«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия  
Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №2

Вариант 4

Студент:

*Ильин Н. С.*

*Р3210*

Преподаватель:

*Наумова Н. А.*

Санкт-Петербург, 2025 г.

**Оглавление**

[**Цель работы**: 3](#_Toc192517794)

[**Рабочие формулы:** 3](#_Toc192517795)

[**Листинг программы** 15](#_Toc192517796)

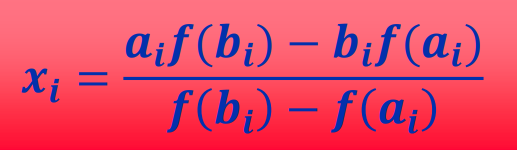
[**Выводы**: 16](#_Toc192517797)

# **Цель работы**:

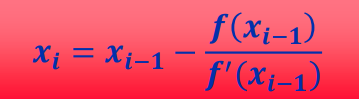
Изучить численные методы решения систем линейных алгебраических уравнений и реализовать программу для решения СЛАУ методом простых итераций.

**Рабочие формулы:**

Рабочая формула метода хорд:



Рабочая формула метода Ньютона:



Рабочая формула метода простой итерации:

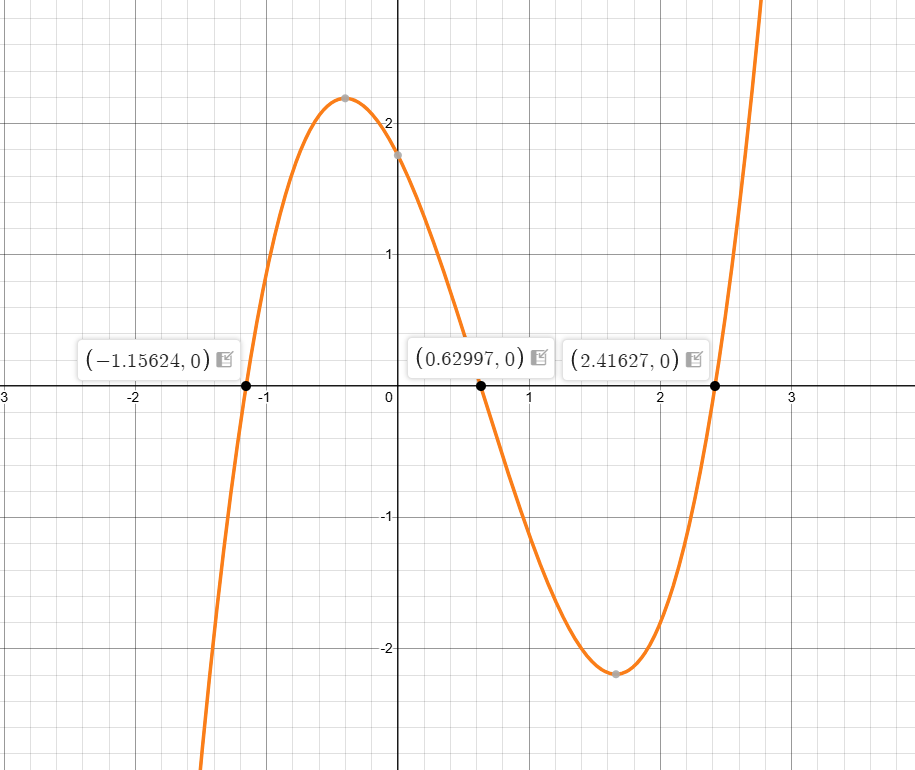


**1 Вычислительная реализация задачи:**

**1 часть. Решение нелинейного уравнения**

|  |
| --- |
| x3-1,89x2-2x+1,76 |

**1)**



**2.**

Для определения интервалов изоляции корней данного уравнения, можно воспользоваться методом интервалов знакопеременности. Для этого нужно найти значения функции на различных интервалах и определить знак функции на каждом из них.

Получим приближенные значения корней:  
x ≈ -1.2, x ≈ 0.6, x ≈ 2.4

Теперь нужно разбить ось x на 4 интервала: (-∞, -1.2), (-1.2, 0.6), (0.6, 2.4) и (2.4, +∞). На каждом из этих интервалов нужно определить знак функции.

Знаки функции на каждом интервале будут соответственно

|  |  |  |  |
| --- | --- | --- | --- |
| (-∞, -3.9) | (-3.9, -1.5) | (-1.5, 1.4) | (1.4, +∞) |
| - | + | - | + |

1. **Первый корень** находится в интервале (−2,−1).
   * Функция меняет знак с отрицательного на положительный:  
     f(-2)=−9,8<0  
     f(-1)=0,87>0
2. **Второй корень** находится в интервале (0,1).
   * Функция меняет знак с положительного на отрицательный:  
     f(0)=1,76>0f(1)=−1,13<0
3. **Третий корень** находится в интервале (2,3).
   * Функция меняет знак с отрицательного на положительный:  
     f(2)=−1,8<0  
     f(3)=5,75>0

Таким образом, мы получаем три интервала изоляции корней уравнения:

(-2, -1), (0, 1) и (2, 3).

**3.**

x1 ≈   
x2 ≈   
x3 ≈

**4.**

Крайний правый корень – **Метод простой итерации**

Проверка **условия сходимости** метода на выбранном интервале

**Крайний правый корень – Метод простой итерации**  
**Проверка условия сходимости на интервале [2, 3]:**

**Шаг 1: Преобразование к виду x=ϕ(x)**

Выделим x из уравнения:

**Шаг 2: Проверка условия сходимости**  
Производная ϕ(x):

На интервале **[2, 3]**:

* ϕ′(2)=-0.5+0.44=-0.06
* ϕ′(3)=-0.222+0.130=-0.092

Максимальный модуль производной:

**Условие сходимости выполняется** (q=0.092), метод применим.  
Начальное приближение: x0=2.5  
Критерий остановки:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | xk | xk+1 | f(xk+1) | | xk+1- xk| |
| 1 | 2.500 | 2.408 | -0.050 | 0.092 |
| 2 | 2.408 | 2.417 | 0.005 | 0.009 |
| 3 | 2.417 | **2.416** | -0.000 | 0.001 |

Корень: x ≈ **2.416**

Крайний левый корень – Метод половинного деления

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | a | b | x | f(a) | f(b) | f(x) | |a – b| |
| 1 | -2.000 | -1.000 | -1.500 | -9.800 | 0.870 | -2.867 | 1.000 |
| 2 | -1.500 | -1.000 | -1.250 | -2.867 | 0.870 | -0.646 | 0.500 |
| 3 | -1.250 | -1.000 | -1.125 | -0.646 | 0.870 | 0.194 | 0.250 |
| 4 | -1.250 | -1.125 | -1.188 | -0.646 | 0.194 | -0.205 | 0.125 |
| 5 | -1.188 | -1.125 | -1.156 | -0.205 | 0.194 | -0.000 | 0.062 |
| 6 | -1.156 | -1.125 | -1.141 | -0.000 | 0.194 | 0.098 | 0.031 |
| 7 | -1.156 | -1.141 | -1.148 | -0.000 | 0.098 | 0.049 | 0.016 |
| 8 | -1.156 | -1.148 | -1.152 | -0.000 | 0.049 | 0.025 | 0.008 |
| 9 | -1.156 | -1.152 | -1.154 | -0.000 | 0.025 | 0.012 | 0.004 |
| 10 | -1.156 | -1.154 | -1.155 | -0.000 | 0.012 | 0.006 | 0.002 |

Корень: x ≈ **-1.155**

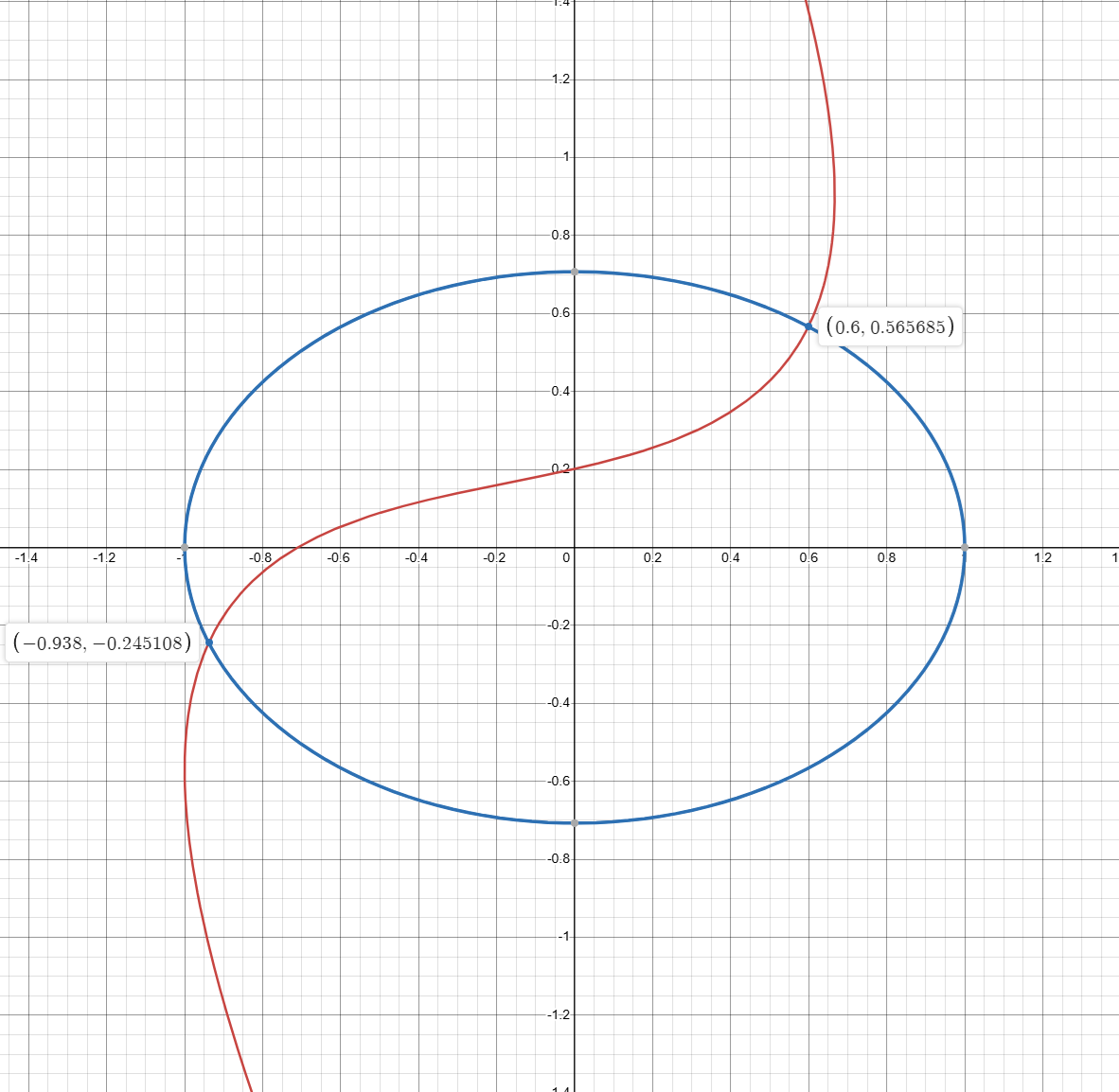
Центральный корень – **Метод секущих**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | xk-1 | xk | xk+1 | f(xk+1) | |xk+1 - xk| |
| 1 | 0.000 | 1.000 | 0.609 | 0.067 | 0.391 |
| 2 | 1.000 | 0.609 | 0.631 | -0.003 | 0.022 |
| 3 | 0.609 | 0.631 | **0.630** | 0.000 | 0.001 |

Корень: x ≈ **0.630**

**2. Решение системы нелинейных уравнений**

Метод Ньютона



**2.**

Решением системы уравнений являются точки пересечения эллипса и  
, следовательно, система имеет не более двух различных решений.

*, , ,*

**Корень 1:** Шаг 1: Выбираем

Шаг 2. Решаем полученную систему.

Шаг 3. Вычисляем приближения:

Итерация 1:

Итерация 2:

Итерация 3:

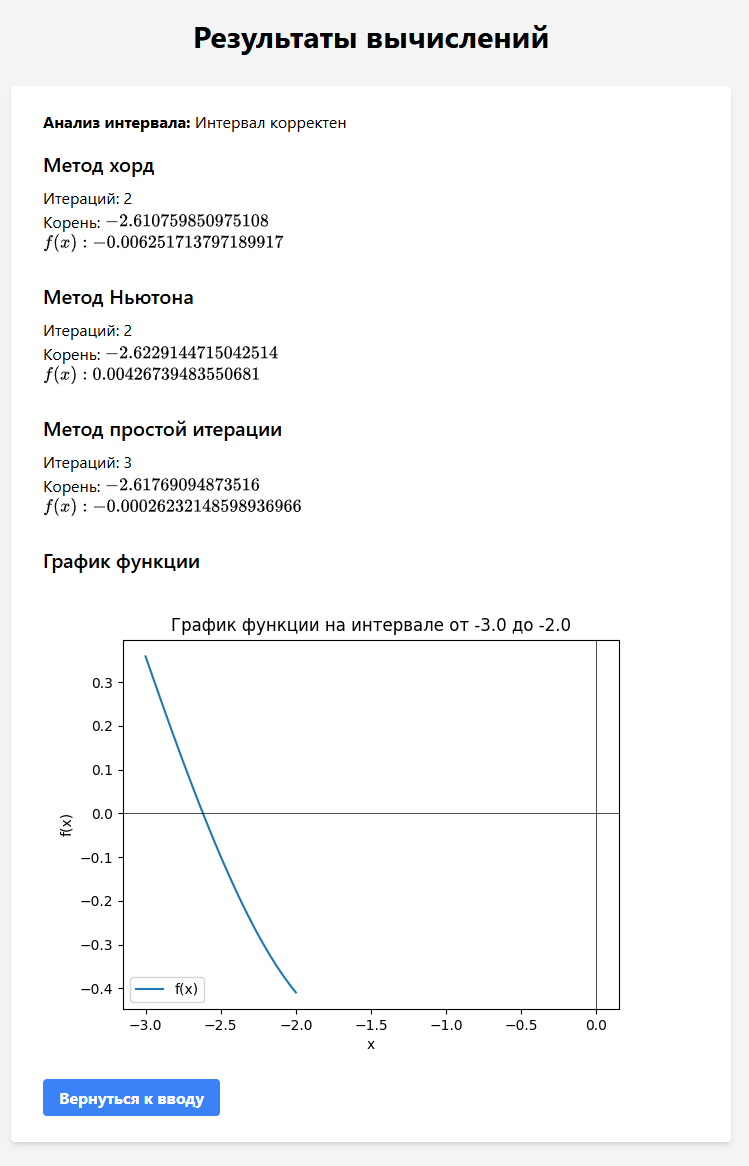
Корень:

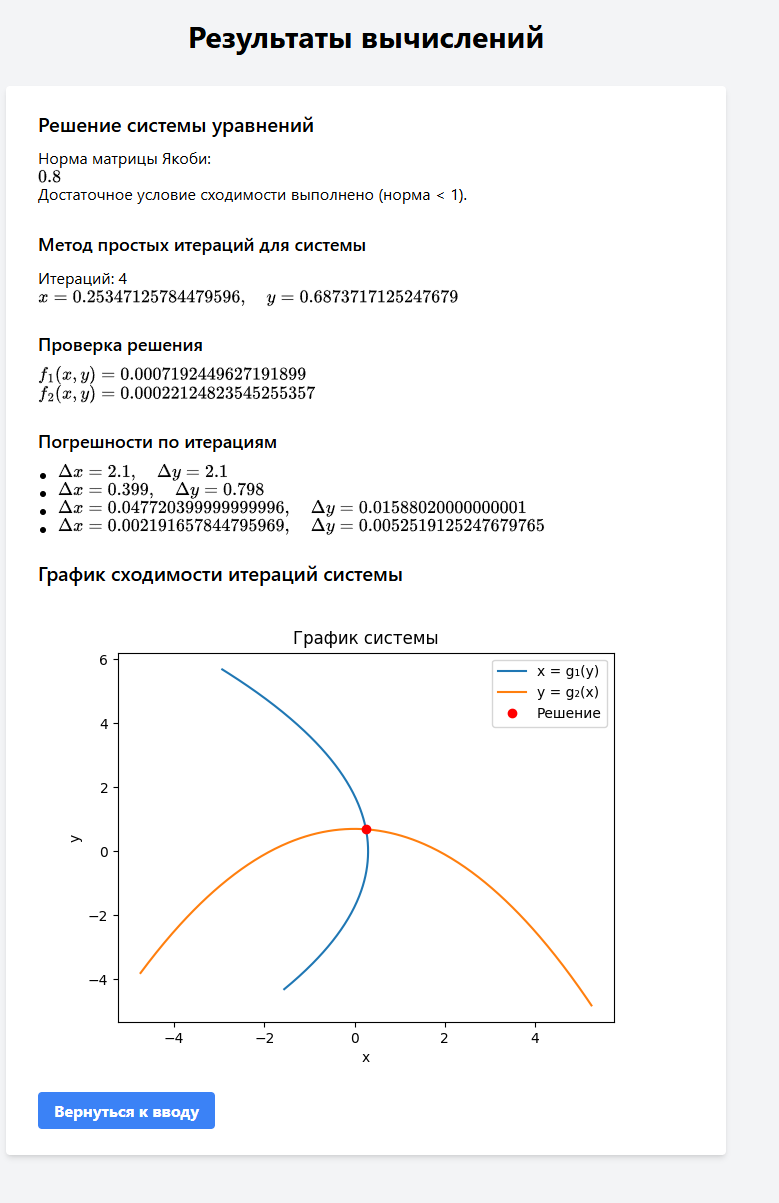
Аналогично находим **другой корень**: (−0.9387, −0.2461)

**2. Программная реализация задачи**

Github: https://github.com/MrTheFall/computational\_math/tree/main/lab2

**Результаты выполнения программы при различных исходных данных:**

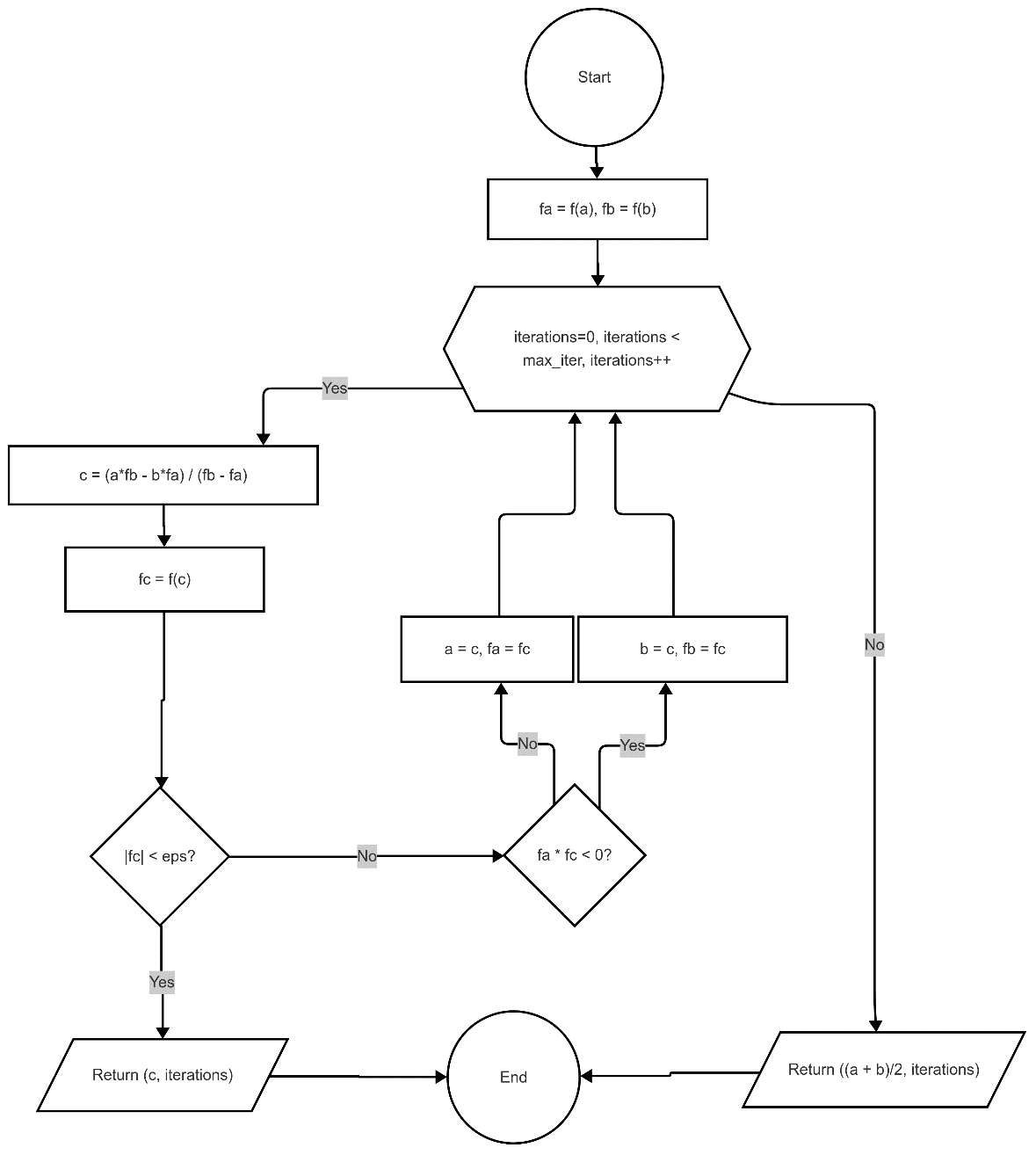




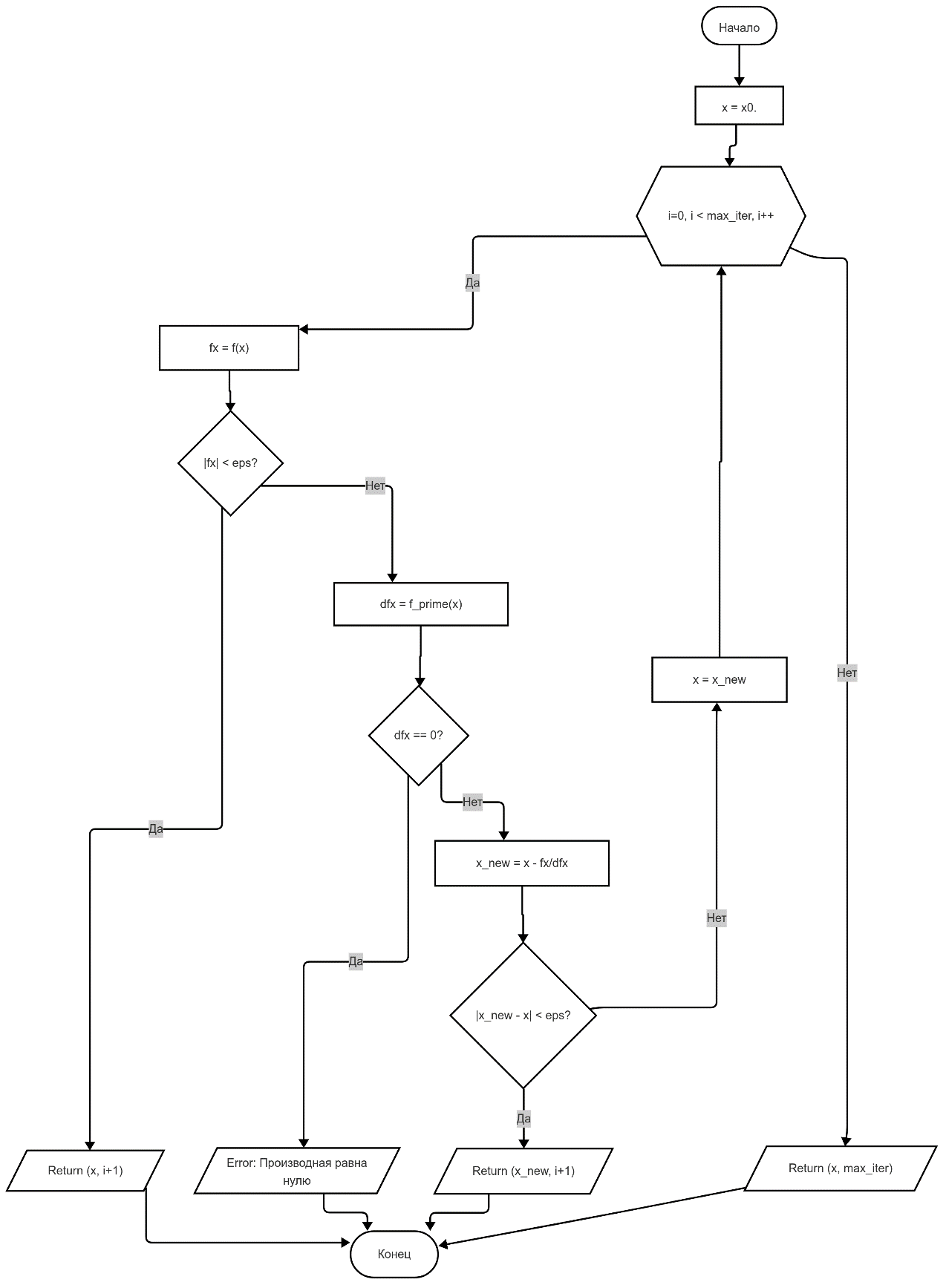
**3. Блок схемы реализуемых методов**

Нелинейные уравнения:

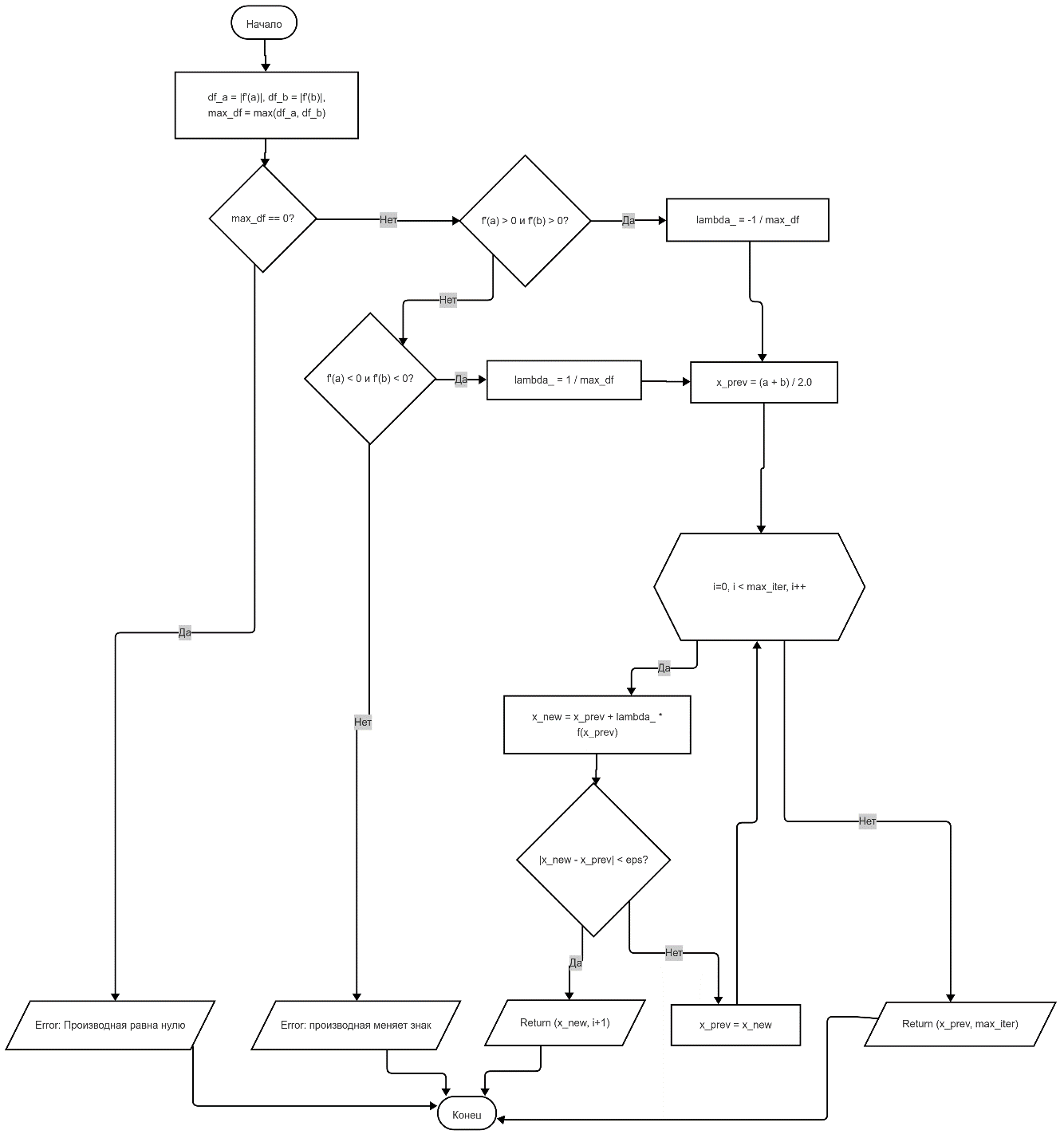
Метод Хорд:



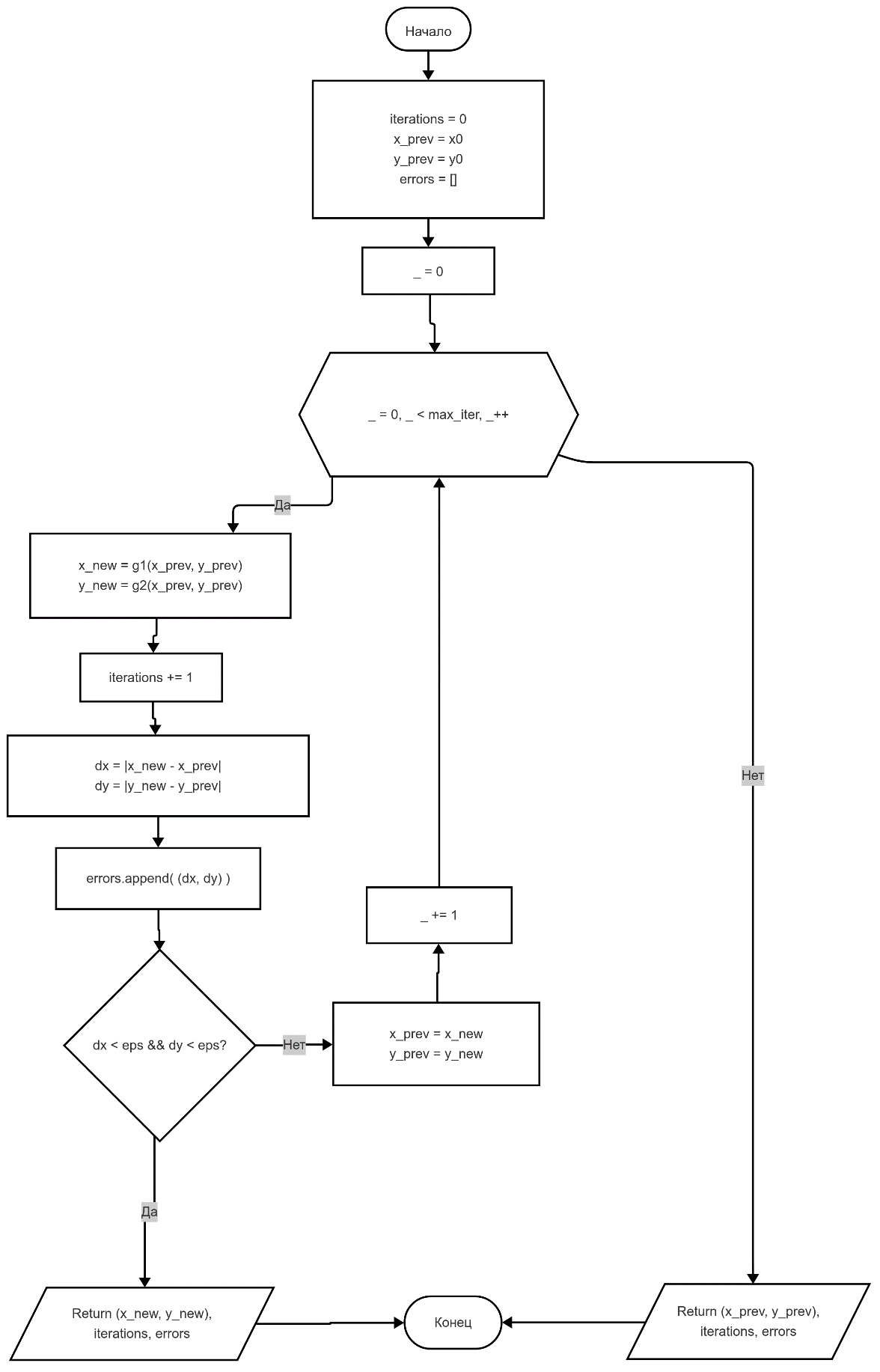
Метод Ньютона:



Метод простых итераций:



Метод простых итераций (Система)



# **Листинг программы**

|  |
| --- |
| def chord\_method(f, a, b, eps, max\_iter=1000):  fa, fb = f(a), f(b)  iterations = 0  for \_ in range(max\_iter):  c = (a \* fb - b \* fa) / (fb - fa)  fc = f(c)  iterations += 1  if abs(fc) < eps:  return c, iterations  if fa \* fc < 0:  b, fb = c, fc  else:  a, fa = c, fc  return (a + b) / 2, iterations  def newton\_method(f, f\_prime, x0, eps, max\_iter=1000):  x = x0  for i in range(max\_iter):  fx = f(x)  if abs(fx) < eps:  return x, i + 1  dfx = f\_prime(x)  if dfx == 0:  raise ValueError("Производная равна нулю")  x\_new = x - fx / dfx  if abs(x\_new - x) < eps:  return x\_new, i + 1  x = x\_new  return x, max\_iter  def simple\_iteration\_method(f, f\_prime, a, b, eps, max\_iter=1000):  df\_a = abs(f\_prime(a))  df\_b = abs(f\_prime(b))  max\_df = max(df\_a, df\_b)    if max\_df == 0:  raise ValueError("Производная на интервале равна нулю, метод неприменим")  if f\_prime(a) > 0 and f\_prime(b) > 0:  lambda\_ = -1 / max\_df  elif f\_prime(a) < 0 and f\_prime(b) < 0:  lambda\_ = 1 / max\_df  else:  raise ValueError("Производная меняет знак на интервале, метод может не сходиться")  x\_prev = (a + b) / 2.0  for i in range(max\_iter):  x\_new = x\_prev + lambda\_ \* f(x\_prev)  if abs(x\_new - x\_prev) < eps:  return x\_new, i + 1  x\_prev = x\_new    return x\_prev, max\_iter  def simple\_iteration\_system(g1, g2, x0, y0, eps, max\_iter=1000):  iterations = 0  x\_prev, y\_prev = x0, y0  errors = []  for \_ in range(max\_iter):  x\_new = g1(x\_prev, y\_prev)  y\_new = g2(x\_prev, y\_prev)  iterations += 1  dx = abs(x\_new - x\_prev)  dy = abs(y\_new - y\_prev)  errors.append((dx, dy))  if dx < eps and dy < eps:  return (x\_new, y\_new), iterations, errors  x\_prev, y\_prev = x\_new, y\_new  return (x\_prev, y\_prev), iterations, errors |

# **Выводы**:

В результате выполнения данной лабораторной работой изучил численные методы решения нелинейных уравнений и систем нелинейных уравнений с использованием языка программирования Python. В рамках работы были вычислены корни заданных уравнений и систем с помощью различных численных методов, а также созданы графики функций и блок-схемы. Было разработано веб-приложение с графическим интерфейсом.