Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Изображение выглядит как Шрифт, логотип, Графика, белый

Автоматически созданное описание**

Лабораторная работа №4 по дисциплине «Вычислительная математика»

Вариант 13

Выполнил:

Студент группы P3212

Метель Леонард Валерьевич

Преподаватель:

г. Санкт-Петербург

2025

Вычислительная реализация задачи

Вариант   
Изображение выглядит как текст, Шрифт, белый, линия

Автоматически созданное описание

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | 0 | 0,4 | 0,8 | 1,2 | 1,6 | 2 | 2,4 | 2,8 | 3,2 | 3,6 | 4 |
| Y | 0 | 0,95 | 1,85 | 2,47 | 2,54 | 2,14 | 1,61 | 1,16 | 0,84 | 0,61 | 0,46 |

𝜑 (𝑥, 𝑎, 𝑏) = 𝑎𝑥 + 𝑏

Сумма квадратов отклонений запишется следующим образом:

𝑆 = 𝑆(𝑎, 𝑏) =

Для нахождения а и b необходимо найти минимум функции 𝑆(𝑎, 𝑏)

Введем обозначения:

SX = = 22 SXX = = 61,6 SY = = 14,63 SXY = = 27,004

Получим систему уравнений для нахождения параметров а и b:

из которой находим (правило Крамера):

∆ = 𝑆𝑋𝑋 \* 𝑛 − 𝑆𝑋 \* 𝑆𝑋 = 193,6

∆1= 𝑆𝑋𝑌 \* 𝑛 − 𝑆𝑋 \* 𝑆𝑌 = -24,816.

∆2= 𝑆𝑋𝑋 \* 𝑆𝑌 − 𝑆𝑋 \* 𝑆𝑋𝑌 = 307,12.

a = ∆1/∆ = -0,13

b = ∆2/∆ = 1,59

 Таким образом линейное приближение:   
𝜑 (𝑥, 𝑎, 𝑏) = -0,13x + 1,59

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| xi) | 1,59 | 1,54 | 1,49 | 1,43 | 1,382 | 1,33 | 1,27 | 1,22 | 1,17 | 1,12 | 1,07 |
|  | 1,59 | 0,59 | 0,36 | 1,04 | 1,16 | 0,81 | 0,34 | 0,06 | 0,33 | 0,51 | 0,41 |

0,78

Рассмотрим в качестве эмпирической формулы квадратичную функцию:

𝜑(𝑥, 𝑎0, 𝑎1, 𝑎2) = 𝑎0 + 𝑎1𝑥 + 𝑎2𝑥2

=. 14,63 = 27,004

= 62,19

Итак квадратичное приближение:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| xi) | 1,06 | 1,56 | 1,90 | 2,08 | 2,11 | 1,98 | 1,70 | 1,26 | 1,17 | 0,66 | -0,085 |
|  | 0,11 | 0,31 | 0,57 | 0,46 | 0,03 | 0,37 | 0,54 | 0,42 | 0,33 | 0,05 | 0,35 |

0,36

Лучшее приближение – квадратичное

График  
Изображение выглядит как линия, График, диаграмма, Параллельный

Автоматически созданное описание

Программная реализация:

1. **import** numpy as np
2. **import** matplotlib.pyplot as plt
3. **import** sys
4. **import** os
5. **import** re
7. **class** ApproximationMethod:
8. **def** \_\_init\_\_(self):
9. self.x **=** None
10. self.y **=** None
11. self.n **=** 0
12. self.approximations **=** {}
13. self.best\_approximation **=** None
15. **def** input\_data(self, source**=**'console'):
16. **if** source **==** 'console':
17. **while** True:
18. **try**:
19. n **=** input("Введите количество точек (8-12): ")
20. self.n **=** int(n)
21. **if** **not** 8 <**=** self.n <**=** 12:
22. print("Ошибка: Количество точек должно быть от 8 до 12. Попробуйте еще раз.")
23. **continue**
25. self.x **=** np.zeros(self.n)
26. self.y **=** np.zeros(self.n)
28. print("Введите пары значений x и y:")
29. i **=** 0
30. **while** i < self.n:
31. **try**:
32. values **=** input(f"Точка {i+1} (x y): ").split()
33. **if** len(values) !**=** 2:
34. print(f"Ошибка: Неверный формат ввода для точки {i+1}. Нужно ввести два числа, разделенных пробелом.")
35. **continue**
37. x\_val **=** float(values[0])
38. y\_val **=** float(values[1])
39. self.x[i] **=** x\_val
40. self.y[i] **=** y\_val
41. i **+=** 1  # Переходим к следующей точке только если ввод корректен
42. **except** ValueError:
43. print(f"Ошибка: Введите числовые значения для точки {i+1}. Попробуйте еще раз.")
45. **return**  # Успешный ввод, выходим из цикла
46. **except** ValueError:
47. print("Ошибка: Введите корректное числовое значение для количества точек. Попробуйте еще раз.")
48. **else**:
49. attempts **=** 0
50. max\_attempts **=** 3
51. **while** attempts < max\_attempts:
52. **try**:
53. **if** **not** os.path.exists(source):
54. print(f"Ошибка: Файл {source} не найден. Попробуйте еще раз.")
55. source **=** input("Введите корректное имя файла: ")
56. attempts **+=** 1
57. **continue**
59. with open(source, 'r') as f:
60. lines **=** f.readlines()
62. points **=** []
63. **for** line **in** lines:
64. matches **=** re.findall(r'-?\d+\.?\d\*', line)
65. **if** len(matches) >**=** 2:
66. points.append((float(matches[0]), float(matches[1])))
68. self.n **=** len(points)
69. **if** **not** 8 <**=** self.n <**=** 12:
70. print(f"Ошибка: В файле {source} содержится {self.n} точек, а должно быть от 8 до 12.")
71. source **=** input("Введите другое имя файла: ")
72. attempts **+=** 1
73. **continue**
75. self.x **=** np.array([point[0] **for** point **in** points])
76. self.y **=** np.array([point[1] **for** point **in** points])
77. **return**
79. **except** Exception as e:
80. print(f"Ошибка при чтении файла: {e}")
81. **if** attempts < max\_attempts **-** 1:
82. source **=** input("Введите другое имя файла: ")
83. attempts **+=** 1
85. print(f"После {max\_attempts} неудачных попыток чтения файла, переключаемся на ввод с консоли.")
86. self.input\_data('console')
88. **def** approximate(self):
89. self.lineral()
90. self.polynomial(2)
91. self.polynomial(3)
92. self.exponential()
93. self.logarithmic()
94. self.power()
96. self.find\_best()
98. **def** lineral(self):
99. **try**:
100. a, b **=** np.polyfit(self.x, self.y, 1)
101. f\_x **=** a **\*** self.x **+** b
102. errors **=** f\_x **-** self.y
103. mse **=** np.mean(errors**\*\***2)
104. rmse **=** np.sqrt(mse)
105. r **=** np.corrcoef(self.x, self.y)[0, 1]
106. r\_squared **=** r**\*\***2
108. self.approximations['linear'] **=** {
109. 'type': 'linear',
110. 'params': {'a': a, 'b': b},
111. 'function': **lambda** x: a **\*** x **+** b,
112. 'formula': f'f(x) = {a:.6f}\*x + {b:.6f}',
113. 'f\_x': f\_x,
114. 'errors': errors,
115. 'rmse': rmse,
116. 'r': r,
117. 'r\_squared': r\_squared
118. }
119. **except** Exception as e:
120. print(f"Ошибка при линейной аппроксимации: {e}")
121. self.approximations['linear'] **=** None
123. **def** polynomial(self, degree):
124. **try**:
125. coeffs **=** np.polyfit(self.x, self.y, degree)
126. f\_x **=** np.polyval(coeffs, self.x)
127. errors **=** f\_x **-** self.y
128. mse **=** np.mean(errors**\*\***2)
129. rmse **=** np.sqrt(mse)
130. ss\_total **=** np.sum((self.y **-** np.mean(self.y))**\*\***2)
131. ss\_residual **=** np.sum(errors**\*\***2)
132. r\_squared **=** 1 **-** (ss\_residual **/** ss\_total)
134. formula **=** "f(x) = "
135. **for** i, coef **in** enumerate(coeffs):
136. power **=** degree **-** i
137. **if** power > 1:
138. formula **+=** f"{coef:.6f}\*x^{power} + "
139. **elif** power **==** 1:
140. formula **+=** f"{coef:.6f}\*x + "
141. **else**:
142. formula **+=** f"{coef:.6f}"
144. self.approximations[f'polynomial\_{degree}'] **=** {
145. 'type': f'polynomial\_{degree}',
146. 'params': {'coeffs': coeffs},
147. 'function': **lambda** x: np.polyval(coeffs, x),
148. 'formula': formula,
149. 'f\_x': f\_x,
150. 'errors': errors,
151. 'rmse': rmse,
152. 'r\_squared': r\_squared
153. }
154. **except** Exception as e:
155. print(f"Ошибка при полиномиальной аппроксимации степени {degree}: {e}")
156. self.approximations[f'polynomial\_{degree}'] **=** None
158. **def** exponential(self):
159. **try**:
160. **if** np.any(self.y <**=** 0):
161. **raise** ValueError("Экспоненциальная аппроксимация невозможна для отрицательных значений y")
163. ln\_y **=** np.log(self.y)
164. b, ln\_a **=** np.polyfit(self.x, ln\_y, 1)
165. a **=** np.exp(ln\_a)
166. f\_x **=** a **\*** np.exp(b **\*** self.x)
167. errors **=** f\_x **-** self.y
168. mse **=** np.mean(errors**\*\***2)
169. rmse **=** np.sqrt(mse)
170. ss\_total **=** np.sum((self.y **-** np.mean(self.y))**\*\***2)
171. ss\_residual **=** np.sum(errors**\*\***2)
172. r\_squared **=** 1 **-** (ss\_residual **/** ss\_total)
174. self.approximations['exponential'] **=** {
175. 'type': 'exponential',
176. 'params': {'a': a, 'b': b},
177. 'function': **lambda** x: a **\*** np.exp(b **\*** x),
178. 'formula': f'f(x) = {a:.6f}\*exp({b:.6f}\*x)',
179. 'f\_x': f\_x,
180. 'errors': errors,
181. 'rmse': rmse,
182. 'r\_squared': r\_squared
183. }
184. **except** Exception as e:
185. print(f"Ошибка при экспоненциальной аппроксимации: {e}")
186. self.approximations['exponential'] **=** None
188. **def** logarithmic(self):
189. **try**:
190. **if** np.any(self.x <**=** 0):
191. **raise** ValueError("Логарифмическая аппроксимация невозможна для отрицательных значений x")
193. ln\_x **=** np.log(self.x)
194. a, b **=** np.polyfit(ln\_x, self.y, 1)
196. f\_x **=** a **\*** np.log(self.x) **+** b
197. errors **=** f\_x **-** self.y
198. mse **=** np.mean(errors**\*\***2)
199. rmse **=** np.sqrt(mse)
200. ss\_total **=** np.sum((self.y **-** np.mean(self.y))**\*\***2)
201. ss\_residual **=** np.sum(errors**\*\***2)
202. r\_squared **=** 1 **-** (ss\_residual **/** ss\_total)
204. self.approximations['logarithmic'] **=** {
205. 'type': 'logarithmic',
206. 'params': {'a': a, 'b': b},
207. 'function': **lambda** x: a **\*** np.log(x) **+** b,
208. 'formula': f'f(x) = {a:.6f}\*ln(x) + {b:.6f}',
209. 'f\_x': f\_x,
210. 'errors': errors,
211. 'rmse': rmse,
212. 'r\_squared': r\_squared
213. }
214. **except** Exception as e:
215. print(f"Ошибка при логарифмической аппроксимации: {e}")
216. self.approximations['logarithmic'] **=** None
218. **def** power(self):
219. **try**:
220. **if** np.any(self.x <**=** 0) **or** np.any(self.y <**=** 0):
221. **raise** ValueError("Степенная аппроксимация невозможна для отрицательных значений x или y")
223. ln\_x **=** np.log(self.x)
224. ln\_y **=** np.log(self.y)
225. b, ln\_a **=** np.polyfit(ln\_x, ln\_y, 1)
226. a **=** np.exp(ln\_a)
228. f\_x **=** a **\*** np.power(self.x, b)
229. errors **=** f\_x **-** self.y
230. mse **=** np.mean(errors**\*\***2)
231. rmse **=** np.sqrt(mse)
232. ss\_total **=** np.sum((self.y **-** np.mean(self.y))**\*\***2)
233. ss\_residual **=** np.sum(errors**\*\***2)
234. r\_squared **=** 1 **-** (ss\_residual **/** ss\_total)
236. self.approximations['power'] **=** {
237. 'type': 'power',
238. 'params': {'a': a, 'b': b},
239. 'function': **lambda** x: a **\*** np.power(x, b),
240. 'formula': f'f(x) = {a:.6f}\*x^{b:.6f}',
241. 'f\_x': f\_x,
242. 'errors': errors,
243. 'rmse': rmse,
244. 'r\_squared': r\_squared
245. }
246. **except** Exception as e:
247. print(f"Ошибка при степенной аппроксимации: {e}")
248. self.approximations['power'] **=** None
250. **def** find\_best(self):
251. min\_rmse **=** float('inf')
252. **for** appr\_type, data **in** self.approximations.items():
253. **if** data **is** **not** None **and** data['rmse'] < min\_rmse:
254. min\_rmse **=** data['rmse']
255. self.best\_approximation **=** appr\_type
257. **def** print\_results(self, output**=**'console'):
258. results **=** ["\n" **+** "="**\***80, "РЕЗУЛЬТАТЫ АППРОКСИМАЦИИ", "="**\***80]
259. results.append(f"\nИсходные данные (всего {self.n} точек):")
260. **for** i **in** range(self.n):
261. results.append(f"Точка {i+1}: x = {self.x[i]:.6f}, y = {self.y[i]:.6f}")
263. results.append("\nРезультаты аппроксимации:")
264. **for** appr\_type, data **in** self.approximations.items():
265. **if** data **is** None:
266. results.append(f"\n{appr\_type.upper()}: не удалось выполнить аппроксимацию")
267. **continue**
269. results.append(f"\n{appr\_type.upper()}:")
270. results.append(f"Формула: {data['formula']}")
271. results.append(f"Среднеквадратическое отклонение (RMSE): {data['rmse']:.6f}")
273. **if** appr\_type **==** 'linear':
274. r **=** data['r']
275. results.append(f"Коэффициент корреляции Пирсона: {r:.6f}")
276. r\_abs **=** abs(r)
277. **if** r\_abs < 0.3:
278. corr **=** "очень слабая"
279. **elif** r\_abs < 0.5:
280. corr **=** "слабая"
281. **elif** r\_abs < 0.7:
282. corr **=** "средняя"
283. **elif** r\_abs < 0.9:
284. corr **=** "сильная"
285. **else**:
286. corr **=** "очень сильная"
287. direction **=** "положительная" **if** r > 0 **else** "отрицательная"
288. results.append(f"Интерпретация: {corr} {direction} корреляция")
290. r\_squared **=** data['r\_squared']
291. results.append(f"Коэффициент детерминации (R^2): {r\_squared:.6f}")
292. **if** r\_squared < 0.3:
293. determ **=** "Модель объясняет малую часть дисперсии данных"
294. **elif** r\_squared < 0.5:
295. determ **=** "Модель объясняет некоторую часть дисперсии данных"
296. **elif** r\_squared < 0.7:
297. determ **=** "Модель хорошо объясняет дисперсию данных"
298. **else**:
299. determ **=** "Модель очень хорошо объясняет дисперсию данных"
300. results.append(f"Интерпретация: {determ}")
302. results.append("\nТочки аппроксимации:")
303. **for** i **in** range(min(5, self.n)):  # Выводим только первые 5 точек для краткости
304. error **=** data['errors'][i]
305. results.append(f"Точка {i+1}: x = {self.x[i]:.6f}, y = {self.y[i]:.6f}, f(x) = {data['f\_x'][i]:.6f}, e = {error:.6f}")
306. **if** self.n > 5:
307. results.append("...")
309. results.append("\n" **+** "="**\***80)
310. results.append(f"НАИЛУЧШАЯ АППРОКСИМАЦИЯ: {self.best\_approximation.upper()}")
311. results.append(f"Формула: {self.approximations[self.best\_approximation]['formula']}")
312. results.append(f"Среднеквадратическое отклонение (RMSE): {self.approximations[self.best\_approximation]['rmse']:.6f}")
313. results.append("="**\***80)
315. **if** output **==** 'console':
316. **for** line **in** results:
317. print(line)
318. **else**:
319. with open(output, 'w') as f:
320. **for** line **in** results:
321. f.write(line **+** "\n")
322. print(f"Результаты сохранены в файл {output}")
324. **def** plot\_graphs(self, save\_to\_file**=**None):
325. fig, axs **=** plt.subplots(2, 3, figsize**=**(15, 10))
326. axs **=** axs.flatten()
328. x\_min, x\_max **=** np.min(self.x), np.max(self.x)
329. y\_min, y\_max **=** np.min(self.y), np.max(self.y)
331. x\_range, y\_range **=** x\_max **-** x\_min, y\_max **-** y\_min
332. x\_min, x\_max **=** x\_min **-** 0.2 **\*** x\_range, x\_max **+** 0.2 **\*** x\_range
333. y\_min, y\_max **=** y\_min **-** 0.2 **\*** y\_range, y\_max **+** 0.2 **\*** y\_range
335. x\_grid **=** np.linspace(x\_min, x\_max, 1000)
337. type\_names **=** {
338. 'linear': 'Линейная',
339. 'polynomial\_2': 'Полиномиальная 2-й степени',
340. 'polynomial\_3': 'Полиномиальная 3-й степени',
341. 'exponential': 'Экспоненциальная',
342. 'logarithmic': 'Логарифмическая',
343. 'power': 'Степенная'
344. }
346. **for** i, (appr\_type, data) **in** enumerate(self.approximations.items()):
347. **if** data **is** None:
348. axs[i].set\_title(f"{type\_names.get(appr\_type, appr\_type)}: не удалось")
349. **continue**
351. **try**:
352. **if** appr\_type **in** ['logarithmic', 'power']:
353. valid\_x **=** x\_grid[x\_grid > 0]
354. y\_grid **=** data['function'](valid\_x)
355. axs[i].plot(valid\_x, y\_grid, 'r-', linewidth**=**2)
356. **else**:
357. y\_grid **=** data['function'](x\_grid)
358. axs[i].plot(x\_grid, y\_grid, 'r-', linewidth**=**2)
360. axs[i].scatter(self.x, self.y, color**=**'blue', s**=**30)
361. axs[i].set\_title(f"{type\_names.get(appr\_type, appr\_type)}: RMSE={data['rmse']:.4f}")
362. axs[i].set\_xlabel('x')
363. axs[i].set\_ylabel('y')
364. axs[i].text(0.05, 0.95, data['formula'], transform**=**axs[i].transAxes,
365. fontsize**=**8, verticalalignment**=**'top', bbox**=**dict(boxstyle**=**'round', facecolor**=**'wheat', alpha**=**0.5))
366. axs[i].set\_xlim(x\_min, x\_max)
367. axs[i].set\_ylim(y\_min, y\_max)
369. **if** appr\_type **==** self.best\_approximation:
370. axs[i].set\_facecolor('#e0ffe0')
371. axs[i].set\_title(f"{type\_names.get(appr\_type, appr\_type)}: НАИЛУЧШАЯ! RMSE={data['rmse']:.4f}")
372. **except** Exception as e:
373. print(f"Ошибка при построении графика для {appr\_type}: {e}")
375. fig.tight\_layout()
377. **if** save\_to\_file:
378. plt.savefig(save\_to\_file, dpi**=**300)
379. print(f"Графики сохранены в файл {save\_to\_file}")
380. **else**:
381. plt.show()
383. **def** main():
384. approx **=** ApproximationMethod()
385. **while** True:
386. source **=** input("Выберите источник данных (1 - консоль, 2 - файл): ")
387. **if** source **==** '1':
388. approx.input\_data('console')
389. **break**
390. **elif** source **==** '2':
391. filename **=** input("Введите имя файла с данными: ")
392. approx.input\_data(filename)
393. **break**
394. **else**:
395. print("Неверный выбор. Выберите 1 или 2.")
397. **try**:
398. approx.approximate()
399. **except** Exception as e:
400. print(f"Ошибка при аппроксимации: {e}")
401. **return**
403. **while** True:
404. output **=** input("Выберите способ вывода результатов (1 - консоль, 2 - файл, 3 - оба): ")
405. **if** output **==** '1':
406. approx.print\_results('console')
407. **break**
408. **elif** output **==** '2':
409. **while** True:
410. **try**:
411. filename **=** input("Введите имя файла для сохранения результатов: ")
412. approx.print\_results(filename)
413. **break**
414. **except** Exception as e:
415. print(f"Ошибка при сохранении результатов: {e}. Попробуйте другое имя файла.")
416. **break**
417. **elif** output **==** '3':
418. **while** True:
419. **try**:
420. filename **=** input("Введите имя файла для сохранения результатов: ")
421. approx.print\_results('console')
422. approx.print\_results(filename)
423. **break**
424. **except** Exception as e:
425. print(f"Ошибка при сохранении результатов: {e}. Попробуйте другое имя файла.")
426. **break**
427. **else**:
428. print("Неверный выбор. Выберите 1, 2 или 3.")
430. **while** True:
431. graph\_option **=** input("Построить графики? (y/n): ").lower()
432. **if** graph\_option **in** ['y', 'yes', 'да', 'д']:
433. **while** True:
434. save\_option **=** input("Сохранить графики в файл? (y/n): ").lower()
435. **if** save\_option **in** ['y', 'yes', 'да', 'д']:
436. **while** True:
437. **try**:
438. image\_file **=** input("Введите имя файла для сохранения графиков: ")
439. approx.plot\_graphs(save\_to\_file**=**image\_file)
440. **break**
441. **except** Exception as e:
442. print(f"Ошибка при сохранении графиков: {e}. Попробуйте другое имя файла.")
443. **break**
444. **elif** save\_option **in** ['n', 'no', 'нет', 'н']:
445. **try**:
446. approx.plot\_graphs()
447. **except** Exception as e:
448. print(f"Ошибка при построении графиков: {e}")
449. **break**
450. **else**:
451. print("Неверный выбор. Введите 'y' или 'n'.")
452. **break**
453. **elif** graph\_option **in** ['n', 'no', 'нет', 'н']:
454. **break**
455. **else**:
456. print("Неверный выбор. Введите 'y' или 'n'.")
458. **if** \_\_name\_\_ **==** "\_\_main\_\_":
459. **try**:
460. main()
461. **except** Exception as e:
462. print(f"Произошла ошибка: {e}")
463. sys.exit(1)