Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

до виконаної лабораторної роботи № 1

з дисципліни

“Чисельні методи”

на тему:

***«Абсолютна та відносна похибка»***

Виконав

студент групи *ОІ-11 сп*

*Вальчевський П. В.*

Викладач:

*Сенета М. Я.*

Львів – 2023

## Лабораторна робота № 1

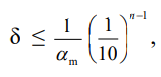
*Тема роботи:*  **«Абсолютна та відносна похибка».**

*Мета роботи:*вивчити поняття абсолютної та відносної похибки та методи їх оцінювання.

***Порядок виконання роботи***

1. **Основні теоретичні відомості**

*Теорема*. Якщо додатне наближене число а має п точних десяткових знаків, то відносна похибка δ цього числа задовольняє умову:



де ат – перша значуща цифра числа а.

*Похибкa суми*. Абсолютна похибка алгебраїчної суми декількох наближених чисел не перевищує суми абсолютних похибок цих чисел. Доведення. Нехай x1, x2, …, хп – задані наближені числа. Розглянемо їх алгебраїчну суму:



Тоді похибка цієї алгебраїчної суми Дм буде складатися з алгебраїчної суми похибок доданків, тобто



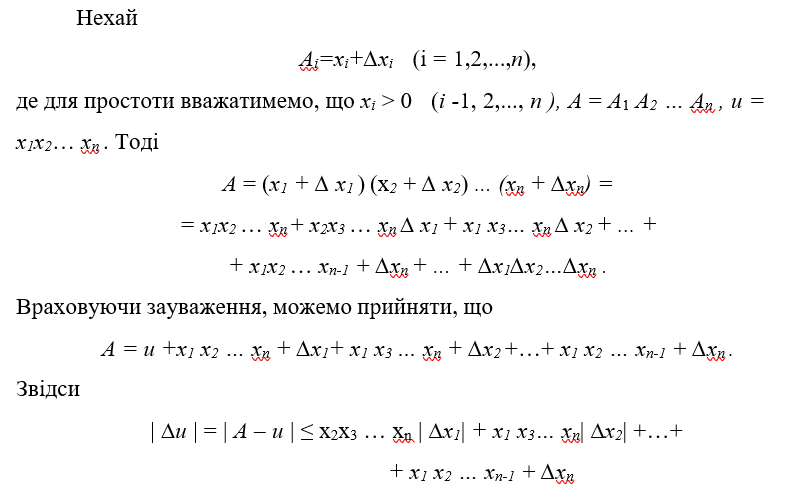
*Похибкa різниці*. Розглянемо різницю двох наближених чисел х1 та х2: и = х1-х2 . Тоді, на підставі наслідку з похибки суми

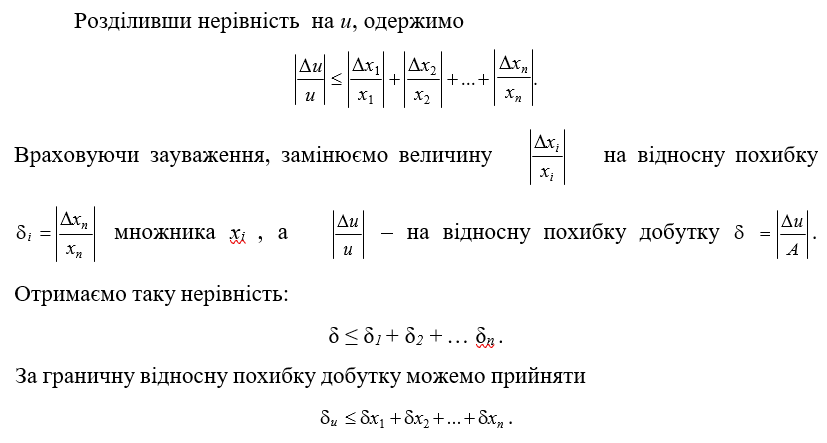


де А – точне значення різниці х1-х2. 3 останньої формули випливає, що для близьких чисел х1 та х2 гранична відносна похибка буде досить велика. Тому в обчислювальних алгоритмах бажано уникати віднімання близьких чисел.

Зауваження. При подальшому розгляді похибок арифметичних операцій, а також при розгляді похибок функцій припускатимемо, що похибки значно менші за абсолютною величиною від самих наближених величин, тож ними можна знехтувати в сумах, котрі містять одночасно наближену величину і її похибку як доданки; і завжди можна обмежитися членами, лінійними відносно похибок, нехтуючи членами більш високого порядку. Це означає, що наступні питання, пов'язані з похибками, розглядатимемо дещо грубо, проте елементарно. Адже строгий підхід під час розгляду цих питань не дає бажаних наочних результатів.

Похибкaдобутку.





Похибка степеня. Нехай *А* = *(х +* ∆ *х)т* , *и* = *хт ,* де *т* – натуральне число, *х* > 0.

Використовуючи похибки добутку, одержуємо

|∆*u*| < *mxm - 1*|∆*x*|, δ ≤ mδ1,

де δ – відносна похибка степеня; δ1 – відносна похибка аргументу *х*. Тому за граничні абсолютну та відносну похибки степеня можемо прийняти

∆*u= mxm - 1*∆*x,* δ*u=* mδ*x* .

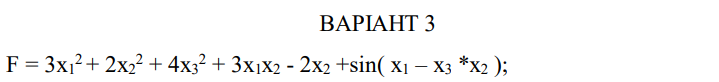
Із наведених похибок арифметичних операцій випливає, що операції додавання та віднімання (при великій різниці між числами) не погіршують точності результату порівняно з точністю алгебраїчних доданків, а операції множення, ділення і піднесення до степеня суттєво погіршують точність результату.

*Використовувані формули для виконання лабораторної роботи*

формула відносної похибки

формула наближеного числа через n знаків.

**Номер варіанту – 3.**

1. 
2. **Умова завдання**

Оцінити абсолютну та відносну похибку обчислення величини F при умові:

А) заданих точних цифр введених значень величин аргументів x1 , x2 , x3

Б) заданих значеннях величин аргументів x1 , x2 , x3 з похибкою ∆= N\*10^(-3), де N–номер варіанта N=3.

Програма повинна знаходити кількість точних десяткових знаків числа, визначати першу значущу цифру, абсолютну та відносну похибки аргументів (аргументи вводяться користувачем при запуску), абсолютну та відносну похибки заданої функції.

Доповнити текст програми таким чином, щоб у результатах виконання програми було відображено інформацію: група, ПІБ студента, номер ЛР і варіанту завдання.

1. **Алгоримт розв’язання завдання**
2. Створення класів для пункту А і Б.
3. Створення полів для збереження відповідних значень, які необхідні для обчислення.
4. Створення методів для обчислення і визначення величин (згідно теоретичних відомостей та алгоритму виконаного завдання вручну).
5. Послідовне розміщення методів у конструкторі класу для правильного виконання програми.
6. Визначення виняткових ситуацій та створення обробки для них.
7. Створення головної функції у новому класі, де виконується програма.
8. Ввід даних і перегляд результату. Відлаходження програми при некоректній роботі.
9. Коментування блоків коду.
10. **Виконане завдання вручну**
11. Пункт А:
    1. Обираємо конкретні значення змінних х1 = 0,051, х2 = 6,123, х3 = 3,07 й кількість значущих цифр n1 = 2, n2 = 3, n3 = 2;
    2. Визначаємо першу значущу цифру а1 = 5, а2 = 6, а3 = 3.
    3. Визначаємо відносну похибку х1, х2, х3  за формулою , де значення значущої цифри (m), n – кількість значущих цифр. Отже, результат обчилень: . Для подальшого обчислення будуть використовуватися ці дробові значення, але буде переведено у відсотковий вигляд в програмі.
    4. Визначаємо абсолютну похибку х1, х2, х3  за формулою , де значення відносної похибки змінної, х – значення змінної. Отже, результат обчилень: . Для подальшого обчислення будуть використовуватися ці дробові значення.
    5. Обчислюю значення функції підставивши змінні: .
    6. Обчислюю частинну похідну функції у формулі абсолютної похибки функції для її розрахунку: . Кінцевий вигляд моєї частинної похідної й її значення після підстановки змінних:
    7. Обчислюю відносну похибку функції за формулою: . Отримав значення , але буде переведено у відсотковий вигляд в програмі.
12. Пункт Б:
    1. Обираємо конкретні значення змінних х1 = 0,051, х2 = 6,123, х3 = 3,07 й обраховуємо абсолютну похибку змінної за формулою , де N – номер варіанту. Для усіх змінних буде абсолютна похибка .
    2. Визначаємо першу значущу цифру кожної змінної a1 = 5, a2 = 6, a3 = 3 й записую їх степінь m1 = -2, m2 = 0, m3 = 0 (визначається із формули: ). Зауважу, що тепер взято символ m для позначення степеня першої значущої цифри.
    3. З формули виводжу формулу . Обраховую кількість значущих цифр за виведеною формулою: n1 = 1, n2 = 3, n3 = 3.
    4. Визначаємо відносну похибку х1, х2, х3  за формулою , де значення значущої цифри, n – кількість значущих цифр. Отже, результат обчилень: . Для подальшого обчислення будуть використовуватися ці дробові значення, але буде переведено у відсотковий вигляд в програмі.
    5. Обчислюю значення функції підставивши змінні: .
    6. Обчислюю частинну похідну функції у формулі абсолютної похибки функції для її розрахунку: . Кінцевий вигляд моєї частинної похідної й її значення після підстановки змінних:
    7. Обчислюю відносну похибку функції за формулою: . Отримав значення , але буде переведено у відсотковий вигляд в програмі.
13. **Код програмної реалізації**
14. *Файл TaskA.java (реалізація пункту А)*
15. package lr1;
16. /\*\*
17. \*
18. \* @author Valchevskyi
19. \*/
20. public final class TaskA {
21. private boolean correctData; // Змінна, яка показує коректність обчислень.
22. private double valueFx; // Значення функції.
23. private double deltaFx; // Абсолютна похибка функції.
24. private double dFx; // Відносна похибка функції.
25. private double x1, x2, x3; // Значення змінних.
26. private int x1m, x2m, x3m; // Значення першої значущої цифри змінних.
27. private int x1n, x2n, x3n; // Кількість значущих цифр змінних.
28. private double x1d, x2d, x3d; // Відносні похибки змінних
29. private double x1delta, x2delta, x3delta; // Абсолютні похибки змінних.
31. public TaskA(double x1, double x2, double x3, int x1n, int x2n, int x3n) {
32. try {
33. setX(x1, x2, x3);
34. setN(x1n, x2n, x3n);
35. setM();
36. setD();
37. setDelta();
38. setFx();
39. setDeltaFx();
40. setDFx();
41. correctData = true;
42. } catch (Exception e) {
43. System.out.println(e.getMessage());
44. correctData = false;
45. }
46. }
48. // Повернення стану коректності даних.
49. public boolean getCorrectData() {
50. return correctData;
51. }
53. // Встановлення змінних.
54. public void setX(double x1, double x2, double x3) throws Exception {
55. if (x1 != 0 && x2 != 0 && x3 != 0) {
56. this.x1 = x1;
57. this.x2 = x2;
58. this.x3 = x3;
59. } else {
60. throw new Exception("\tПомилка: Змінна не може бути нулем!");
61. }
62. }
64. // Встановлення кількості значущих цифр, якщо одна змінна має 0 значущих цифр - виключення.
65. public void setN(int x1n, int x2n, int x3n) throws Exception {
66. if (x1n > 0 && x2n > 0 && x3n > 0) {
67. this.x1n = x1n;
68. this.x2n = x2n;
69. this.x3n = x3n;
70. } else {
71. throw new Exception("\tПомилка: Значущі цифри мають бути більші за нуль!");
72. }
73. }
75. // Встановлення першої значущої цифри, якщо одна змінна має 0 значущих цифр - виключення.
76. public void setM() throws Exception {
77. x1m = getXm(x1);
78. x2m = getXm(x2);
79. x3m = getXm(x3);
80. if (!(x1m > 0 && x2m > 0 && x3m > 0)) {
81. throw new Exception("\tПомилка: Перша значуща цифра має бути більша за нуль!");
82. }
83. }
85. // Отримання першої значущої цифри з числа, якщо немає взаглі значущих цифр - повернення нуля.
86. private int getXm(double x) {
87. String xStr = String.valueOf(x);
88. for (int i = 0; i < xStr.length(); i++) {
89. int value = Character.getNumericValue(xStr.charAt(i));
90. if (value != 0 && value != -1) { // Значення -1 - кома або знак мінус, 0 - 0.
91. return value;
92. }
93. }
94. return 0;
95. }
97. // Встановлення відносних похибок змінних.
98. public void setD() {
99. x1d = getXd(x1n, x1m);
100. x2d = getXd(x2n, x2m);
101. x3d = getXd(x3n, x3m);
102. }
104. // Розрахунок відносної похибки змінної.
105. private double getXd(int n, int m) {
106. double value = 1.0 / m \* Math.pow(1.0 / 10, n - 1);
107. return (n >= 2) ? 0.5 \* value : value;
108. }
110. // Встановлення абсолютних похибок.
111. public void setDelta() {
112. x1delta = getXdelta(x1d, x1);
113. x2delta = getXdelta(x2d, x2);
114. x3delta = getXdelta(x3d, x3);
115. }
117. // Розрахунок абсолютних похибок зміної.
118. private double getXdelta(double d, double x) {
119. return Math.abs(d \* x);
120. }
122. // Встановлення значення функції.
123. public void setFx() {
124. valueFx = 3 \* Math.pow(x1, 2) + 2 \* Math.pow(x2, 2) + 4 \* Math.pow(x3, 2) + 3 \* x1 \* x2 - 2 \* x2 + Math.sin(x1 - x3 \* x2);
125. }
127. // Встановлення абсолютної похибки функції.
128. public void setDeltaFx() {
129. double partX1 = 6 \* x1 + 3 \* x2 + Math.cos(x1 - x2 \* x3);
130. double partX2 = 4 \* x2 + 3 \* x1 - 2 + Math.cos(x1 - x2 \* x3) \* -x3;
131. double partX3 = 8 \* x3 + Math.cos(x1 - x2 \* x3) \* -x2;
132. deltaFx = Math.abs(partX1) \* x1delta + Math.abs(partX2) \* x2delta + Math.abs(partX3) \* x3delta;
133. }
135. // Встановлення відносної похибки фукнції.
136. public void setDFx() throws Exception {
137. if (valueFx == 0) {
138. throw new Exception("Ділення на нуль. Ви ввели такі значення за яких функція приймає значення 0!");
139. }
140. dFx = Math.abs(deltaFx / valueFx);
141. }
143. // Вивід усіх результатів.
144. @Override
145. public String toString() {
146. StringBuilder stringBuilder = new StringBuilder("Результати пункту А (група ОІ-11 сп; Вальчевський Павло Володимирович; ЛР № 1; варіант № 3):\n");
147. int countNumberInPercent = 2;
148. int countNumberInExpression = 5;
149. if (correctData) {
150. stringBuilder.append("\tx1: ").append(x1).append("\n");
151. stringBuilder.append("\tx2: ").append(x2).append("\n");
152. stringBuilder.append("\tx3: ").append(x3).append("\n");
153. stringBuilder.append("\tx1 кількість значущих цифр: ").append(x1n).append("\n");
154. stringBuilder.append("\tx2 кількість значущих цифр: ").append(x2n).append("\n");
155. stringBuilder.append("\tx3 кількість значущих цифр: ").append(x3n).append("\n");
156. stringBuilder.append("\tx1 перша значуща цифра: ").append(x1m).append("\n");
157. stringBuilder.append("\tx2 перша значуща цифра: ").append(x2m).append("\n");
158. stringBuilder.append("\tx3 перша значуща цифра: ").append(x3m).append("\n");
159. stringBuilder.append("\tx1 відносна похибка у %: ").append(getPercent(x1d, countNumberInPercent)).append("\n");
160. stringBuilder.append("\tx2 відносна похибка у %: ").append(getPercent(x2d, countNumberInPercent)).append("\n");
161. stringBuilder.append("\tx3 відносна похибка у %: ").append(getPercent(x3d, countNumberInPercent)).append("\n");
162. stringBuilder.append("\tx1 абсолютна похибка: ").append(getRound(x1delta, countNumberInExpression)).append("\n");
163. stringBuilder.append("\tx2 абсолютна похибка: ").append(getRound(x2delta, countNumberInExpression)).append("\n");
164. stringBuilder.append("\tx3 абсолютна похибка: ").append(getRound(x3delta, countNumberInExpression)).append("\n");
165. stringBuilder.append("\tЗначення функції: ").append(getRound(valueFx, countNumberInExpression)).append("\n");
166. stringBuilder.append("\tАбсолютна похибка функції: ").append(getRound(deltaFx, countNumberInExpression)).append("\n");
167. stringBuilder.append("\tВідносна похибка функції у %: ").append(getPercent(dFx, 2)).append("\n");
168. } else {
169. stringBuilder.append("\tЧерез погано введені дані, обчислення неможливе!");
170. }
171. return stringBuilder.toString();
172. }
174. // Повернення дробового числа у відсотковому форматі
175. private double getPercent(double value, int f) {
176. return Math.round(value \* Math.pow(10, f));
177. }
179. // Повернення округленого числа до f-знаків після коми
180. private double getRound(double value, int f) {
181. return Math.round(value \* Math.pow(10, f)) / Math.pow(10, f);
182. }
183. }
184. *Файл TaskB.java (реалізація пункту Б)*
185. package lr1;
186. import java.text.DecimalFormat; // Для створення відповідного формату.
187. /\*\*
188. \*
189. \* @author Valchevskyi
190. \*/
191. public class TaskB {
192. private boolean correctData; // Змінна, яка показує коректність обчислень.
193. private int variant; // Змінна мого варіанту.
194. private double valueFx; // Значення функції.
195. private double deltaFx; // Абсолютна похибка функції.
196. private double dFx; // Відносна похибка функції.
197. private double x1, x2, x3; // Значення змінних.
198. private int x1m, x2m, x3m; // Значення першої значущої цифри змінних.
199. private int x1mPower, x2mPower, x3mPower; // Значення степені першої значущої цифри.
200. private int x1n, x2n, x3n; // Кількість значущих цифр змінних.
201. private double x1d, x2d, x3d; // Відносні похибки змінних
202. private double x1delta, x2delta, x3delta; // Абсолютні похибки змінних.
204. public TaskB(double x1, double x2, double x3) {
205. try {
206. variant = 3;
207. setX(x1, x2, x3);
208. setDelta();
209. setMpower();
210. setN();
211. setM();
212. setD();
213. setFx();
214. setDeltaFx();
215. setDFx();
216. correctData = true;
217. } catch (Exception e) {
218. System.out.println(e.getMessage());
219. correctData = false;
220. }
221. }
223. // Повернення стану коректності даних.
224. public boolean getCorrectData() {
225. return correctData;
226. }
228. // Встановлення змінних.
229. public void setX(double x1, double x2, double x3) throws Exception {
230. if (x1 != 0 && x2 != 0 && x3 != 0) {
231. this.x1 = x1;
232. this.x2 = x2;
233. this.x3 = x3;
234. } else {
235. throw new Exception("\tПомилка: Змінна не може бути нулем!");
236. }
237. }
239. // Встановлення кількості значущих цифр, якщо одна змінна має 0 значущих цифр - виключення.
240. public void setN() throws Exception {
241. x1n = getN(x1mPower, x1delta);
242. x2n = getN(x2mPower, x2delta);
243. x3n = getN(x3mPower, x3delta);
244. if (!(x1n > 0 && x2n > 0 && x3n > 0)) {
245. throw new Exception("\tПомилка: Значущі цифри мають бути більші за нуль!");
246. }
247. }
249. // Отримання кількості значущих цифр.
250. private int getN(double xM, double xDelta) {
251. return (int) Math.abs(xM + 1 - Math.log10(2 \* xDelta));
252. }
254. // Встановлення першої значущої цифри, якщо одна змінна має 0 значущих цифр - виключення.
255. public void setM() throws Exception {
256. x1m = getXm(x1);
257. x2m = getXm(x2);
258. x3m = getXm(x3);
259. if (!(x1m > 0 && x2m > 0 && x3m > 0)) {
260. throw new Exception("\tПомилка: Перша значуща цифра має бути більша за нуль!");
261. }
262. }
264. // Отримання першої значущої цифри з числа, якщо немає взаглі значущих цифр - повернення нуля.
265. private int getXm(double x) {
266. String xStr = String.valueOf(x);
267. for (int i = 0; i < xStr.length(); i++) {
268. int value = Character.getNumericValue(xStr.charAt(i));
269. if (value != 0 && value != -1) { // Значення -1 - кома або знак мінус, 0 - 0.
270. return value;
271. }
272. }
273. return 0;
274. }
276. // Встановлення першої степені першої значущої цифри числа
277. public void setMpower() throws Exception {
278. x1mPower = getXmPower(x1);
279. x2mPower = getXmPower(x2);
280. x3mPower = getXmPower(x3);
281. }
283. // Отримання степені першої значущої цифри числа
284. private int getXmPower(double x) {
285. DecimalFormat decimalFormat = new DecimalFormat("#.################"); // Для зміни формату числа пре перетворенні в рядок.
286. String xStr = decimalFormat.format(x);
287. int indexMinusInStr = xStr.indexOf('-');
288. int indexCommaInText = xStr.indexOf(',');
289. int index = -1;
291. if (indexCommaInText != -1) {
292. String positivePowerInNumber = xStr.substring(indexMinusInStr + 1, indexCommaInText);
293. String negativePowerInNumber = xStr.substring(indexCommaInText + 1, xStr.length());
294. boolean searchingIndex = true;
295. for (int i = 0, j = positivePowerInNumber.length() - 1; i < positivePowerInNumber.length(); i++, j--) { // Прохід позитивного степеня числа.
296. int value = Character.getNumericValue(positivePowerInNumber.charAt(i));
297. if (value != 0) {
298. index = j;
299. searchingIndex = false;
300. break;
301. }
302. }
303. if (searchingIndex) { // Прохід від'ємного степеня числа.
304. for (int i = 0, j = -1; i < negativePowerInNumber.length(); i++, j--) {
305. int value = Character.getNumericValue(negativePowerInNumber.charAt(i));
306. if (value != 0) {
307. index = j;
308. break;
309. }
310. }
311. }
312. } else {
313. String positivePowerInNumber = xStr.substring(indexMinusInStr + 1, xStr.length());
314. for (int i = 0, j = positivePowerInNumber.length() - 1; i < positivePowerInNumber.length(); i++, j--) { // Прохід позитивного степеня числа.
315. int value = Character.getNumericValue(positivePowerInNumber.charAt(i));
316. if (value != 0) {
317. index = j;
318. break;
319. }
320. }
321. }
323. return index;
324. }
326. // Встановлення відносних похибок змінних.
327. public void setD() {
328. x1d = getXd(x1n, x1m);
329. x2d = getXd(x2n, x2m);
330. x3d = getXd(x3n, x3m);
331. }
333. // Розрахунок відносної похибки змінної.
334. private double getXd(int n, int m) {
335. double value = 1.0 / m \* Math.pow(1.0 / 10, n - 1);
336. return (n >= 2) ? 0.5 \* value : value;
337. }
339. // Встановлення абсолютних похибок.
340. public void setDelta() {
341. x1delta = getXdelta();
342. x2delta = getXdelta();
343. x3delta = getXdelta();
344. }
346. // Розрахунок абсолютних похибок зміної.
347. private double getXdelta() {
348. return variant \* Math.pow(10, -3);
349. }
351. // Встановлення значення функції.
352. public void setFx() {
353. valueFx = 3 \* Math.pow(x1, 2) + 2 \* Math.pow(x2, 2) + 4 \* Math.pow(x3, 2) + 3 \* x1 \* x2 - 2 \* x2 + Math.sin(x1 - x3 \* x2);
354. }
356. // Встановлення абсолютної похибки функції.
357. public void setDeltaFx() {
358. double partX1 = 6 \* x1 + 3 \* x2 + Math.cos(x1 - x2 \* x3);
359. double partX2 = 4 \* x2 + 3 \* x1 - 2 + Math.cos(x1 - x2 \* x3) \* -x3;
360. double partX3 = 8 \* x3 + Math.cos(x1 - x2 \* x3) \* -x2;
361. deltaFx = Math.abs(partX1) \* x1delta + Math.abs(partX2) \* x2delta + Math.abs(partX3) \* x3delta;
362. }
364. // Встановлення відносної похибки фукнції.
365. public void setDFx() throws Exception {
366. if (valueFx == 0) {
367. throw new Exception("Ділення на нуль. Ви ввели такі значення за яких функція приймає значення 0!");
368. }
369. dFx = Math.abs(deltaFx / valueFx);
370. }
372. // Вивід усіх результатів.
373. @Override
374. public String toString() {
375. StringBuilder stringBuilder = new StringBuilder("Результати пункту Б (група ОІ-11 сп; Вальчевський Павло Володимирович; ЛР № 1; варіант № 3):\n");
376. int countNumberInPercent = 2;
377. int countNumberInExpression = 5;
378. if (correctData) {
379. stringBuilder.append("\tx1: ").append(x1).append("\n");
380. stringBuilder.append("\tx2: ").append(x2).append("\n");
381. stringBuilder.append("\tx3: ").append(x3).append("\n");
382. stringBuilder.append("\tx1 кількість значущих цифр (йшли розрахунки): ").append(x1n).append("\n");
383. stringBuilder.append("\tx2 кількість значущих цифр (йшли розрахунки): ").append(x2n).append("\n");
384. stringBuilder.append("\tx3 кількість значущих цифр (йшли розрахунки): ").append(x3n).append("\n");
385. stringBuilder.append("\tx1 перша значуща цифра: ").append(x1m).append("\n");
386. stringBuilder.append("\tx2 перша значуща цифра: ").append(x2m).append("\n");
387. stringBuilder.append("\tx3 перша значуща цифра: ").append(x3m).append("\n");
388. stringBuilder.append("\tx1 степінь першої значущої: ").append(x1mPower).append("\n");
389. stringBuilder.append("\tx2 степінь першої значущої: ").append(x2mPower).append("\n");
390. stringBuilder.append("\tx3 степінь першої значущої: ").append(x3mPower).append("\n");
391. stringBuilder.append("\tx1 відносна похибка у %: ").append(getPercent(x1d, countNumberInPercent)).append("\n");
392. stringBuilder.append("\tx2 відносна похибка у %: ").append(getPercent(x2d, countNumberInPercent)).append("\n");
393. stringBuilder.append("\tx3 відносна похибка у %: ").append(getPercent(x3d, countNumberInPercent)).append("\n");
394. stringBuilder.append("\tx1 абсолютна похибка (задане через варіант): ").append(getRound(x1delta, countNumberInExpression)).append("\n");
395. stringBuilder.append("\tx2 абсолютна похибка (задане через варіант): ").append(getRound(x2delta, countNumberInExpression)).append("\n");
396. stringBuilder.append("\tx3 абсолютна похибка (задане через варіант): ").append(getRound(x3delta, countNumberInExpression)).append("\n");
397. stringBuilder.append("\tЗначення функції: ").append(getRound(valueFx, countNumberInExpression)).append("\n");
398. stringBuilder.append("\tАбсолютна похибка функції: ").append(getRound(deltaFx, countNumberInExpression)).append("\n");
399. stringBuilder.append("\tВідносна похибка функції у %: ").append(getPercent(dFx, countNumberInPercent)).append("\n");
400. } else {
401. stringBuilder.append("\tЧерез погано введені дані, обчислення неможливе!");
402. }
403. return stringBuilder.toString();
404. }
406. // Повернення дробового числа у відсотковому форматі
407. private double getPercent(double value, int f) {
408. return Math.round(value \* Math.pow(10, f));
409. }
411. // Повернення округленого числа до f-знаків після коми
412. private double getRound(double value, int f) {
413. return Math.round(value \* Math.pow(10, f)) / Math.pow(10, f);
414. }
415. }
416. *Файл LR1.java (виконання програми у головній функції)*

package lr1;

import java.util.Scanner;

/\*\*

\*

\* @author Valchevskyi

\*/

public class LR1 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) { // Виконання програми через головну функцію проекту.

while (true) {

try {

Scanner scanner = new Scanner(System.in);

double x1 = inputDouble("x1 (дробове число)", scanner);

double x2 = inputDouble("x2 (дробове число)", scanner);

double x3 = inputDouble("x3 (дробове число)", scanner);

int x1n = inputInt("x1 к-сть значущих цифр (ціле число)", scanner);

int x2n = inputInt("x2 к-сть значущих цифр (ціле число)", scanner);

int x3n = inputInt("x3 к-сть значущих цифр (ціле число)", scanner);

TaskA taskA = new TaskA(x1, x2, x3, x1n, x2n, x3n);

System.out.println(taskA.toString());

TaskB taskB = new TaskB(x1, x2, x3);

System.out.println(taskB.toString());

if (taskA.getCorrectData() && taskB.getCorrectData()) {

break;

}

} catch (Exception e) {

System.out.println("\tПомилка: Уведіть числа згідно повідомлень!");

}

}

}

public static double inputDouble(String nameVar, Scanner scanner) {

System.out.print(nameVar + ": ");

return scanner.nextDouble();

}

public static int inputInt(String nameVar, Scanner scanner) {

System.out.print(nameVar + ": ");

return scanner.nextInt();

}

}

**Скріншот отриманого результату**

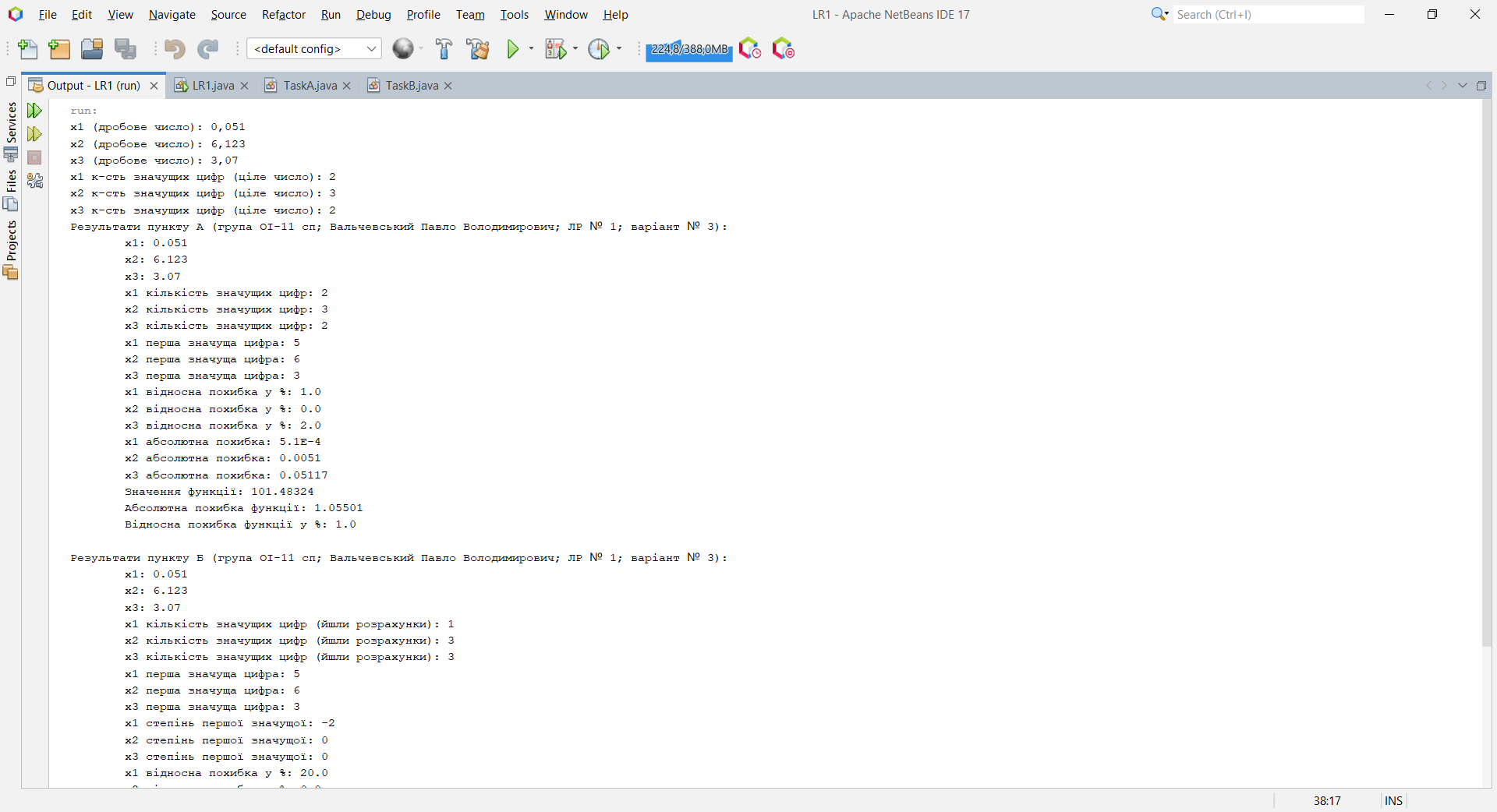
1. 

Рисунок 1 Результат виконання програми з параметрами розв’язаних вручну.

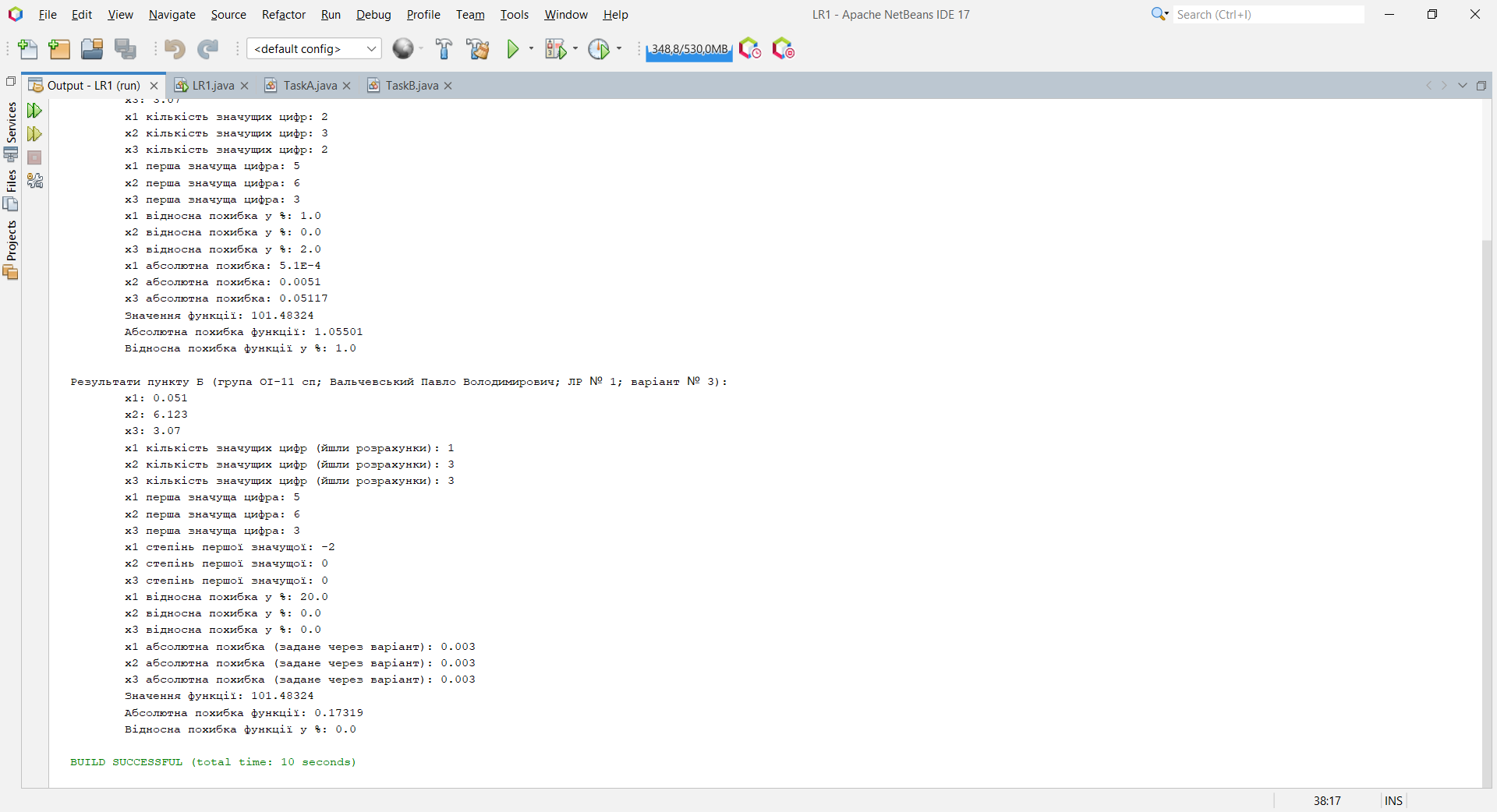
1. 

Рисунок 2 Результат виконання програми (продовження) з параметрами розв’язаних вручну.

1. **Висновки (з аналізом результату)**

Згідно результатів та підстановки різних параметрів у тому числі й параметрів розв’язаних вручну під час виконання програми – алгоритм виконується правильно та без помилок (з урахуванням виняткових ситуацій пов’язаних з обчисленням).

Реалізовуються умови для коректного обчислення, тобто, кожне подальше обчислення буде залежати від результатів попереднього й якщо дані не будуть правильно введені або не відповідатимуть обмеженням, то програма виведе повідомлення про виняткову ситуацію.

Програма працює поки користувач не введе такі дані за яких обрахується пункт А і Б, після обчислення цих пунктів програма завершить роботу і буде виведено усі результати повідомленням.

Програма була відлагоджена й виконана у програмному середовищі NetBeans 17 за допомогою JDK-17 (версія Java 17.0.8).