Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 «Программная инженерия» –

Системное и прикладное программное обеспечение

**Отчёт**

**По лабораторной работе №2**

**По вычислительной математике**

**Вариант: 3**

Выполнил:

студент 2 курса

Батманов Даниил Евгеньевич

Группа: Р3207

НЕ Принял:

Рыбаков Степан Дмитриевич

Отчёт принят «\_\_»\_\_\_\_\_2024 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург, 2024

**Оглавление**

[Задание 3](#_Toc162833322)

[Рабочие формулы 3](#_Toc162833323)

[Часть I: вычислительная 3](#_Toc162833324)

[Часть II: программная 7](#_Toc162833325)

[Вывод 19](#_Toc162833326)

# 

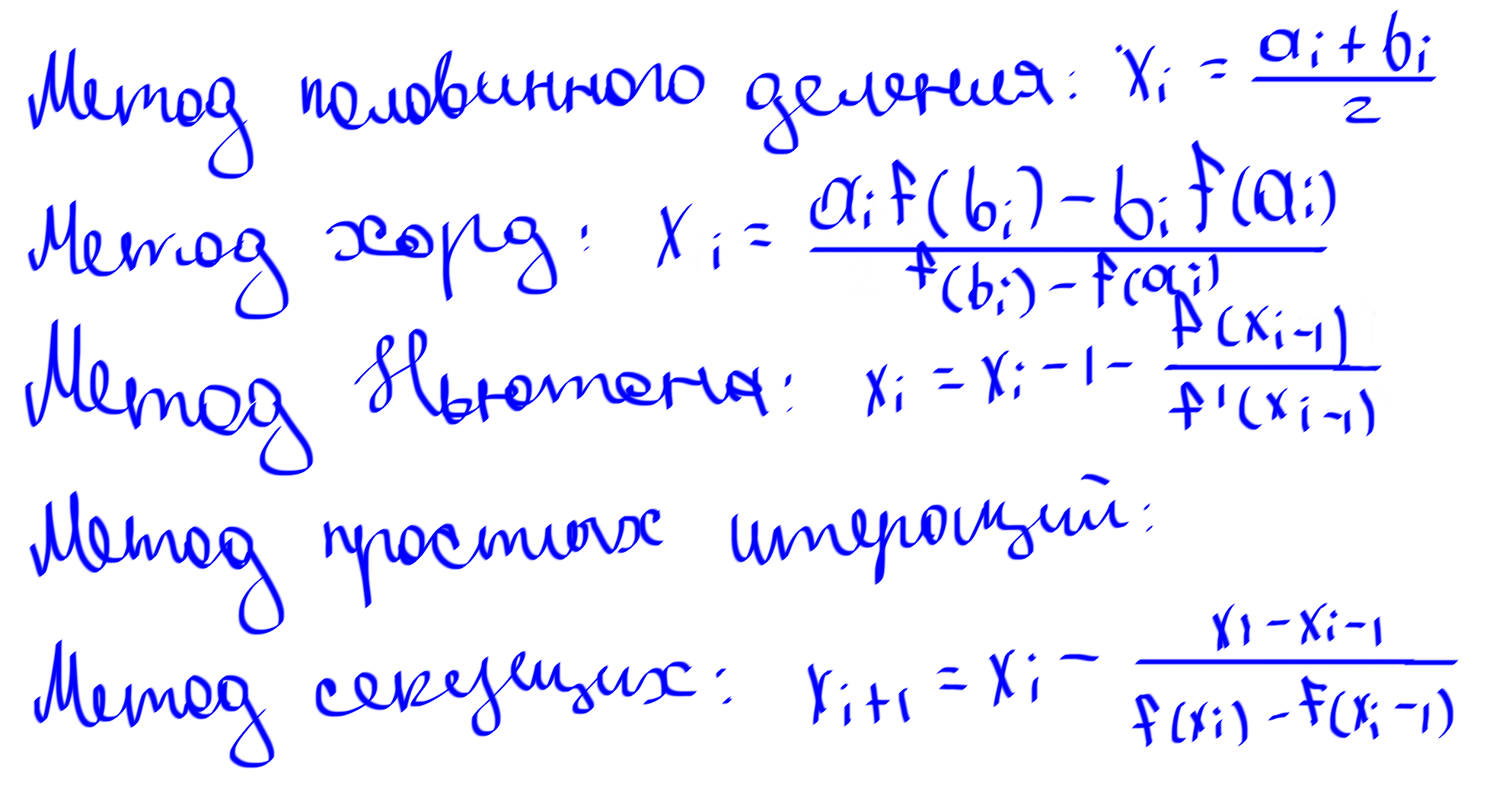
# 

# 

# Задание

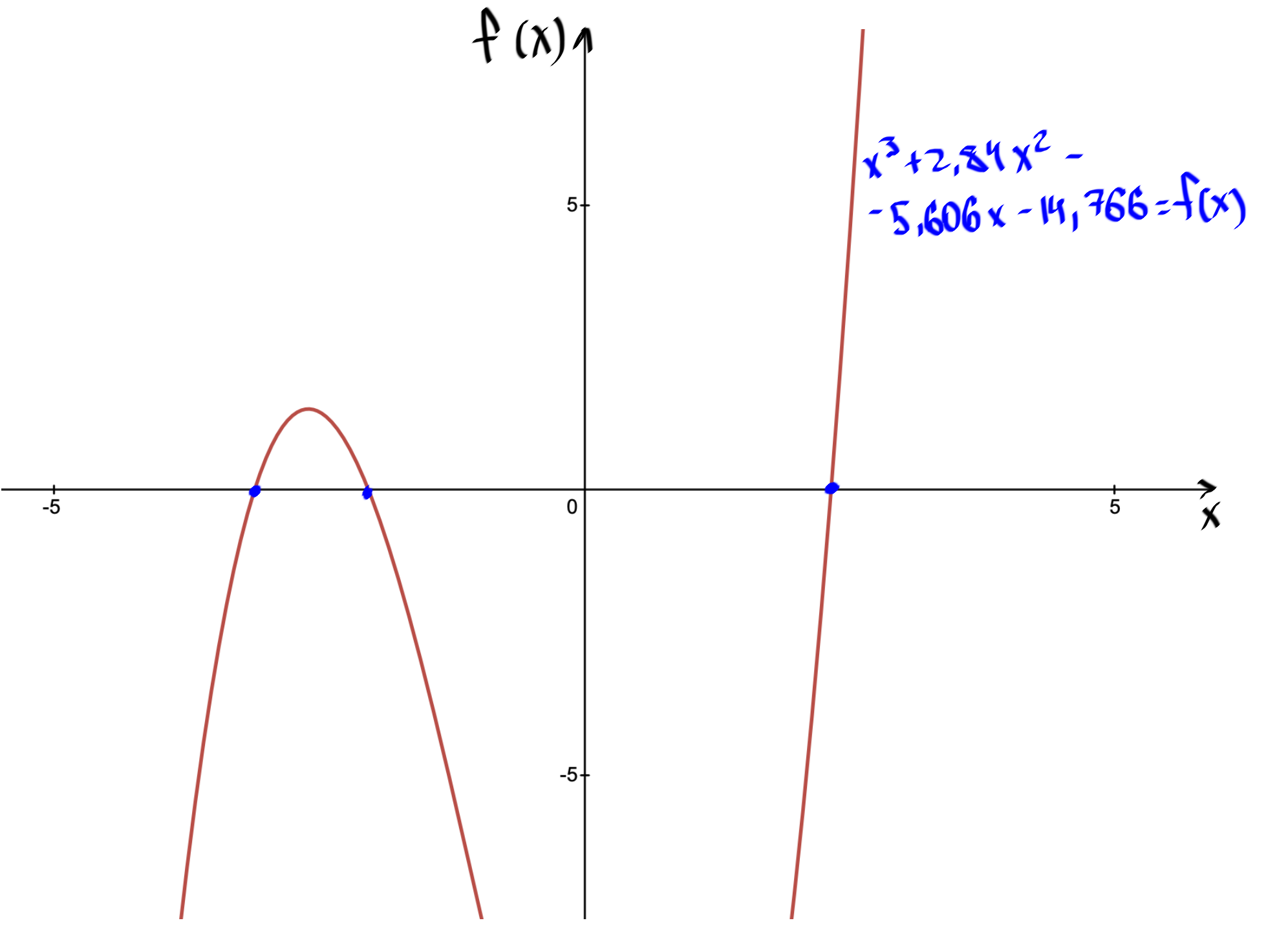
Цель работы: изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

# Рабочие формулы



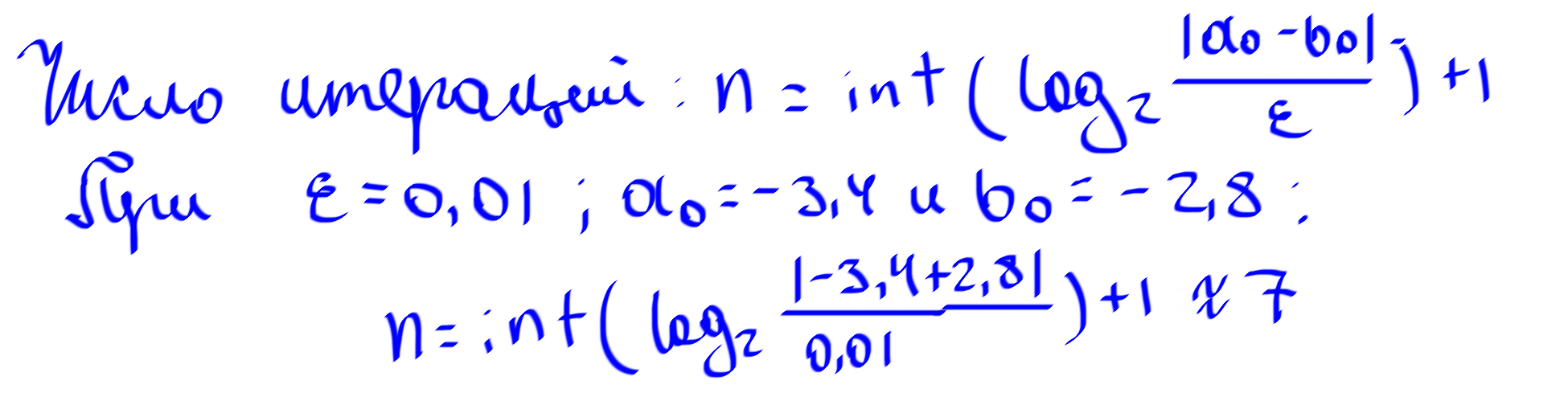
# Часть I: вычислительная

***Исходная функция:***



* Крайний левый корень – **метод** **половинного** **деления**
* Крайний правый корень – **метод простой итерации**
* Центральный корень – **метод Ньютона**

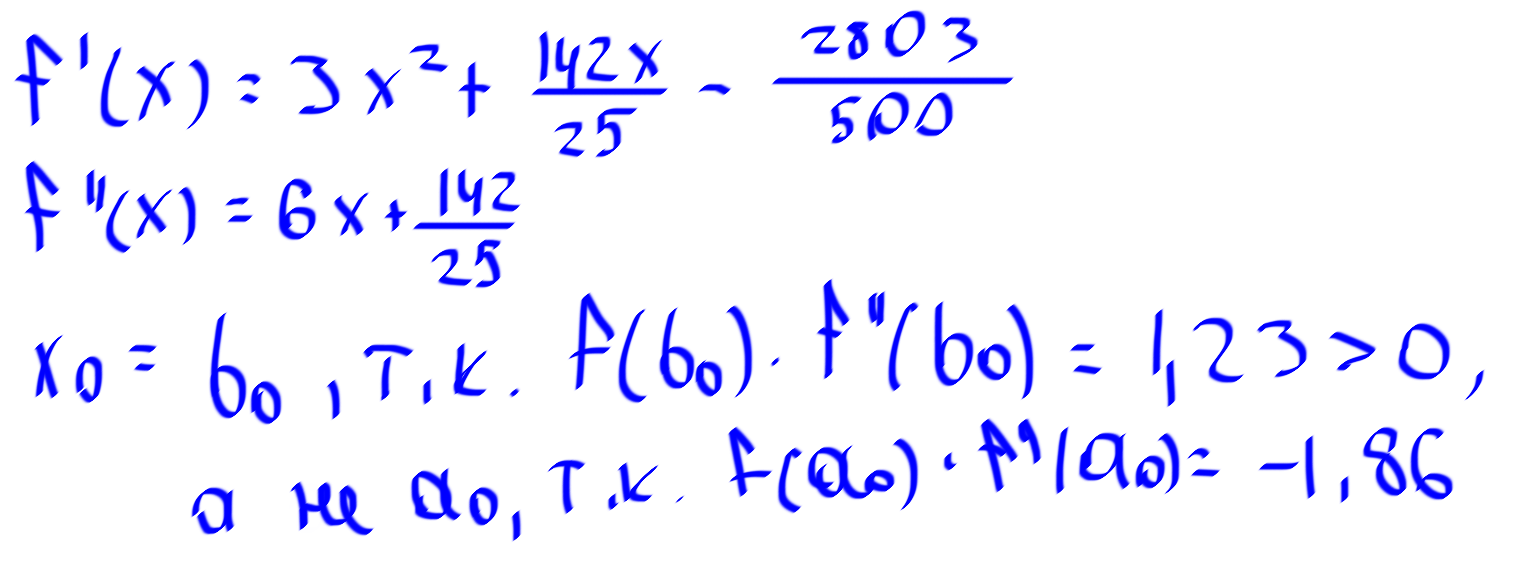
**Крайний левый корень:**



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№ шага** | **a** | **b** | **x** | **f(a)** | **f(b)** | **f(x)** | **|a - b|** |
| **1** | -3,4 | -2,8 | -3,1 | -2,1792 | 1,2444 | 0,114 | 0,6 |
| **2** | -3,4 | -3,1 | -3,25 | -2,1792 | 0,114 | -0,877125 | 0,3 |
| **3** | -3,4 | -3,25 | -3,325 | -2,1792 | -0,877125 | -1,4880281 | 0,15 |
| **4** | -3,4 | -3,325 | -3,3625 | -2,1792 | -1,4880281 | -1,8234223 | 0,075 |
| **5** | -3,4 | -3,3625 | -3,38125 | -2,1792 | -1,8234223 | -1,9987434 | 0,0375 |
| **6** | -3,4 | -3,38125 | -3,390625 | -2,1792 | -1,9987434 | -2,0883273 | 0,01875 |
| **7** | -3,4 | -3,390625 | -3,3953125 | -2,1792 | -2,0883273 | -2,1336022 | 0,009375 |



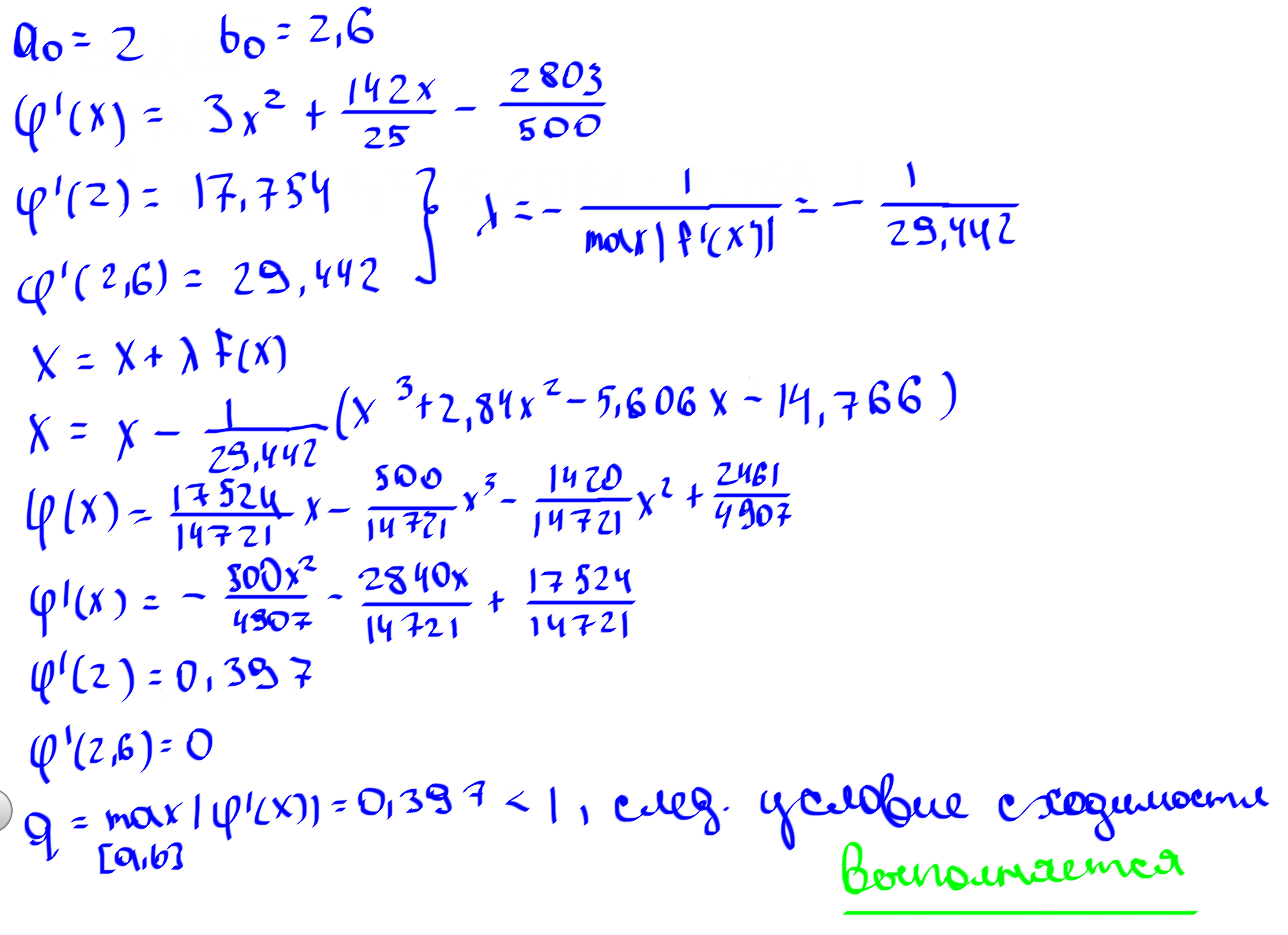
**Центральный корень:**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ шага** | **x\_i** | **f(x\_i)** | **f'(x\_i)** | **x\_(i + 1)** | **|x\_(i + 1) – x\_i|** |
| **0** | -2 | -0,194 | -40,966 | -2,0047356 | 0,00473563 |

0,00473563 < 0.01

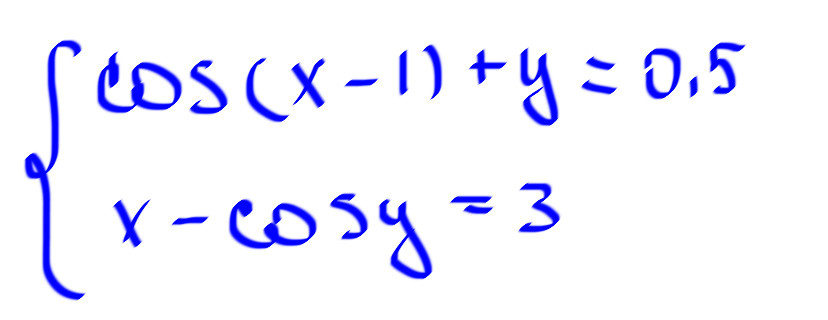
**Крайний правый корень:**

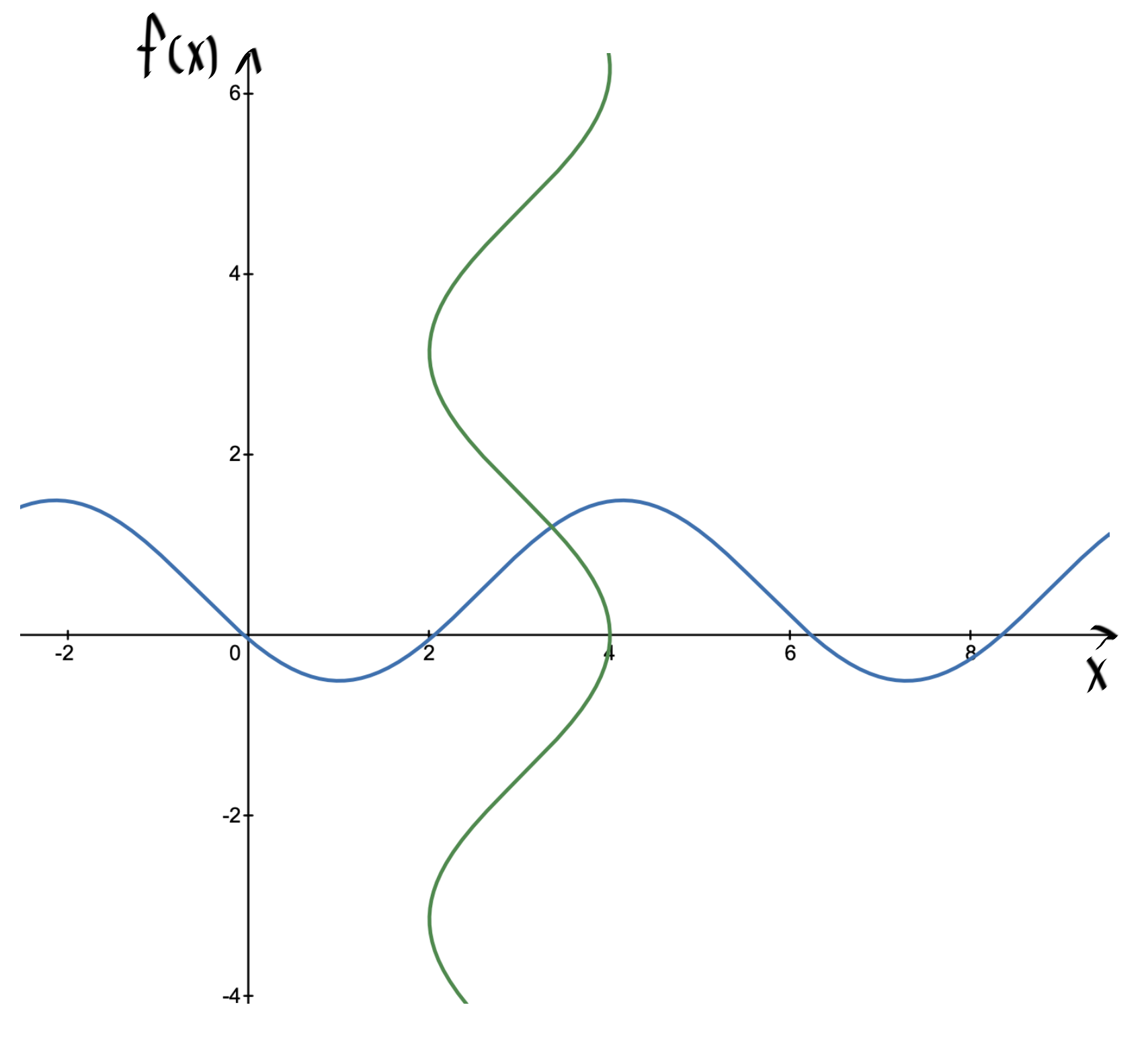


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ шага** | **x\_k** | **x\_(k + 1)** | **fi(x\_(k+1))** | **f(x\_(k + 1))** | **|x\_(k + 1) – x\_k|** |
| **0** | 2 | 2,224 | 2,29771706 | -2,1863087 | 0,224 |
| **1** | 2,224 | 2,29771706 | 2,31500355 | -0,5224057 | 0,07371706 |
| **2** | 2,29771706 | 2,31500355 | 2,31854051 | -0,1170012 | 0,01728649 |
| **3** | 2,31500355 | **2,31854051** | 2,31923977 | -0,0333326 | 0,00353695 |

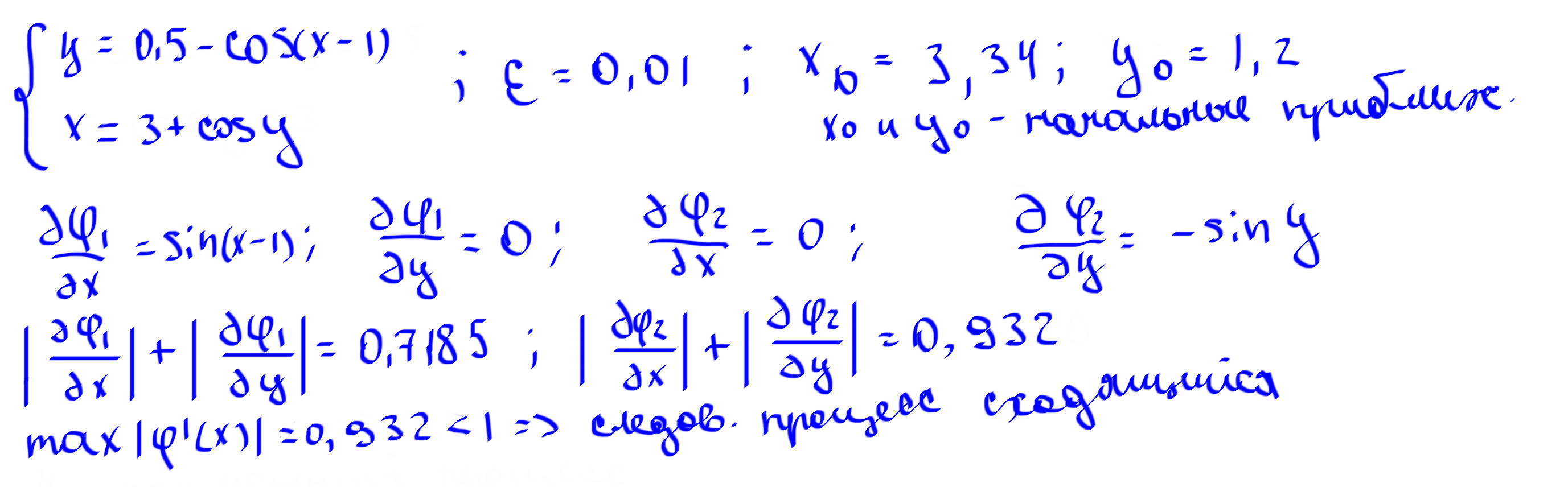
|x\_(k + 1) – x\_k| < 0.01

***Исходная система:***





**Метод простой итерации**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ шага** | **x** | **y** | **|y\_i – y\_(i - 1)|** | **|x\_i – x\_(i - 1)|** |
| **0** | 3,34 | 1,2 | 1,2 | 3,34 |
| **1** | 3,36235775 | 1,19556333 | 0,004436674 | 0,022357754 |
| **2** | 3,36648933 | 1,21145141 | 0,015888083 | 0,004131573 |
| **3** | 3,35166107 | 1,21434873 | 0,002897322 | 0,01482826 |
| **4** | 3,34894733 | 1,20389391 | 0,010454819 | 0,002713735 |
| **5** | **3,35872574** | **1,20196374** | 0,001930169 | 0,009778405 |

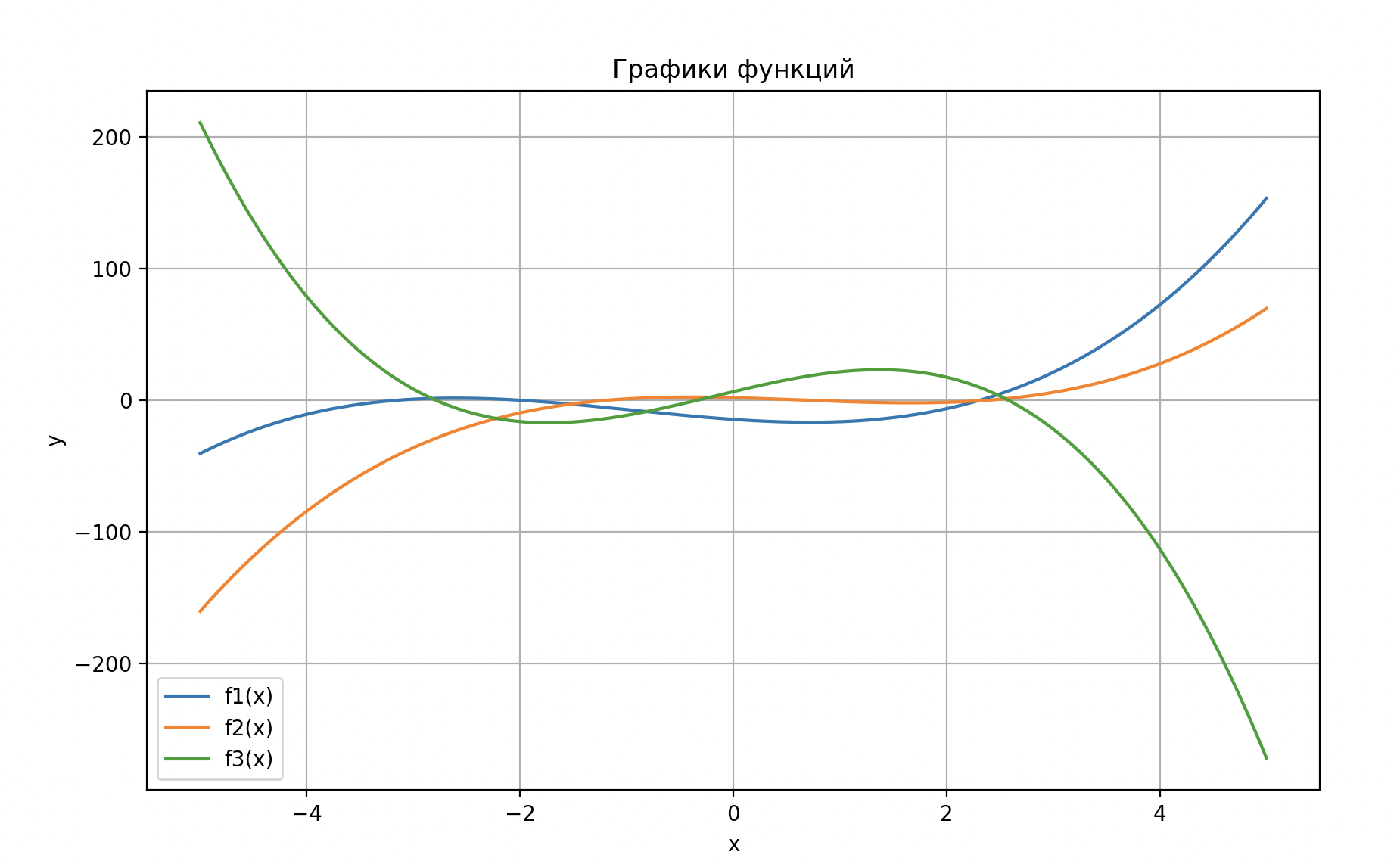
0,009778405 < 0.01 и 0,001930169 < 0.01

# Часть II: программная

Исходный код для уравнений:

import numpy as np  
import matplotlib.pyplot as plt  
  
  
def separate():  
 print("--------------------------")  
  
  
def get\_data\_from\_user():  
 data = []  
 separate()  
 print("Невероятный решатор нелинейных уравнений")  
 separate()  
 draw\_func()  
 data.append(int(input("Выберите функцию из списка:\n"  
 "1) f(x) = x^3 + 2.84x^2 - 5.606x - 14.766\n"  
 "2) f(x) = x^3 - 1.89x^2 - 2x + 1.76\n"  
 "3) f(x) = -2.7x^3 - 1.48x^2 + 19.23x + 6.35\n"  
 "Номер выбранной функции: ")))  
 separate()  
 data.append(float(input("Выберите левую границу интервала/начальное приближение к корню слева: ")))  
 separate()  
 data.append(float(input("Выберите правую границу интервала/начальное приближение к корню справа: ")))  
 separate()  
 data.append(float(input("Задайте погрешность вычисления: ")))  
 separate()  
 return data  
  
  
def get\_data\_from\_file(file\_path):  
 data = []  
 try:  
 with open(file\_path, 'r') as file:  
 data.append(int(file.readline()))  
 data.append(float(file.readline()))  
 data.append(float(file.readline()))  
 data.append(float(file.readline()))  
 print("Спасибо, данные считанны корректно")  
 except FileNotFoundError:  
 print("Файл не найден")  
 except ValueError:  
 print("Неверный формат данных в файле")  
 return data  
  
  
def f1(x):  
 return x \*\* 3 + 2.84 \* x \* x - 5.606 \* x - 14.766  
  
  
def f2(x):  
 return x \*\* 3 - 1.89 \* x \* x - 2 \* x + 1.76  
  
  
def f3(x):  
 return -2.7 \* x \*\* 3 - 1.48 \* x \* x + 19.23 \* x + 6.35  
  
  
def draw\_func():  
 x\_values = np.linspace(-5, 5, 400)  
 y1\_values = f1(x\_values)  
 y2\_values = f2(x\_values)  
 y3\_values = f3(x\_values)  
  
 # Построение графиков  
 plt.figure(figsize=(10, 6))  
  
 plt.plot(x\_values, y1\_values, label='f1(x)')  
 plt.plot(x\_values, y2\_values, label='f2(x)')  
 plt.plot(x\_values, y3\_values, label='f3(x)')  
  
 plt.title('Графики функций')  
 plt.xlabel('x')  
 plt.ylabel('y')  
 plt.legend()  
  
 plt.grid(True)  
 plt.show()  
  
  
def check\_x(tttype, a\_1m, b\_1m, epsilon\_1m):  
 check = 0  
 if ttype == 1 and not((a\_1m <= -3.12 <= b\_1m) or (a\_1m <= -2.04 <= b\_1m) or (a\_1m <= 2.32 <= b\_1m) and a\_1m <= b\_1m):  
 check = 1  
 elif ttype == 2 and not((a\_1m <= -1.156 <= b\_1m) or (a\_1m <= 0.63 <= b\_1m) or (a\_1m <= 2.416 <= b\_1m) and a\_1m <= b\_1m):  
 check = 1  
 elif ttype == 3 and not((a\_1m <= -2.795 <= b\_1m) or (a\_1m <= -0.327 <= b\_1m) or (a\_1m <= 2.574 <= b\_1m) and a\_1m <= b\_1m):  
 check = 1  
 return check  
  
  
def halving(tttype, a\_1m, b\_1m, epsilon\_1m):  
 if check\_x(tttype, a\_1m, b\_1m, epsilon\_1m):  
 return "На данном промежутке нет корней, выберите другой промежуток"  
 a\_copy = a\_1m  
 b\_copy = b\_1m  
 counter = 0  
 while True:  
 counter += 1  
 x = (a\_copy + b\_copy) / 2  
 if tttype == 1:  
 y\_a = f1(a\_copy)  
 y\_b = f1(b\_copy)  
 y\_x = f1(x)  
 if abs(a\_copy - b\_copy) < epsilon\_1m:  
 return x, y\_x, counter  
 else:  
 if y\_a < y\_b:  
 b\_copy = x  
 else:  
 a\_copy = x  
 if tttype == 2:  
 y\_a = f2(a\_copy)  
 y\_b = f2(b\_copy)  
 y\_x = f2(x)  
 if abs(a\_copy - b\_copy) < epsilon\_1m:  
 return x, y\_x, counter  
 else:  
 if y\_a < y\_b:  
 b\_copy = x  
 else:  
 a\_copy = x  
 if tttype == 3:  
 y\_a = f3(a\_copy)  
 y\_b = f3(b\_copy)  
 y\_x = f3(x)  
 if abs(a\_copy - b\_copy) < epsilon\_1m:  
 return x, y\_x, counter  
 else:  
 if y\_a < y\_b:  
 b\_copy = x  
 else:  
 a\_copy = x  
  
  
def secuch(tttype, a\_1m, b\_1m, epsilon\_1m):  
 if check\_x(tttype, a\_1m, b\_1m, epsilon\_1m):  
 return "На данном промежутке нет корней, выберите другой промежуток"  
 x\_0 = a\_1m  
 x\_1 = x\_0 + 0.3  
 counter = 0  
 while True:  
 counter += 1  
 if tttype == 1:  
 x\_i1 = x\_1 - (x\_1 - x\_0) / (f1(x\_1) - f1(x\_0)) \* f1(x\_1)  
 y\_i1 = f1(x\_i1)  
 if abs(x\_i1 - x\_1) < epsilon\_1m:  
 return x\_i1, y\_i1, counter  
 else:  
 x\_0 = x\_1  
 x\_1 = x\_i1  
 if tttype == 2:  
 x\_i1 = x\_1 - (x\_1 - x\_0) / (f2(x\_1) - f2(x\_0)) \* f2(x\_1)  
 y\_i1 = f2(x\_i1)  
 if abs(x\_i1 - x\_1) < epsilon\_1m:  
 return x\_i1, y\_i1, counter  
 else:  
 x\_0 = x\_1  
 x\_1 = x\_i1  
 if tttype == 3:  
 x\_i1 = x\_1 - (x\_1 - x\_0) / (f3(x\_1) - f3(x\_0)) \* f3(x\_1)  
 y\_i1 = f3(x\_i1)  
 if abs(x\_i1 - x\_1) < epsilon\_1m:  
 return x\_i1, y\_i1, counter  
 else:  
 x\_0 = x\_1  
 x\_1 = x\_i1  
  
  
  
def dx1(x):  
 return 3 \* x \* x + 422 \* x / 25 - 2803 / 500  
  
  
def dx2(x):  
 return 3 \* x \* x - 189 \* x / 50 - 2  
  
  
def dx3(x):  
 return -81 \* x \* x / 10 - 74 \* x / 25 + 1923 / 100  
  
  
def simple\_it(tttype, a\_1m, b\_1m, epsilon\_1m):  
 counter = 0  
 if tttype == 1:  
 lam = 1 / max(dx1(a\_1m), dx1(b\_1m))  
 if f1((a\_1m + b\_1m) / 2) > 0:  
 lam = -1 \* lam  
 if not(1 + lam \* dx1(a\_1m) < 1 and 1 + lam \* dx1(b\_1m) < 1):  
 return "На данном промежутке условие сходимости не выполняется, выберите другой промежуток"  
 else:  
 x\_0 = a\_1m  
 while True:  
 counter += 1  
 x\_1 = x\_0 + lam \* f1(x\_0)  
 y\_1 = f1(x\_1)  
 if abs(x\_1 - x\_0) < epsilon\_1m:  
 return x\_1, y\_1, counter  
 else:  
 x\_0 = x\_1  
 if tttype == 2:  
 lam = 1 / max(dx2(a\_1m), dx2(b\_1m))  
 if f2((a\_1m + b\_1m) / 2) > 0:  
 lam = -1 \* lam  
 if not(1 + lam \* dx2(a\_1m) < 1 and 1 + lam \* dx2(b\_1m) < 1):  
 return "На данном промежутке условие сходимости не выполняется, выберите другой промежуток"  
 else:  
 x\_0 = a\_1m  
 while True:  
 counter += 1  
 x\_1 = x\_0 + lam \* f2(x\_0)  
 y\_1 = f2(x\_1)  
 if abs(x\_1 - x\_0) < epsilon\_1m:  
 return x\_1, y\_1, counter  
 else:  
 x\_0 = x\_1  
 if tttype == 3:  
 lam = 1 / max(dx3(a\_1m), dx3(b\_1m))  
 if f3((a\_1m + b\_1m) / 2) > 0:  
 lam = -1 \* lam  
 if not(1 + lam \* dx3(a\_1m) < 1 and 1 + lam \* dx3(b\_1m) < 1):  
 return "На данном промежутке условие сходимости не выполняется, выберите другой промежуток"  
 else:  
 x\_0 = a\_1m  
 while True:  
 counter += 1  
 x\_1 = x\_0 + lam \* f3(x\_0)  
 y\_1 = f3(x\_1)  
 if abs(x\_1 - x\_0) < epsilon\_1m:  
 return x\_1, y\_1, counter  
 else:  
 x\_0 = x\_1  
  
  
separate()  
way\_of\_input = input("Выберите способ вывода значений:\n"  
 "0) С клавиатуры\n"  
 "1) Из файла\n"  
 "Введите номер выбранного варианта: ")  
separate()  
if way\_of\_input:  
 mas = get\_data\_from\_user()  
 ttype = mas[0]  
 a = mas[1]  
 b = mas[2]  
 epsilon = mas[3]  
else:  
 path = input("Укажите путь до файла: ")  
 separate()  
 mas = get\_data\_from\_file(path)  
 ttype = mas[0]  
 a = mas[1]  
 b = mas[2]  
 epsilon = mas[3]  
  
print("Метод половинного деления: " + str(halving(ttype, a, b, epsilon)) + " (x, y, число итераций)")  
print("Метод секущих: " + str(secuch(ttype, a, b, epsilon)) + " (x, y, число итераций)")  
print("Метод простых итераций: " + str(simple\_it(ttype, a, b, epsilon)) + " (x, y, число итераций)")  
separate()  
troll = input("Записать данные файл?\n"  
 "0 – нет\n"  
 "1 – да\n"  
 "Ответ: ")  
separate()  
if troll:  
 fff = input("Укажите путь до файла: ")  
 with open(fff, 'w') as file:  
 file.write(str(halving(ttype, a, b, epsilon)))  
 file.write(str(secuch(ttype, a, b, epsilon)))  
 file.write(str(simple\_it(ttype, a, b, epsilon)))  
 print("Результаты успешно записаны в файл")

Вывод программы для уравнений:

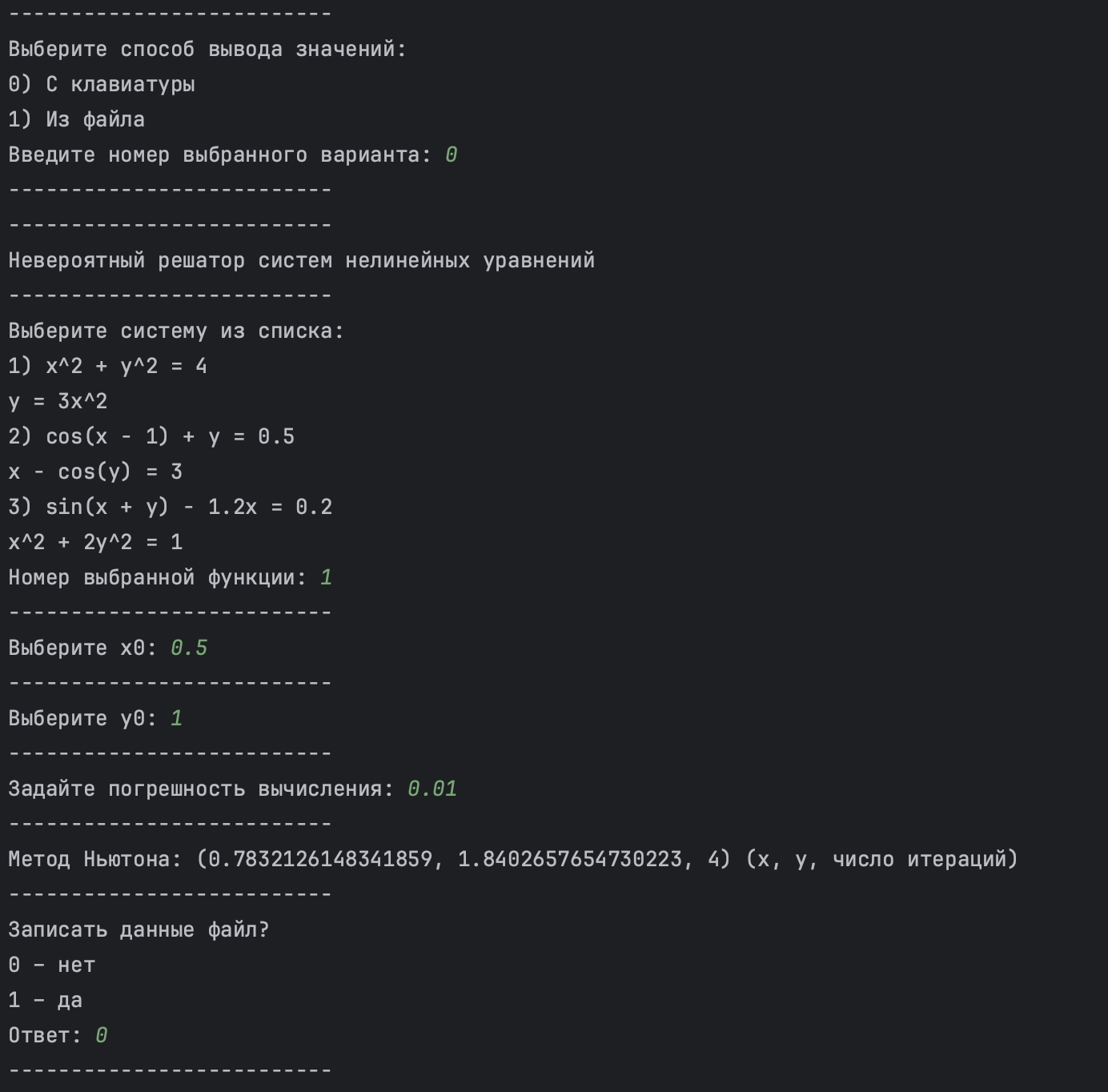


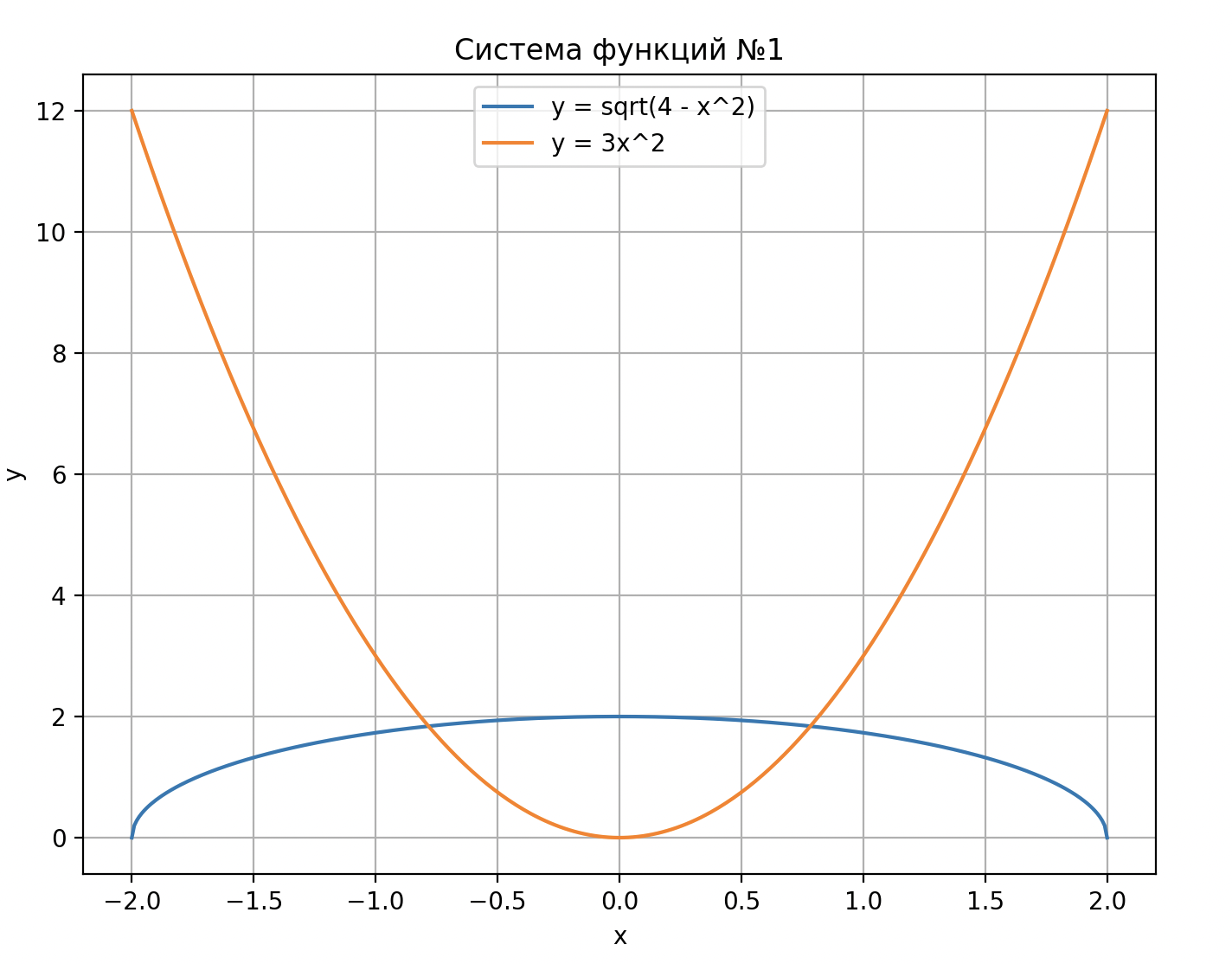


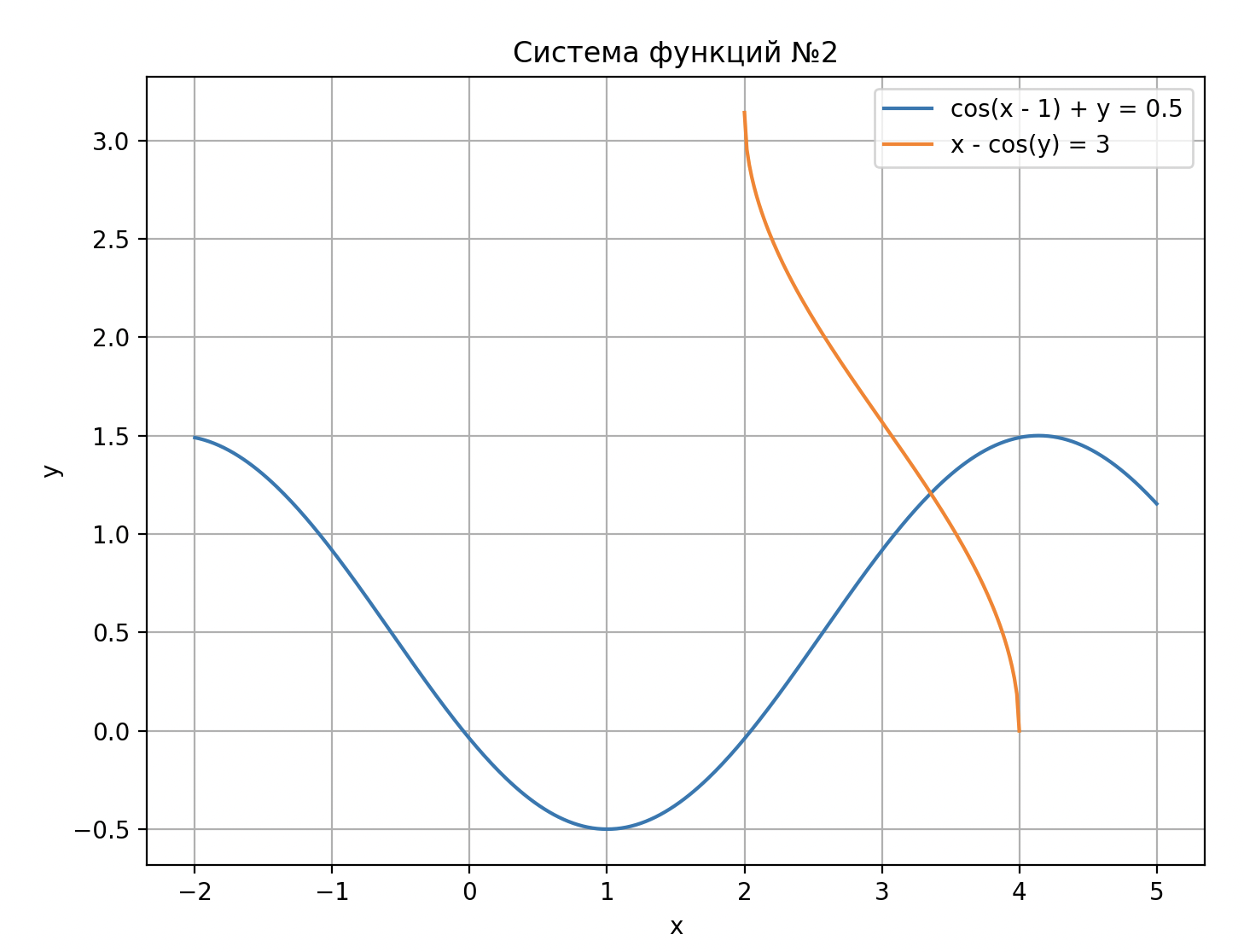
Исходный код для систем:

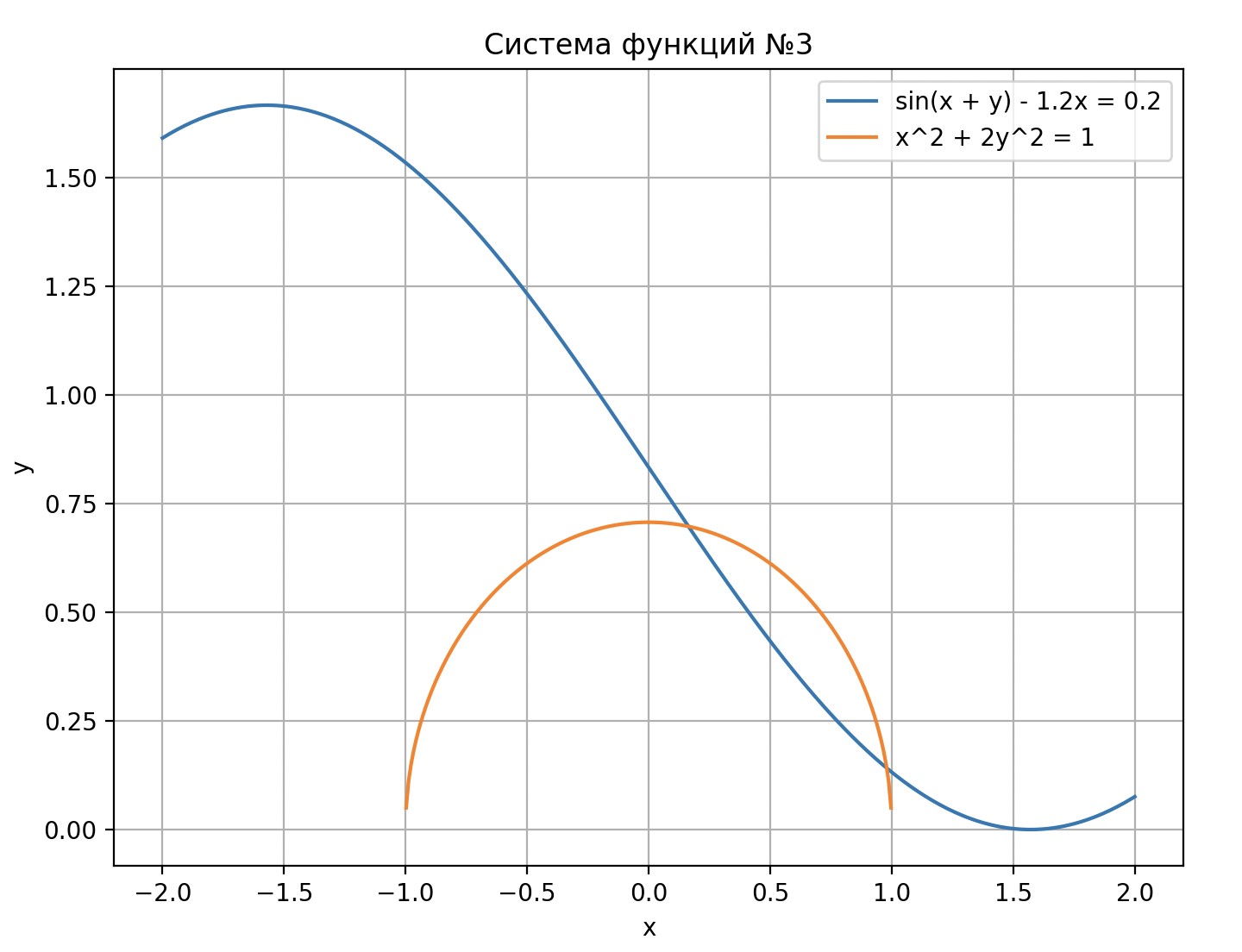
import math  
  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
def separate():  
 print("--------------------------")  
  
  
def system1(x):  
 y1 = np.sqrt(4 - x \*\* 2)  
 y2 = 3 \* x \*\* 2  
 return y1, y2  
  
  
def system2(x):  
 y1 = 0.5 - np.cos(x - 1)  
 y2 = np.arccos(x - 3)  
 return y1, y2  
  
  
def system3(x):  
 y1 = (1 - np.sin(x)) / 1.2  
 y2 = np.sqrt((1 - x \*\* 2) / 2)  
 return y1, y2  
  
  
def plot\_system(system\_number):  
 plt.figure(figsize=(8, 6))  
 plt.grid(True)  
  
 if system\_number == 1:  
 x\_values = np.linspace(-2, 2, 400)  
 y1, y2 = system1(x\_values)  
 plt.plot(x\_values, y1, label='y = sqrt(4 - x^2)')  
 plt.plot(x\_values, y2, label='y = 3x^2')  
 plt.xlabel('x')  
 plt.ylabel('y')  
 plt.title('Система функций №1')  
 plt.legend()  
  
 elif system\_number == 2:  
 x\_values = np.linspace(-2, 5, 400)  
 y1, y2 = system2(x\_values)  
 plt.plot(x\_values, y1, label='cos(x - 1) + y = 0.5')  
 plt.plot(x\_values, y2, label='x - cos(y) = 3')  
 plt.xlabel('x')  
 plt.ylabel('y')  
 plt.title('Система функций №2')  
 plt.legend()  
  
 elif system\_number == 3:  
 x\_values = np.linspace(-2, 2, 400)  
 y1, y2 = system3(x\_values)  
 plt.plot(x\_values, y1, label='sin(x + y) - 1.2x = 0.2')  
 plt.plot(x\_values, y2, label='x^2 + 2y^2 = 1')  
 plt.xlabel('x')  
 plt.ylabel('y')  
 plt.title('Система функций №3')  
 plt.legend()  
  
 else:  
 print("Неправильный номер системы. Введите число от 1 до 3.")  
 return  
  
 plt.show()  
  
  
def get\_data\_from\_user():  
 data = []  
 separate()  
 print("Невероятный решатор систем нелинейных уравнений")  
 separate()  
 data.append(int(input("Выберите систему из списка:\n"  
 "1) x^2 + y^2 = 4\n"  
 "y = 3x^2\n"  
 "2) cos(x - 1) + y = 0.5\n"  
 "x - cos(y) = 3\n"  
 "3) sin(x + y) - 1.2x = 0.2\n"  
 "x^2 + 2y^2 = 1\n"  
 "Номер выбранной функции: ")))  
 separate()  
 plot\_system(data[0])  
 data.append(float(input("Выберите x0: ")))  
 separate()  
 data.append(float(input("Выберите y0: ")))  
 separate()  
 data.append(float(input("Задайте погрешность вычисления: ")))  
 separate()  
 return data  
  
  
def get\_data\_from\_file(file\_path):  
 data = []  
 try:  
 with open(file\_path, 'r') as file:  
 data.append(int(file.readline()))  
 data.append(float(file.readline()))  
 data.append(float(file.readline()))  
 data.append(float(file.readline()))  
 print("Спасибо, данные считанны корректно")  
 except FileNotFoundError:  
 print("Файл не найден")  
 except ValueError:  
 print("Неверный формат данных в файле")  
 return data  
  
  
def newton(tttype, x0, y0, eps):  
 count = 0  
 while True:  
 count += 1  
 if tttype == 1:  
 dx = (4 - x0 \* x0 - 6 \* x0 \* x0 \* y0 + y0 \* y0) / (2 \* x0 + 12 \* x0 \* y0)  
 dy = 3 \* x0 \* x0 - y0 + 6 \* x0 \* dx  
 if tttype == 2:  
 dx = (0.5 - y0 - math.cos(x0 - 1)) / (-math.sin(x0 - 1))  
 dy = (3 + math.cos(y0) - x0 - dx) / (3 \* math.sin(y0))  
 if tttype == 3:  
 dx = (1 - 2 \* y0 \* y0 - x0 \* x0 - 4 \* y0 \* ((0.2 + 1.2 \* x0 - math.sin(x0 + y0)) / math.cos(x0 + y0))) / (2 \* x0 - 4 \* y0)  
 dy = (0.2 + 1.2 \* x0 - math.sin(x0 + y0) - dx \* math.cos(x0 + y0)) / math.cos(x0 + y0)  
 x1 = dx + x0  
 y1 = dy + y0  
 if abs(x1 - x0) <= eps and abs(y1 - y0) <= eps:  
 return x1, y1, count  
 else:  
 x0 = x1  
 y0 = y1  
  
  
separate()  
way\_of\_input = input("Выберите способ вывода значений:\n"  
 "0) С клавиатуры\n"  
 "1) Из файла\n"  
 "Введите номер выбранного варианта: ")  
separate()  
if way\_of\_input:  
 mas = get\_data\_from\_user()  
 ttype = mas[0]  
 a = mas[1]  
 b = mas[2]  
 epsilon = mas[3]  
else:  
 path = input("Укажите путь до файла: ")  
 separate()  
 mas = get\_data\_from\_file(path)  
 ttype = mas[0]  
 a = mas[1]  
 b = mas[2]  
 epsilon = mas[3]  
  
  
print("Метод Ньютона: " + str(newton(ttype, a, b, epsilon)) + " (x, y, число итераций)")  
separate()  
troll = input("Записать данные файл?\n"  
 "0 – нет\n"  
 "1 – да\n"  
 "Ответ: ")  
separate()  
if troll == 1:  
 fff = input("Укажите путь до файла: ")  
 with open(fff, 'w') as file:  
 file.write(str(newton(ttype, a, b, epsilon)))  
 print("Результаты успешно записаны в файл")

Вывод программы для систем:









# Вывод

В ходе выполнения данной лабораторной работы я изучил численные методы решения нелинейных уравнений и их систем, нашёл корни заданного линейного уравнения/системы нелинейных уравнений и реализовал некоторые методы решения на языке Python. Это очень пригодится мне в жизни, но я пока не очень понял где.