\_vector(vec\_b);
149. return vec\_b;
150. }
151. void lab2()
152. {
153. using std::endl;
154. double cond = 0;
155. double p1 = 1.0;
156. double p2 = 0.1;
157. double p3 = 0.01;
158. double p4 = 0.0001;
159. double p5 = 0.000001;
160. double\* b1 = set\_vector\_b(p1);
161. double\* b2 = set\_vector\_b(p2);
162. double\* b3 = set\_vector\_b(p3);
163. double\* b4 = set\_vector\_b(p4);
164. double\* b5 = set\_vector\_b(p5);
165. double\* A1 = set\_matrix\_A(p1);
166. double\* A2 = set\_matrix\_A(p2);
167. double\* A3 = set\_matrix\_A(p3);
168. double\* A4 = set\_matrix\_A(p4);
169. double\* A5 = set\_matrix\_A(p5);
170. double P1[DIM\_8][DIM\_8] = {
171. {p1 + 27, -6, -1, -6, -3,  -4, -3, -4},
172. {-6,   35, -1, -6, -5,  -6, -3, -8,},
173. {-1,   -1, 19, -6, -8,  -2,  0, -1,},
174. {-6,   -6, -6, 36, -4,  -3, -4, -7,},
175. {-3,   -5, -8, -4, 25,   0, -1, -4,},
176. {-4,   -6, -2, -3,  0,  28,  -8, -5,},
177. {-3,   -3,  0, -4, -1,  -8, 21, -2,},
178. {-4,   -8, -1, -7, -4,  -5, -2, 31}
179. };
180. double P2[DIM\_8][DIM\_8] = {
181. {p2 + 27, -6, -1, -6, -3,  -4, -3, -4},
182. {-6,   35, -1, -6, -5,  -6, -3, -8,},
183. {-1,   -1, 19, -6, -8,  -2,  0, -1,},
184. {-6,   -6, -6, 36, -4,  -3, -4, -7,},
185. {-3,   -5, -8, -4, 25,   0, -1, -4,},
186. {-4,   -6, -2, -3,  0,  28,  -8, -5,},
187. {-3,   -3,  0, -4, -1,  -8, 21, -2,},
188. {-4,   -8, -1, -7, -4,  -5, -2, 31}
189. };
190. double P3[DIM\_8][DIM\_8] = {
191. {p3 + 27, -6, -1, -6, -3,  -4, -3, -4},
192. {-6,   35, -1, -6, -5,  -6, -3, -8,},
193. {-1,   -1, 19, -6, -8,  -2,  0, -1,},
194. {-6,   -6, -6, 36, -4,  -3, -4, -7,},
195. {-3,   -5, -8, -4, 25,   0, -1, -4,},
196. {-4,   -6, -2, -3,  0,  28,  -8, -5,},
197. {-3,   -3,  0, -4, -1,  -8, 21, -2,},
198. {-4,   -8, -1, -7, -4,  -5, -2, 31}
199. };
200. double P4[DIM\_8][DIM\_8] = {
201. {p4 + 27, -6, -1, -6, -3,  -4, -3, -4},
202. {-6,   35, -1, -6, -5,  -6, -3, -8,},
203. {-1,   -1, 19, -6, -8,  -2,  0, -1,},
204. {-6,   -6, -6, 36, -4,  -3, -4, -7,},
205. {-3,   -5, -8, -4, 25,   0, -1, -4,},
206. {-4,   -6, -2, -3,  0,  28,  -8, -5,},
207. {-3,   -3,  0, -4, -1,  -8, 21, -2,},
208. {-4,   -8, -1, -7, -4,  -5, -2, 31}
209. };
210. double P5[DIM\_8][DIM\_8] = {
211. {p5 + 27, -6, -1, -6, -3,  -4, -3, -4},
212. {-6,   35, -1, -6, -5,  -6, -3, -8,},
213. {-1,   -1, 19, -6, -8,  -2,  0, -1,},
214. {-6,   -6, -6, 36, -4,  -3, -4, -7,},
215. {-3,   -5, -8, -4, 25,   0, -1, -4,},
216. {-4,   -6, -2, -3,  0,  28,  -8, -5,},
217. {-3,   -3,  0, -4, -1,  -8, 21, -2,},
218. {-4,   -8, -1, -7, -4,  -5, -2, 31}
219. };
220. *// output result*
221. out << "---------------------------- A1 \* x = b1 ---------------------------------\n";
222. out << "p1=" << p1 << "\n\n";
223. out << "A1:\n";
224. show\_matrix(A1);
225. double\* invA1 = inv(A1);
226. double\* x1\_1 = multiply\_by\_vector(invA1, b1);
227. out << "x1(invA1\*b1)=\n";
228. show\_vector(x1\_1);
229. double\* x2\_1 = decomp\_solve\_2d(P1, b1);
230. out << "|x1-x2|=\n";
231. show\_vector(minus(x1\_1, x2\_1));
232. out << "||x2-x1||/||x1||=" << get\_relative\_norm(x1\_1, x2\_1) << '\n';
233. out << "---------------------------- A2 \* x = b2 ---------------------------------\n";
234. out << "p2=" << p2 << "\n\n";
235. out << "A2:\n";
236. show\_matrix(A2);
237. double\* invA2 = inv(A2);
238. double\* x1\_2 = multiply\_by\_vector(invA2, b2);
239. out << "x1(invA2\*b2)=\n";
240. show\_vector(x1\_2);
241. double\* x2\_2 = decomp\_solve\_2d(P2, b2);
242. out << "|x1-x2|=\n";
243. show\_vector(minus(x1\_2, x2\_2));
244. out << "||x2-x1||/||x1||=" << get\_relative\_norm(x1\_2, x2\_2) << '\n';
245. out << "---------------------------- A3 \* x = b3 ---------------------------------\n";
246. out << "p3=" << p3 << "\n\n";
247. out << "A3:\n";
248. show\_matrix(A3);
249. double\* invA3 = inv(A3);
250. double\* x1\_3 = multiply\_by\_vector(invA3, b3);
251. out << "x1(invA3\*b3)=\n";
252. show\_vector(x1\_3);
253. double\* x2\_3 = decomp\_solve\_2d(P3, b3);
254. out << "|x1-x2|=\n";
255. show\_vector(minus(x1\_3, x2\_3));
256. out << "||x2-x1||/||x1||=" << get\_relative\_norm(x1\_3, x2\_3) << '\n';
257. out << "---------------------------- A4 \* x = b4 ---------------------------------\n";
258. out << "p4=" << p4 << "\n\n";
259. out << "A4:\n";
260. show\_matrix(A4);
261. double\* invA4 = inv(A4);
262. double\* x1\_4 = multiply\_by\_vector(invA4, b4);
263. out << "x1(invA4\*b4)=\n";
264. show\_vector(x1\_4);
265. double\* x2\_4 = decomp\_solve\_2d(P4, b4);
266. out << "|x1-x2|=\n";
267. show\_vector(minus(x1\_4, x2\_4));
268. out << "||x2-x1||/||x1||=" << get\_relative\_norm(x1\_4, x2\_4) << '\n';
269. out << "---------------------------- A5 \* x = b5 ---------------------------------\n";
270. out << "p5=" << p5 << "\n\n";
271. out << "A5:\n";
272. show\_matrix(A5);
273. double\* invA5 = inv(A5);
274. double\* x1\_5 = multiply\_by\_vector(invA5, b5);
275. out << "x1(invA5\*b5)=\n";
276. show\_vector(x1\_5);
277. double\* x2\_5 = decomp\_solve\_2d(P5, b5);
278. out << "|x1-x2|=\n";
279. show\_vector(minus(x1\_5, x2\_5));
280. out << "||x2-x1||/||x1||=" << get\_relative\_norm(x1\_5, x2\_5) << '\n';
281. }

# Вывод

Исходя из таблицы 1, можно сказать, что при изменении параметра p в меньшую сторону увеличивается и число обусловленности(cond), и величины delta=||x1-x2||/||x1||.