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 №4**

**По методам оптимизации**

**Вариант: 4**

Выполнил:

студент 2 курса

Батманов Даниил Евгеньевич

Группа: Р3207

Приняла:

Селина Елена Георгиевна

Отчёт принят «\_\_»\_\_\_\_\_2024 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург, 2024

**Оглавление**

[Задание 3](#_Toc160626351)

[Ручное решение 3](#_Toc160626353)

[Исходный код программы 4](#_Toc160626354)

[Результат работы программы 6](#_Toc160626355)

[Вывод 6](#_Toc160626356)

# 

# 

# 

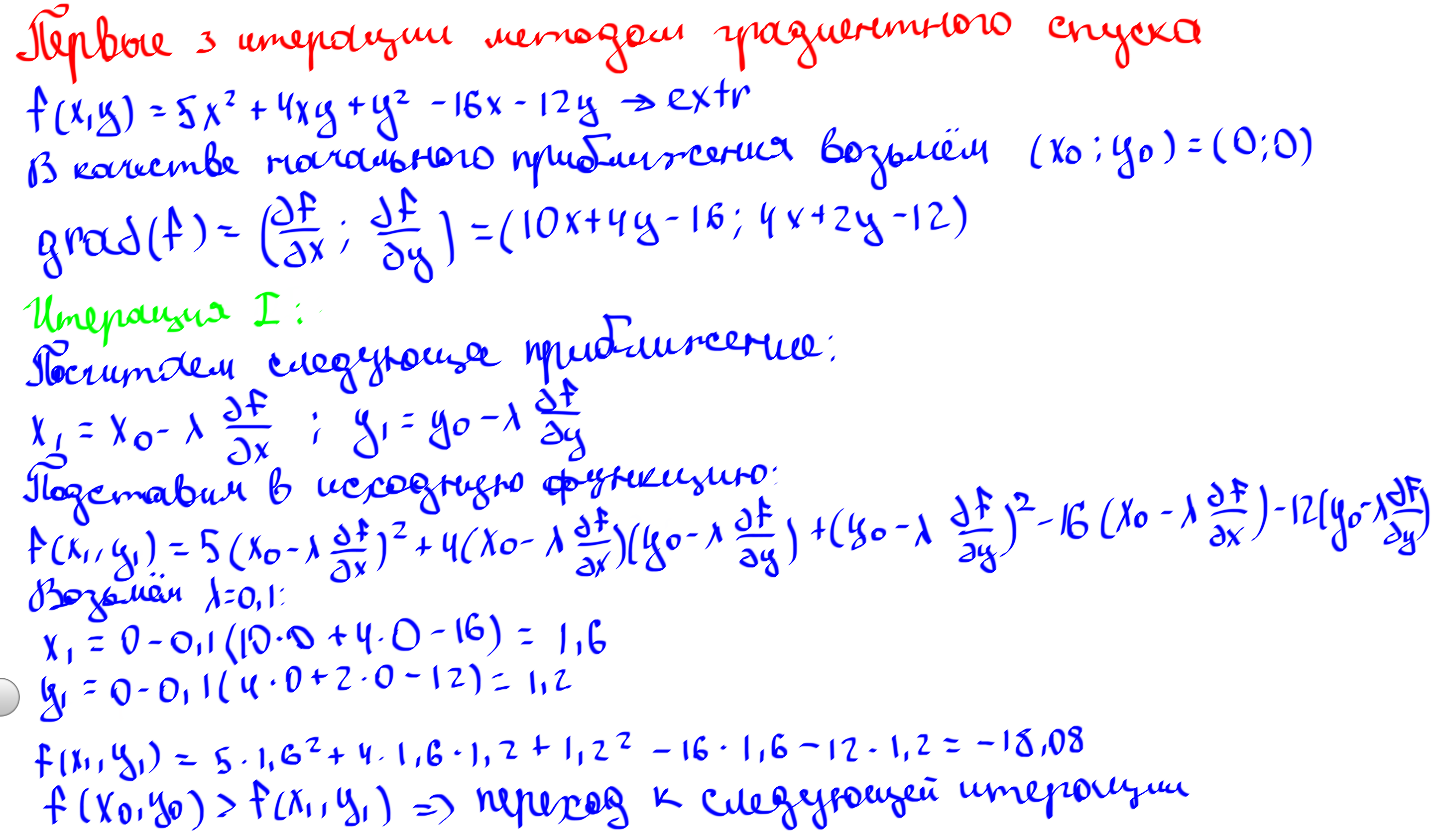
# Задание

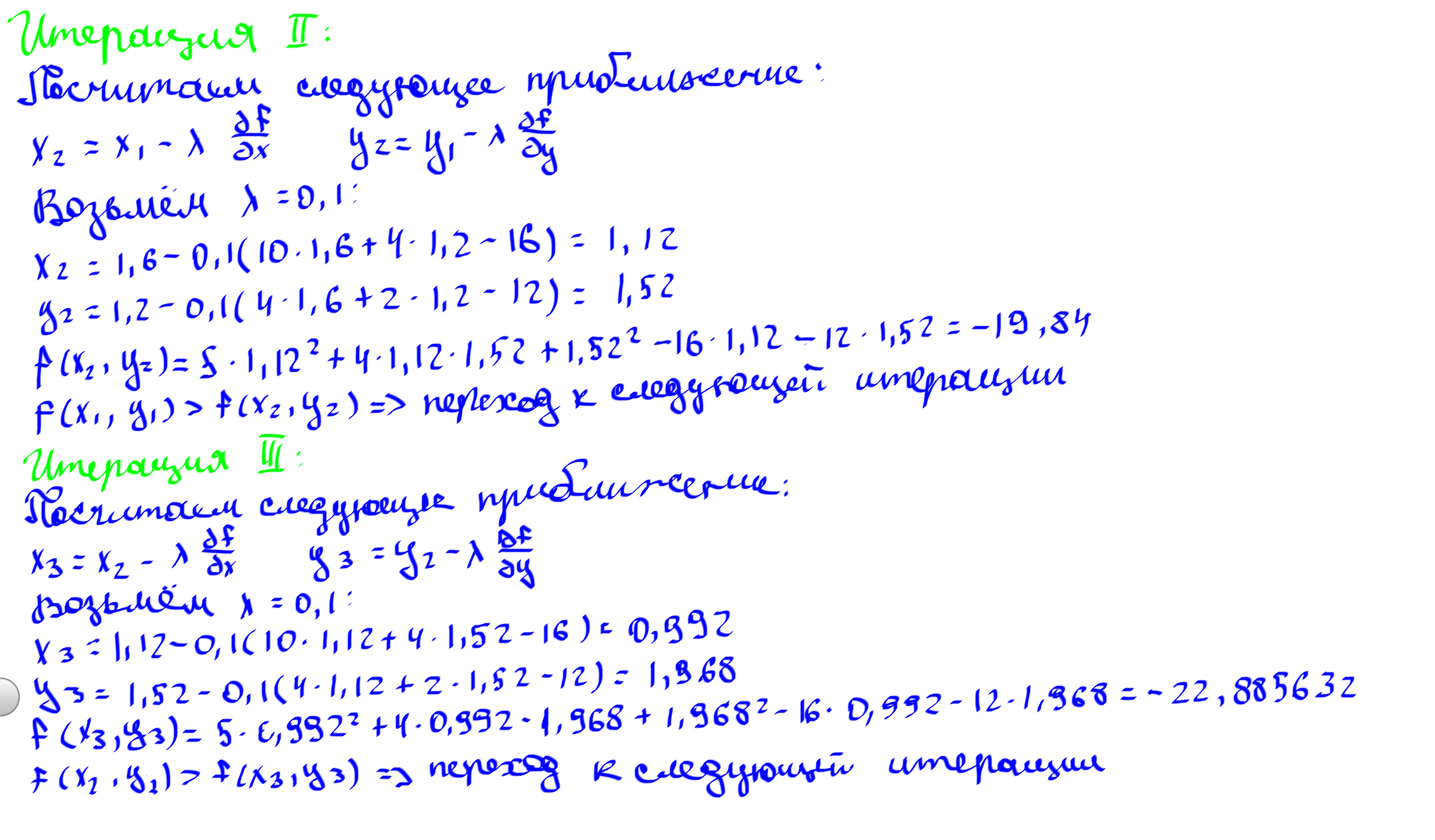
# Задание 1. Решить задачу безусловной минимизации функции двух переменных используя метод градиентного спуска.

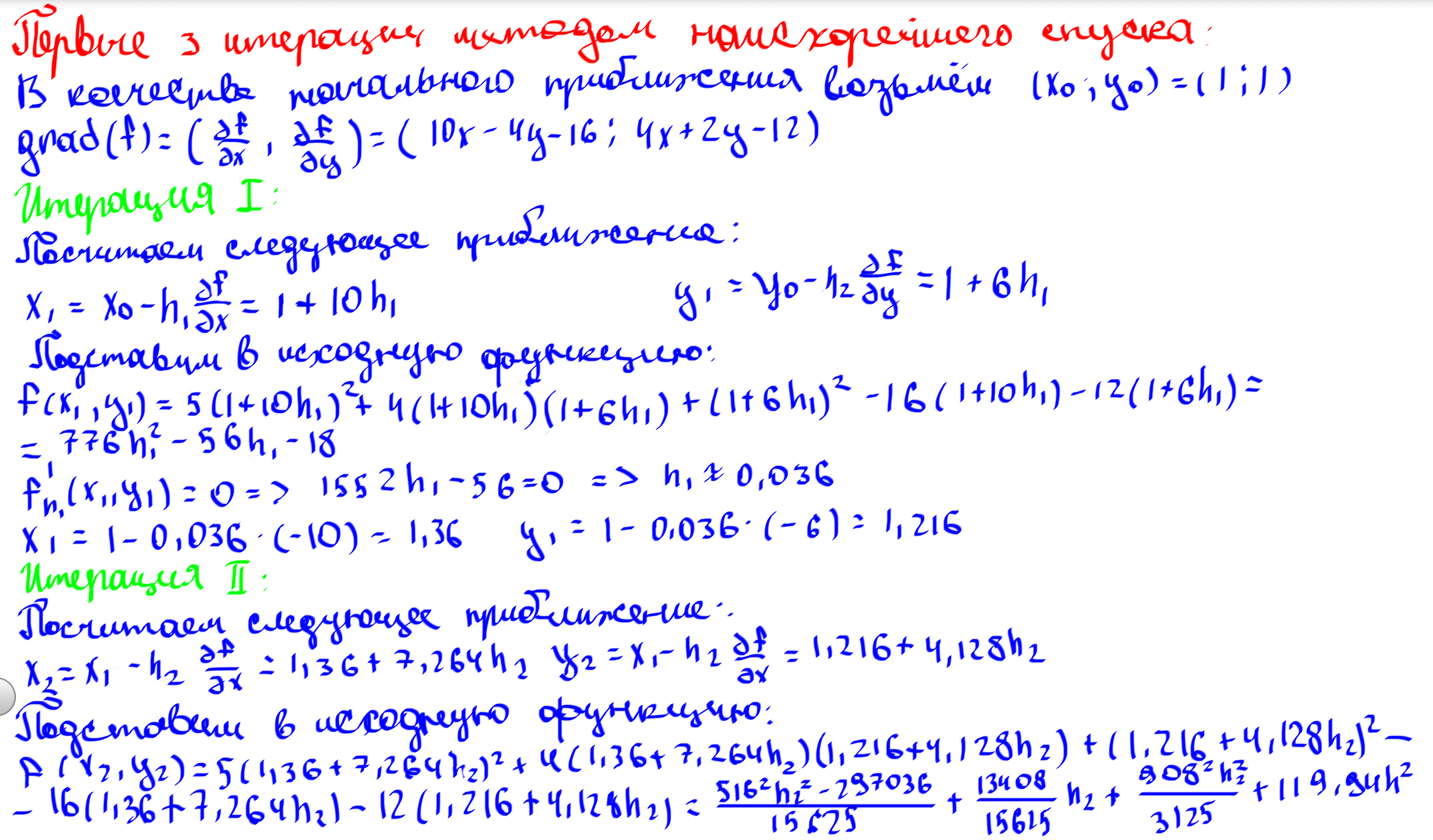
# Задание 2. Решить задачу безусловной минимизации функции двух переменных используя метод наискорейшего спуска.

# Три итерации метода выполнить вручную + написать программу на одном из языков программирования.

# Ручное решение







# Исходный код программы

def f(x, y):  
 return 5 \* x \*\* 2 + 4 \* x \* y + y \*\* 2 - 16 \* x - 12 \* y  
  
  
def gradient(x, y):  
 return [10 \* x + 4 \* y - 16, 4 \* x + 2 \* y - 12]  
  
  
def gradient\_module(x, y):  
 dx, dy = gradient(x, y)  
 return (dx \*\* 2 + dy \*\* 2) \*\* 0.5  
  
  
x = 0  
y = 0  
step\_value = 0.1  
epsilon = 0.05  
  
  
while True:  
 f\_current = f(x, y)  
 if gradient\_module(x, y) <= epsilon:  
 break  
 step = step\_value  
 while True:  
 dx, dy = gradient(x, y)  
 x\_new = x - step \* dx  
 y\_new = y - step \* dy  
 f\_new = f(x\_new, y\_new)  
 if f\_new < f\_current:  
 break  
 step /= 2  
 x = x\_new  
 y = y\_new  
  
print(x, y)

def f(x, y):  
 return 5 \* x \*\* 2 + 4 \* x \* y + y \*\* 2 - 16 \* x - 12 \* y  
  
  
def gradient(x, y):  
 return [10 \* x + 4 \* y - 16, 4 \* x + 2 \* y - 12]  
  
  
def gradient\_module(x, y):  
 dx, dy = gradient(x, y)  
 return (dx \*\* 2 + dy \*\* 2) \*\* 0.5  
  
  
def calculate\_step(x, y):  
 dx, dy = gradient(x, y)  
 return (- 10 \* x - 4 \* x \* dy - 4 \* y \* dx - 2 \* y + 16 \* dx + 12 \* dy) / (- 10 \* dx \* dx - 2 \* dy \* dx - 2 \* dy \* dy)  
  
  
x = 1  
y = 1  
epsilon = 0.05  
  
  
while True:  
 if gradient\_module(x, y) <= epsilon:  
 break  
 step = calculate\_step(x, y)  
  
 dx, dy = gradient(x, y)  
 x = x - step \* dx  
 y = y - step \* dy  
  
  
print(x, y)

# Результат работы программы

*-3.9448597367579183 13.866879628648144*

# Вывод

В ходе выполнения данной лабораторной работы я ознакомился с градиентными методами: градиентного и наискорейшего спуска; реализовал их, как программный продукт.