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 №2 по дисциплине «Вычислительная математика»

Вариант 13

Выполнил:

Студент группы P3212

Метель Леонард Валерьевич

Преподаватель:

г. Санкт-Петербург

2025

**Цель работы:**

Изучить численные методы решения нелинейных уравнений и их

систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений,

выполнить программную реализацию методов

**Вычислительная реализация задачи:**

Часть 1



Изображение выглядит как линия, График, диаграмма, Параллельный

Автоматически созданное описание

Корни: -7.292; 0.345; 2.137

Крайний правый корень: Метод простой итерации

Описание метода:

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Таблица:

Интервал - [2, 3]; x0= 2;

Проверка сходимости:

Метод простой итерации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № итерации | xk | xk+1 | F(xk+1) | | xk + 1 - xk | |
| 1 | 2 | 2,055 | 2,089 | 0,055 |
| 2 | 2,055 | 2,089 | 2,109 | 0,034 |
| 3 | 2,089 | 2,109 | 2,121 | 0,020 |
| 4 | 2,109 | 2,121 | 2,128 | 0,012 |
| 5 | 2,121 | 2,128 | 2,132 | 0,007 |

Крайний левый корень: Метод Хорд

Описание метода:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Таблица:

Интервал - [-8, -7]

*n = 3*

*x\* = -7,292*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № шага | a | x | F(a) | F(x) | |xk+1 - xk| |
| 1 | –8 | -7.247 | -59,82 | 3,271 | 0,247 |
| 2 | -8 | -7,286 | -59,82 | 0,497 | 0,019 |
| 3 | -8 | -7,291 | -59,82 | 0,137 | 0,005 |

Метод Хорд

Центральный корень: Метод Ньютона

Описание метода:

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Интервал: [0, 1]

Проверим условие сходимости:

F``(x) = 6x + 481/50  
Изображение выглядит как линия, диаграмма, График, Параллельный

Автоматически созданное описание

F`(x) = 3x2 + 481/50x – 1737/100

Изображение выглядит как линия, График, Параллельный, диаграмма

Автоматически созданное описание

Функции не меняют знак на рассматриваемом интервале

f(0) \* f```(0) = 51,75 > 0 – x0 = 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | xk | f(xk) | f `( xk) | xk+1 | | xk + 1 - xk | |
| 1 | 0 | 5,387 | -17,371 | 0,309 | 0,309 |
| 2 | 0,309 | 0,501 | -14,110 | 0,344 | 0,035 |
| 3 | 0,344 | 0,014 | -13,705 | 0,345 | 0,001 |

Метод Ньютона

2 Часть



Изображение выглядит как линия, диаграмма, График

Автоматически созданное описание

Решение:   
x = 1,142

y = -0,289

Описание метода:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Система:

2x + sin(y) = 2

y + cos(x - 1)= 0,7

Приводим к эквивалентному виду:

*y = 0,7 – cos(x-1) =*

Посчитаем частные производные:

Проверим сходимость:

1 < x < 1,5

-0,5 < y < 0

Процесс сходящийся

Таблица:

x = 1; y = -0.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № итерации | xk | xk+1 | xk+1-xk | yk | yk+1 | yk+1-yk |
| 1 | 1 | 1,099 | 0,099 | -0.2 | -0,3 | 0,1 |
| 2 | 1,099 | 1,147 | 0,048 | -0,3 | -0,295 | 0,005 |
| 3 | 1,147 | 1,145 | 0,002 | -0,295 | -0,289 | 0,006 |

Программная реализация



Код на python:

1. **import** math
2. **import** random
3. **import** numpy as np
4. **import** matplotlib.pyplot as plt
6. **def** r\_uravnenie(degree, min\_k, max\_k):
8. koefs **=** [random.randint(min\_k, max\_k) **for** \_ **in** range(degree **+** 1)]
9. **def** uravnenie(x):
10. **return** sum(koefs[i] **\*** x**\*\***i **for** i **in** range(len(koefs)))
11. **return** uravnenie, koefs

14. **def** plot\_function\_system(f\_system, x\_min, x\_max, y\_min, y\_max, root, title**=**"График системы с решением"):
15. **def** f1(x, y):
16. **return** f\_system([x, y])[0]
18. **def** f2(x, y):
19. **return** f\_system([x, y])[1]
21. X, Y **=** np.meshgrid(
22. np.linspace(x\_min, x\_max, 300),
23. np.linspace(y\_min, y\_max, 300)
24. )
26. Z1 **=** np.zeros\_like(X)
27. Z2 **=** np.zeros\_like(X)
29. **for** i **in** range(X.shape[0]):
30. **for** j **in** range(X.shape[1]):
31. Z1[i, j] **=** f1(X[i, j], Y[i, j])
32. Z2[i, j] **=** f2(X[i, j], Y[i, j])
34. plt.figure(figsize**=**(7, 6))
36. plt.contour(X, Y, Z1, levels**=**[0], colors**=**'red')
37. plt.contour(X, Y, Z2, levels**=**[0], colors**=**'blue')

40. x, y **=** root
41. plt.axhline(0, color**=**'black', linewidth**=**0.8)
42. plt.axvline(0, color**=**'black', linewidth**=**0.8)
44. plt.plot(x, y, 'go', label**=**f'Решение ({x:.4f}, {y:.4f})')
45. plt.text(x, y, f"({x:.4f}, {y:.4f})", fontsize**=**9, ha**=**'left', va**=**'bottom')
47. plt.grid(True)
48. plt.xlim(x\_min, x\_max)
49. plt.ylim(y\_min, y\_max)
50. plt.title(title)
51. plt.legend()
52. plt.show()

55. **def** p\_r\_uravnenie(koefs):
57. p\_koefs **=** []
58. **for** i **in** range(1, len(koefs)):
59. p\_koefs.append(i **\*** koefs[i])
60. **def** p\_uravnenie(x):
61. **return** sum(p\_koefs[i] **\*** x**\*\***i **for** i **in** range(len(p\_koefs)))
62. **return** p\_uravnenie, p\_koefs


66. **def** f\_example(x):
67. # return math.cos(2\*x) - 7 \* math.sin(x) - 4
68. **return** x**\*\***3 **-** x **+** 4
70. **def** df\_example(x):
71. # return -2 \* math.sin(2 \* x) - 7 \* math.cos(x)
72. **return** 3**\***x**\*\***2 **-** 1


76. **def** f\_system(vars\_xy):
77. x, y **=** vars\_xy
78. # return np.array([
79. #     math.cos(x - 1) + y - 0.5,
80. #     x - math.cos(y) - 3
81. # ])
82. **return** np.array([
83. x**\*\***2 **+** y**\*\***2 **-** 4,
84. y **-** 3**\***x**\*\***2
85. ])
87. **def** df\_system(vars\_xy):
88. x, y **=** vars\_xy
89. # return np.array([
90. #      [-math.sin(x - 1), 1],
91. #     [1, math.sin(y)]
92. # ])
93. **return** np.array([
94. [2**\***x, 2**\***y],
95. [**-**6**\***x, 1]
96. ])
98. **def** plot\_function(f, a, b,root,  title**=**"График функции"):
100. x\_vals **=** np.linspace(a, b, 400)
101. y\_vals **=** [f(x) **for** x **in** x\_vals]
102. plt.figure(figsize**=**(7, 4))
103. plt.plot(x\_vals, y\_vals, label**=**'f(x)')
104. plt.axhline(0, color**=**'black', linewidth**=**0.8)
106. y\_root **=** f(root)
107. plt.plot(root, y\_root, 'ro', label**=**f'Решение: x={root:.4f}')
108. plt.text(root, y\_root, f"({root:.4f}, {y\_root:.4f})", fontsize**=**9, ha**=**'left', va**=**'bottom')
110. plt.grid(True)
111. plt.xlim(a, b)
112. ymin, ymax **=** min(y\_vals), max(y\_vals)
113. plt.ylim(ymin **-** abs(ymin)**\***0.1, ymax **+** abs(ymax)**\***0.1)
114. plt.title(title)
115. plt.legend()
116. plt.show()

119. **def** bisection\_method(f, a, b, accuracy, m\_c):
120. **if** f(a) **\*** f(b) > 0:
121. **raise** ValueError("На отрезке [a, b] функция не меняет знак. Корень может отсутствовать или быть не единственным.")
123. count **=** 0
124. **while** count < int(m\_c):
125. count **+=** 1
126. x **=** (a **+** b) **/** 2
127. **if** abs(f(x)) <**=** accuracy **or** abs(b **-** a) <**=** accuracy:
128. **return** x, count, f(x)
130. **if** f(a) **\*** f(x) < 0:
131. b **=** x
132. **else**:
133. a **=** x
135. x **=** (a **+** b) **/** 2
136. **return** x, count, f(x)

139. **def** secant\_method(f, x0, x1, accuracy, m\_c):
140. count **=** 0
141. **for** \_ **in** range(int(m\_c)):
142. count **+=** 1
143. value\_in\_x0 **=** f(x0)
144. value\_in\_x1 **=** f(x1)
145. **if** abs(value\_in\_x1 **-** value\_in\_x0) < 1e**-**14:
146. **raise** ZeroDivisionError("Разница f(x1) и f(x0) слишком мала, деление невозможно.")
147. x2 **=** x1 **-** value\_in\_x1 **\*** (x1 **-** x0) **/** (value\_in\_x1 **-** value\_in\_x0)
148. **if** abs(x2 **-** x1) < accuracy:
149. **return** x2, count, f(x2)
150. x0, x1 **=** x1, x2
151. **return** x2, count, f(x2)

154. **def** iteration\_easy\_method(fi, x0, accuracy, m\_c):
155. count **=** 0
156. **for** \_ **in** range(int(m\_c)):
157. count **+=** 1
158. x1 **=** fi(x0)
159. **if** abs(x1 **-** x0) < accuracy:
160. **return** x1, count
161. x0 **=** x1
162. **return** x0, count

165. **def** compute\_alpha(df, a, b, steps**=**100):
166. mas\_val **=** np.linspace(a, b, steps)
167. max\_val **=** max(abs(df(x)) **for** x **in** mas\_val)
168. **if** (max\_val **==** 0):
169. max\_val **=** 1
170. **return** 1.0 **/** max\_val


174. **def** newton\_method\_system(fun, dfun, x0, accuracy, count):
175. x **=** np.array(x0, dtype**=**float)
176. **for** i **in** range(count):
177. F **=** fun(x)
178. dF **=** dfun(x)
179. **try**:
180. delta **=** np.linalg.solve(dF, F)
181. **except** np.linalg.LinAlgError:
182. **return** x, i
183. x\_new **=** x **-** delta

186. **if** abs(x\_new[0] **-** x[0]) <**=** accuracy **and** abs(x\_new[1] **-** x[1]) <**=** accuracy:
187. **return** x\_new, i **+** 1
188. x **=** x\_new
189. **return** x, count



194. **def** main():
195. print("Хотите решить ОДНОМЕРНОЕ уравнение или СИСТЕМУ?")
196. print("1) Одномерное уравнение")
197. print("2) Система (метод Ньютона)")
198. choice\_global **=** input("Выберите (1 или 2): ")
200. **if** choice\_global **==** '1':
201. print("1) Использовать случайное уравнение?")
202. print("2) Использовать функцию ?")
203. choice **=** input("Выберите (1 или 2): ")
205. **if** choice **==** '1':
206. deg **=** int(input("Введите степень случайного уравнения: "))
207. poly, coeffs **=** r\_uravnenie(deg, **-**5, 5)
208. print(f"Случайно сгенерированное уравнение (коэффициенты): {coeffs[::-1]}")
209. f **=** poly
210. dpoly, dcoeffs **=** p\_r\_uravnenie(coeffs)
211. df **=** dpoly
212. **else**:
213. f **=** f\_example
214. df **=** df\_example
216. print("Введите границы интервала (a b): ")
217. inp **=** input().split()
218. a **=** float(inp[0])
219. b **=** float(inp[1])
221. # plot\_function(f, a, b, title="График выбранной функции на интервале")
223. print("Введите требуемую точность: ")
224. accuracy **=** float(input())
226. print("Введите максимальное число итераций: ")
227. count **=** int(input())
229. print("Какой метод использовать?")
230. print("1) Метод половинного деления")
231. print("2) Метод секущих")
232. print("3) Метод простой итерации")
233. choice\_method **=** input("Выберите (1, 2, 3): ")
235. **if** choice\_method **==** '1':
236. **try**:
237. root, n\_iter, f\_val **=** bisection\_method(f, a, b, accuracy, count)
238. print(f"\nМетод половинного деления:\nНайденный корень: {root}\nКоличество итераций: {n\_iter}\nf(root) = {f\_val}")
239. plot\_function(f, a, b, root, title**=**"График выбранной функции на интервале")
240. **except** ValueError as e:
241. print(f"Ошибка: {e}")
243. **elif** choice\_method **==** '2':
244. **try**:
245. root, n\_iter, f\_val **=** secant\_method(f, a, b, accuracy, count)
246. print(f"\nМетод секущих:\nНайденный корень: {root}\nКоличество итераций: {n\_iter}\nf(root) = {f\_val}")
247. plot\_function(f, a, b, root, title**=**"График выбранной функции на интервале")
248. **except** ZeroDivisionError as e:
249. print(f"Ошибка: {e}")
251. **elif** choice\_method **==** '3':
252. alpha **=** compute\_alpha(df, a, b)
253. print(f"Вычислен alpha = {alpha}")
254. **def** fi(x):
255. **return** x **-** alpha **\*** f(x)
256. x0 **=** (a **+** b) **/** 2
258. root, n\_iter **=** iteration\_easy\_method(fi, x0, accuracy, count)
259. print(f"\nМетод простой итерации:\nНайденный корень: {root}\nКоличество итераций: {n\_iter}\nf(root) = {f(root)}")
260. plot\_function(f, a, b, root, title**=**"График выбранной функции на интервале")
261. **else**:
262. print("Неправильный выбор метода!")
264. **else**:
265. print("Решаем систему методом Ньютона.")
266. print("Для демонстрации возьмём систему со слайдов:\n  1) x^2 + y^2 = 4\n  2) y - 3x^2 = 0\n")
268. print("Введите начальное приближение (x0, y0): ")
269. inp **=** input().split()
270. x0 **=** float(inp[0])
271. y0 **=** float(inp[1])
273. print("Введите требуемую точность: ")
274. accuracy **=** float(input())
276. print("Введите максимальное число итераций: ")
277. count **=** int(input())
279. root, iters **=** newton\_method\_system(f\_system, df\_system, [x0, y0], accuracy, count)
280. **if** (root[0] **==** x0 **and** root[1] **==** y0) **and** (f\_system([x0, y0])[0] !**=** 0 **or** f\_system([x0, y0])[1] !**=** 0):
281. root, iters **=** newton\_method\_system(f\_system, df\_system, [x0 **+** accuracy, y0 **+** accuracy], accuracy, count)
282. print(f\_system([0, 0]))
283. print("\nМетод Ньютона")
284. print(f"Решения: x = {root[0]:.6f}, y = {root[1]:.6f}")
285. print(f"Число итераций: {iters}")
286. print(f"f1(x,y) = {f\_system(root)[0]:.6e}")
287. print(f"f2(x,y) = {f\_system(root)[1]:.6e}")
288. plot\_function\_system(f\_system, root[0] **-** 3, root[0] **+** 3, root[1] **-** 3, root[1]  **+** 3, root, title**=**"График системы с решением")

291. **if** \_\_name\_\_ **==** "\_\_main\_\_":
292. main()

# **Вывод**

В ходе выполнения данной лабораторной работы я познакомился с различными методами решения нелинейных уравнений и систем нелинейных уравнений и программно реализовал следующие:

**Метод половинного деления**

Достоинства:

идейная простота и надежность метода;

* Непритязательность к свойствам функции f (x) — она должна быть лишь непрерывной, а дифференцируемость не предполагается;
* Обладает абсолютной сходимостью.

Рекомендация: применять когда требуется высокая надежность

счета, а скорость несущественна.

Недостатки:

* Если интервал содержит несколько корней, то неизвестно к какому относится вычислительный процесс;
* Медленный метод: имеет линейную сходимость.

**Метод секущих**  
**Достоинства:**

* Более высокая скорость сходимости по сравнению с методом половинного деления.
* Не требует вычисления производной (в отличие от метода Ньютона).

**Недостатки:**

* Может расходиться, если начальные приближения выбраны неудачно.
* Скорость сходимости ниже, чем у метода Ньютона.

**Метод простой итерации**

Достоинства:

* Простота реализации

Недостатки:

* Cходимость в малой окрестности корня и вытекающая отсюда необходимость выбора начального приближения к корню из этой малой окрестности. В противном случае итерационный процесс расходится или сходится к другому корню этого уравнения. Если 𝜑 ′ 𝑥 ≈ 1, то сходимость может быть очень медленной.
* Преобразование исходного уравнения к виду: 𝑥 = 𝜑(𝑥)

#### **2. Метод Ньютона для решения системы нелинейных уравнений**

**Достоинства:**

* Высокая скорость сходимости (квадратичная).
* Эффективен для систем с гладкими функциями.

**Недостатки:**

* Требует вычисления матрицы Якоби (производных) на каждой итерации.
* Чувствителен к выбору начального приближения (может расходиться).
* Сложность реализации для систем высокой размерности.