

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

Мегафакультет компьютерных технологий и управления  
Факультет программной инженерии и компьютерной техники

## Отчёт

по лабораторной работе №2  
по дисциплине «Вычислительная математика»

Вариант №4

Группа: Р3207

Студент: Исмоилов Шахзод Бахтиёрович

Преподаватель: Рыбаков Степан Дмитриевич

Санкт-Петербург  
2025

# Содержание

<b>1</b>	<b>Цель лабораторной работы</b>	<b>2</b>
<b>2</b>	<b>Задание</b>	<b>2</b>
<b>3</b>	<b>Рабочие формулы использованных методов</b>	<b>2</b>
3.1	Метод Ньютона (касательных) . . . . .	2
3.2	Метод половинного деления . . . . .	2
3.3	Метод простой итерации . . . . .	2
<b>4</b>	<b>Графики функций</b>	<b>3</b>
4.1	Задание 1 . . . . .	3
4.2	Задание 2 . . . . .	3
<b>5</b>	<b>Заполненные таблицы вычислительной части №1</b>	<b>3</b>
5.1	Метод половинного деления для уточнения корня $x_1$ . . . . .	5
5.2	Метод секущих для уточнения корня $x_2$ . . . . .	5
5.3	Метод простой итерации для уточнения $x_3$ . . . . .	5
<b>6</b>	<b>Решение системы нелинейных уравнений вычислительной части №2</b>	<b>5</b>
<b>7</b>	<b>Листинг программы</b>	<b>6</b>
<b>8</b>	<b>Результаты выполнения программы</b>	<b>8</b>
<b>9</b>	<b>Вывод</b>	<b>9</b>

# 1 Цель лабораторной работы

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

## 2 Задание

Требования к программе:

Цель Все численные методы должны быть реализованы в виде отдельных подпроцессов/методов/классов. Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3–5 функций, в том числе трансцендентные), из тех, которые предлагает программа. Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность) из файла или с клавиатуры. Выполнить проверку исходных данных: проанализировать наличие корня на введённом интервале. При отсутствии корней выводить сообщение об ошибке. Для методов, требующих начального приближения (Ньютона, секущих, хорд с фиксированным концом, простой итерации) выбор начального приближения и проверку условия сходимости выполнять в программе. Предусмотреть вывод результатов (найденный корень, значение функции в корне, число итераций) в файл или на экран. Организовать вывод графика функции для всего исследуемого интервала (с запасом).

## 3 Рабочие формулы использованных методов

### 3.1 Метод Ньютона (касательных)

Пусть  $y = f(x)$  на отрезке  $[a, b]$ . Инициализация:  $x_0$  на отрезке  $[a, b]$ . В точке  $x_{i-1}$  уравнение касательной:

$$y = f(x_{i-1}) + f'(x_{i-1})(x - x_{i-1}).$$

Рабочая формула метода:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}.$$

Критерий остановки:  $|x_n - x_{n-1}| \leq \varepsilon$ .

### 3.2 Метод половинного деления

Метод заключается в делении интервала пополам. На концах интервала  $[a, b]$  функция имеет разные знаки. Рабочая формула:

$$x_i = \frac{a_i + b_i}{2}.$$

Критерий остановки:  $|x_n - x_{n-1}| \leq \varepsilon$ .

### 3.3 Метод простой итерации

Переписываем  $f(x) = 0$  в виде  $x = \phi(x)$ . Инициализация  $x_0$  на отрезке  $[a, b]$ . Рабочая формула:

$$x_{i+1} = \phi(x_i).$$

Условие сходимости:  $|\phi'(x)| \leq q < 1$  на  $[a, b]$ . Критерий остановки:  $|x_n - x_{n-1}| \leq \varepsilon$ .

## 4 Графики функций

### 4.1 Задание 1

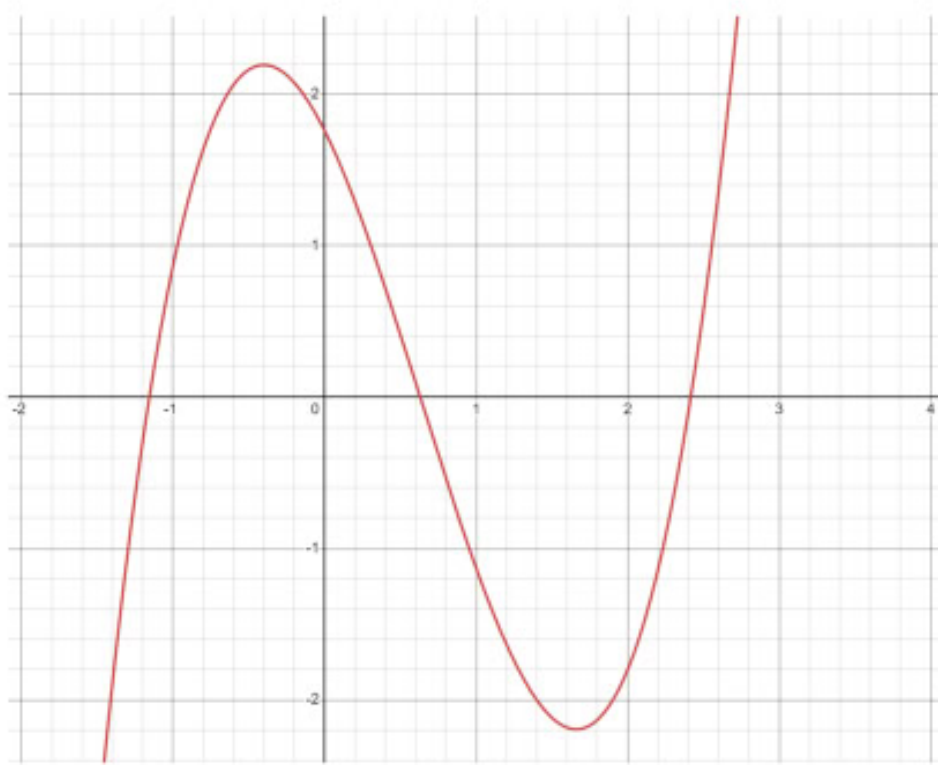


Рис. 1: График функции  $x^3 - 1.89x^2 - 2x + 1.76$ .

### 4.2 Задание 2

## 5 Заполненные таблицы вычислительной части №1

Для функции  $x^3 - 1.89x^2 - 2x + 1.76$ :

1. Графически отделены корни:  $x_1 = -1.156$ ,  $x_2 = 0.63$ ,  $x_3 = 2.416$ .

2. Интервалы изоляции:

- $[-2, 0]$  для  $x_1$
- $[0, 2]$  для  $x_2$
- $[2, 4]$  для  $x_3$

3. Точность  $\varepsilon = 0.01$ .

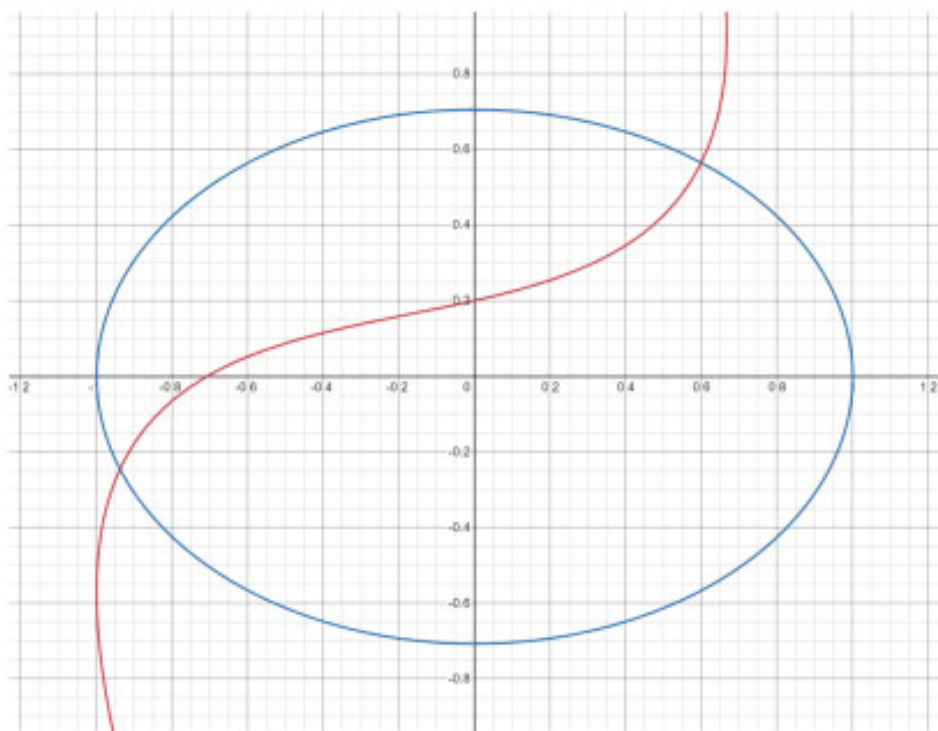


Рис. 2: График системы уравнений  $\sin(x + y) - 1.2x = 0.2$  и  $x^2 + 2y^2 = 1$ .

№	$a$	$b$	$x$	$f(a)$	$f(b)$	$f(x)$	$ a - b $
1	-2	0	-1.000	-9.800	1.760	0.870	2
2	-2	-1	-1.500	-9.800	0.870	-2.867	1
3	-1.5	-1	-1.250	-2.867	0.870	-0.646	0.5
4	-1.25	-1	-1.125	-0.646	0.870	0.194	0.25
5	-1.25	-1.125	-1.188	-0.646	0.194	-0.208	0.125
6	-1.188	-1.125	-1.157	-0.208	0.194	-0.005	0.063
7	-1.157	-1.125	-1.141	-0.005	0.194	0.096	0.032
8	-1.157	-1.141	-1.149	-0.005	0.096	0.046	0.016
9	-1.157	-1.149	<b>-1.153</b>	-0.005	0.046	0.021	$0.008 < \varepsilon$

№	$x_{k-1}$	$x_k$	$x_{k+1}$	$f(x_{k+1})$	$ x_{k+1} - x_k $
1	0	1	0.609	0.06691	0.391
2	1	0.609	<b>0.63086</b>	-0.00282	0.02186
3	0.609	0.63086	<b>0.62997</b>	0	$0.00088 < \varepsilon$

№	$x_k$	$x_{k+1}$	$f(x_{k+1})$	$ x_{k+1} - x_k $
1	2.000	2.140	-1.375	0.140
2	2.140	2.236	-0.983	0.096
3	2.236	2.299	-0.674	0.064
4	2.299	2.341	-0.449	0.042
5	2.341	2.368	-0.294	0.027
6	2.368	2.386	-0.191	0.017
7	2.386	2.397	-0.123	0.011
8	2.397	<b>2.404</b>	-0.079	$0.007 < \varepsilon$

**5.1** Метод половинного деления для уточнения корня  $x_1$

**5.2** Метод секущих для уточнения корня  $x_2$

**5.3** Метод простой итерации для уточнения  $x_3$

## **6** Решение системы нелинейных уравнений вычислительной части №2

### **Левый корень**

Начальное приближение:  $x_0 = -0.94$ ,  $y_0 = -0.25$ .

**Шаг 1.** Запишем систему для  $\Delta x, \Delta y$ :

$$\begin{cases} (\cos(x+y) - \frac{6}{5}) \Delta x + \cos(x+y) \Delta y = \frac{x \cdot 6}{5} + \frac{1}{5} - \sin(x+y), \\ (2x) \Delta x + (4y) \Delta y = -2y^2 + 1 - x^2. \end{cases}$$

На первой итерации система принимает вид:

$$\begin{cases} -0.82834\Delta x + 0.37166\Delta y = 0.00037, \\ -1.88\Delta x - 1\Delta y = -0.0086. \end{cases}$$

**Шаг 2.** Решаем линейную систему: получаем  $\Delta x = -0.06191$ ,  $\Delta y = -0.26827$ .

**Шаг 3.** Вычисляем новые приближения:

$$x_1 = x_0 + \Delta x = -1.00191, \quad y_1 = y_0 + \Delta y = -0.26827.$$

**Шаг 4.** Проверяем условие окончания при  $\varepsilon = 0.01$ :

$$|x_1 - x_0| \leq \varepsilon, \quad |y_1 - y_0| > \varepsilon,$$

продолжаем итерации (до *Шага 12*), после чего получаем:

$$x_3 = -0.93813, \quad y_3 = -0.24489.$$

### **Правый корень**

Начальное приближение:  $x_0 = 0.6$ ,  $y_0 = 0.5$ . **Шаг 1.** Выбираем  $x_0 = 0.6$ ,  $y_0 = 0.5$ . Исходная система:

$$\begin{cases} (\cos(x+y) - \frac{6}{5}) \Delta x + (\cos(x+y)) \Delta y = \frac{x \cdot 6}{5} + \frac{1}{5} - \sin(x+y), \\ (2 \cdot x) \Delta x + (4 \cdot y) \Delta y = -2y^2 + 1 - x^2. \end{cases}$$

**Шаг 2.** Решаем полученную систему. Получаем  $\Delta x = 0.00291$  и  $\Delta y = 0.06826$ . **Шаг 3.** Вычисляем очередные приближения:

$$x_1 = x_0 + \Delta x = 0.6 + 0.00291 = 0.60291, \quad y_1 = y_0 + \Delta y = 0.5 + 0.06826 = 0.56826.$$

**Шаг 4.** Проверяем критерии окончания при  $\varepsilon = 0.01$ :

$$|x_1 - x_0| \leq \varepsilon \quad |y_1 - y_0| \leq \varepsilon, \quad |0.60291 - 0.6| \leq \varepsilon \quad |0.56826 - 0.5| > \varepsilon.$$

**Шаг 5.** Подставляем очередные приближения в систему:

$$\begin{cases} -0.81092\Delta x + 0.38908\Delta y = 0.00228, \\ 1.20581\Delta x + 2.27303\Delta y = -0.00933. \end{cases}$$

**Шаг 6.** Решаем полученную систему. Получаем  $\Delta x = -0.00381$  и  $\Delta y = -0.00208$ . **Шаг 7.** Вычисляем очередные приближения:

$$x_2 = x_1 + \Delta x = 0.60291 - 0.00381 = 0.59909, \quad y_2 = y_1 + \Delta y = 0.56826 - 0.00208 = 0.56618.$$

**Шаг 8.** Проверяем критерии окончания при  $\varepsilon = 0.01$ :

$$|x_2 - x_1| \leq \varepsilon \quad |y_2 - y_1| \leq \varepsilon, \quad |0.59909 - 0.60291| \leq \varepsilon \quad |0.56618 - 0.56826| \leq \varepsilon.$$

## 7 Листинг программы

### Метод половинного деления

Листинг 1: Листинг: Метод половинного деления

```
1 def half_method(quation, method):
2     a, b, inaccuracy = input_selection(quation, method)
3
4     iterations = 1
5     max_iter = 10000
6
7     if (float(b) - float(a) - 0.00001 < 0):
8         print("
9             exit()
10
11     count_roots = count_roots_on_interval(quation, a, b, 0.001)
12     if not validate_roots(count_roots):
13         exit()
14
15     a1, b1 = a, b
16     iterations = 1
17     max_iter = 1000
18     while (b - a) / 2 > inaccuracy and iterations < max_iter:
19         midpoint = (a + b) / 2
20         if quation_solution(quation, midpoint) == 0:
21             return midpoint, iterations
22         elif quation_solution(quation, a) * quation_solution(quation,
23             midpoint) < 0:
24             b = midpoint
25         else:
26             a = midpoint
27             iterations += 1
28     if iterations >= max_iter - 1:
29         print("
30         exit()
31
32     solution = try_to_convert_to_int((a + b) / 2)
33     print(f"
34     print(f"f(
35     print(f"
36     output_data(solution, quation_solution(quation, solution),
37         iterations, str_quation(quation))
```

```

36 draw_graphth(quation, str_quation(quation), a1, b1, solution,
    quation_solution(quation, solution))

```

## Метод Ньютона

Листинг 2: Листинг: Метод Ньютона

```

1 def Newton_method(quation, method):
2     a, b, inaccuracy = input_selection(quation, method)
3
4     approximation = validate_initial_approximation(quation, a, b)
5
6     iterations = 1
7     max_iter = 1000
8     x = approximation
9     while abs(quation_solution(quation, x)) > inaccuracy and iterations
10         < max_iter:
11         x = x - quation_solution(quation, x) / quation_df_solution(
12             quation, x)
13         iterations += 1
14
15     solution = try_to_convert_to_int(x)
16     output_data(solution, quation_solution(quation, solution),
17         iterations, str_quation(quation))
18     draw_graphth(quation, str_quation(quation), a, b, solution,
19         quation_solution(quation, solution))

```

## Метод простой итерации (уравнения)

Листинг 3: Листинг: Метод простой итерации

```

1 def Simple_iteration_method(quation, method):
2     a, b, inaccuary = input_selection(quation, method)
3     q = check_convergencecondition(quation, a, b)
4     approximation = validate_initial_approximation(quation, a, b)
5     x = approximation
6     iterations = 1
7     max_iter = 1000
8     if q > 1:
9         print("
10             ")
11         exit()
12     elif 0 < q <= 0.5:
13         while abs(converted_quation(quation, x) - x) > inaccuary and
14             iterations < max_iter:
15             x = converted_quation(quation, x)
16             iterations += 1
17     else:
18         while abs(converted_quation(quation, x) - x) > ((1 - q) / q) *
19             inaccuary and iterations < max_iter:
20             x = converted_quation(quation, x)
21             iterations += 1

```



```

19
20     solution = try_to_convert_to_int(x)
21     output_data(solution, quation_solution(quation, solution),
22                 iterations, str_quation(quation))
23     draw_graphth(quation, str_quation(quation), a, b, solution,
24                 quation_solution(quation, solution))

```

## Метод простой итерации (системы)

Листинг 4: Листинг: Метод простой итерации для систем

```

1  def simple_iteration_method(system, method):
2      x0 = choose_x()
3      y0 = choose_y()
4      inaccuracy = choose_inaccuracy()
5      max_iter = 1000
6      iteration = 1
7
8      if not check_convergence(system, method, x0, y0):
9          print("
10
11              .")
12
13          return
14
15      for _ in range(max_iter):
16          x = f1(x0, y0, system)
17          y = f2(x0, y0, system)
18          if abs(x - x0) < inaccuracy and abs(y - y0) < inaccuracy:
19              print(f"x = {x}, y = {y}, iterations = {iteration}")
20              break
21          x0, y0 = x, y
22          iteration += 1
23      output_data(x0, y0, f1(x, y, system), iteration)

```

Ссылка на проект: <https://github.com/Djerdn/computation-math/tree/main/lab2>

## 8 Результаты выполнения программы

Выберите вариант:

- 1) Нелинейное уравнение
- 2) Система нелинейных уравнений

1

Выберите уравнение:

- 1)  $x^2 - 3x + 2$
- 2)  $x^3 + 2x^2 - 5$
- 3)  $\cos(x) - x^2$

1

Выберите метод:

- 1) Метод половинного деления
- 2) Метод Ньютона
- 3) Метод простой итерации

```

1
Выберите ввод:
1) Ввод вручную
2) Ввод из файла
1
Введите левую границу интервала: 1
Введите правую границу интервала: 5
Введите точность: 0.01
Ответ = 4.9921875
f(ответ) = 11.94537353515625
Итерации = 9
Выберите формат вывода:
1) В консоль
2) В файл
1
Уравнение:  $x^2 - 3x + 2$ 
Решение: 4.9921875
f(4.9921875) = 11.94537353515625
Итерации: 9

```

Также в интерфейсе matplotlib отображается график решения (пример):

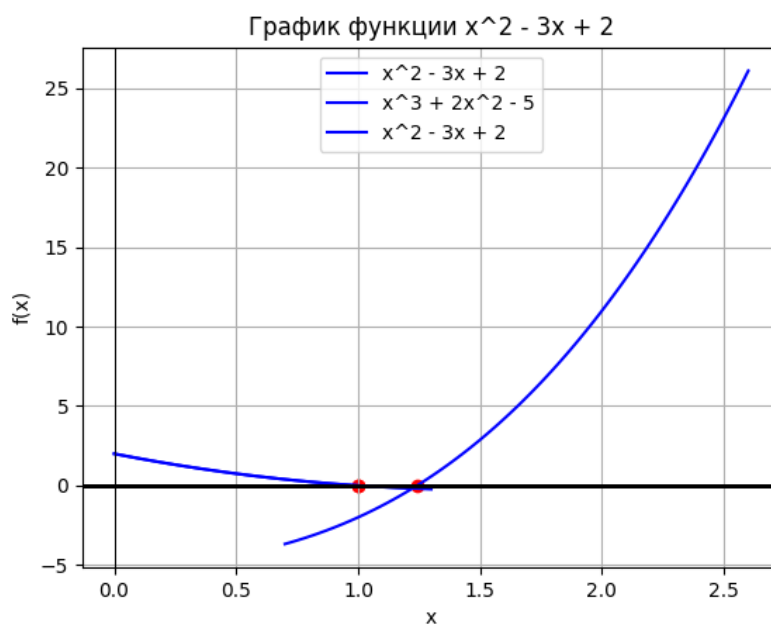


Рис. 3: График функции  $x^2 - 3x + 2$  с найденным корнем.

## 9 Вывод

В работе были рассмотрены и реализованы три метода поиска корней нелинейных уравнений:

- Метод половинного деления (дихотомии).
- Метод Ньютона (касательных).

- Метод простой итерации.

Для каждого метода были проведены следующие этапы:

1. Выбор и проверка исходных данных (интервала или начального приближения).
2. Итерационный процесс до достижения заданной точности  $\varepsilon$ .
3. Вывод найденного корня, значения функции в корне и числа итераций.
4. Построение графика функции с выделенным корнем.

Сравнительный анализ методов:

- Метод дихотомии гарантированно сходится при наличии корня на интервале, однако скорость сходимости линейная:  $n \approx \log_2 \frac{b-a}{\varepsilon}$ .
- Метод Ньютона обладает квадратичной скоростью сходимости при выполнении условия дифференцируемости функции и правильного выбора начального приближения.
- Метод простой итерации имеет скорость сходимости, зависящую от коэффициента Липшица  $q < 1$ .

Работа демонстрирует практическую реализацию численных методов и их сравнение по критериям сходимости и точности.