

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГАОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **технологий** | **Кафедра**  **информационных технологий и вычислительных систем** |

ОТЧЕТ О ВЫПОЛНЕНИИ   
ИНДИВИДУАЛЬНОГО ЗАДАНИЯ ПО ДИСЦИПЛИНЕ   
«Методы оптимизации»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТА | *2* | КУРСА | | *бакалавриата* | ГРУППЫ | *ИДБ-23-17* |
|  | | | *(уровень профессионального образования)* | |  | |

|  |
| --- |
| **Грачев Николай Романовича** |
| *(ФИО)* |

НА ТЕМУ

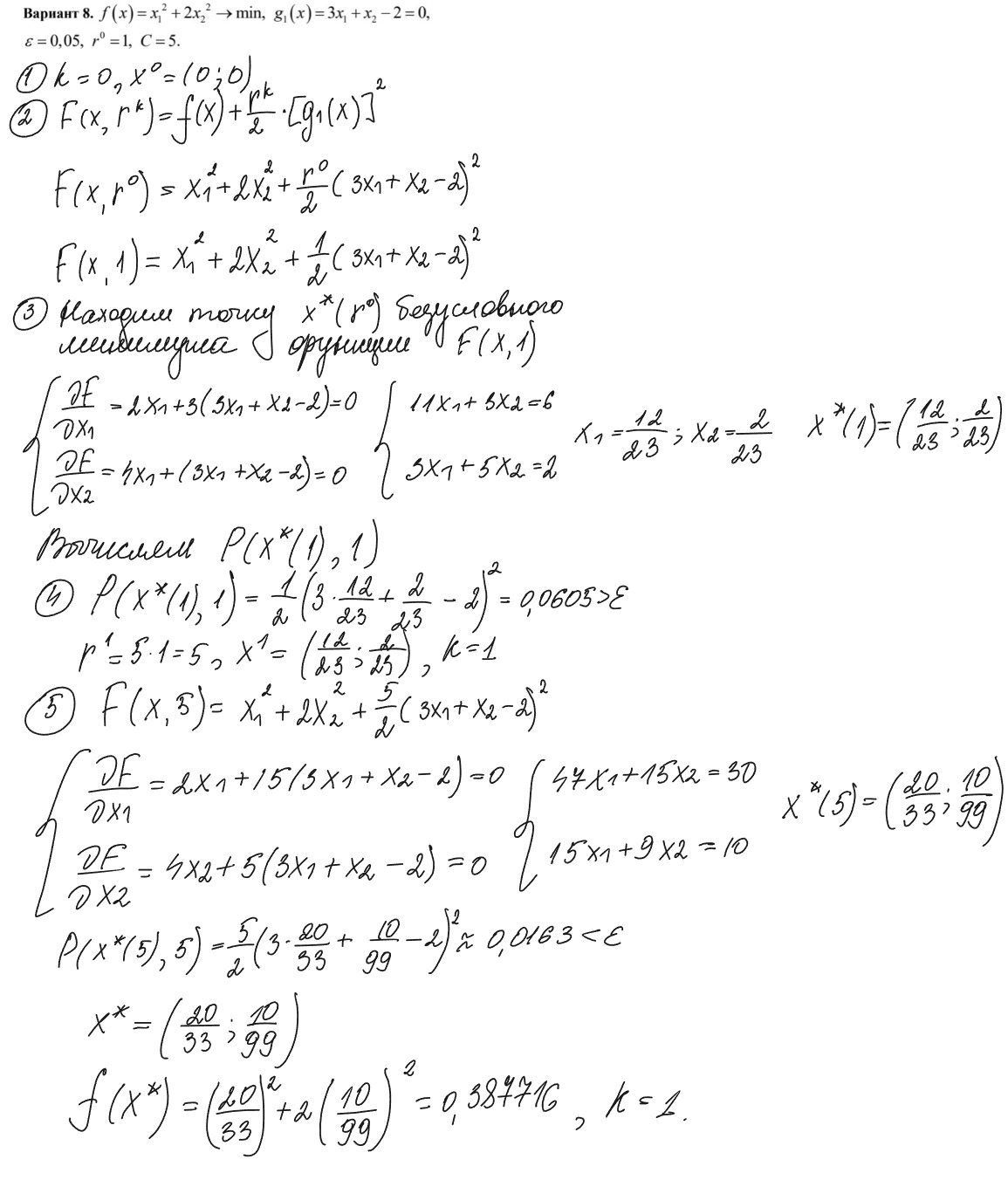
|  |
| --- |
| Лабораторная работа №4 «Методы условной оптимизации» Вариант № 8 |

|  |  |
| --- | --- |
| Направление: | 09.03.04 Программная инженерия |
| Профиль подготовки: | Системный анализ и проектирование программных комплексов |

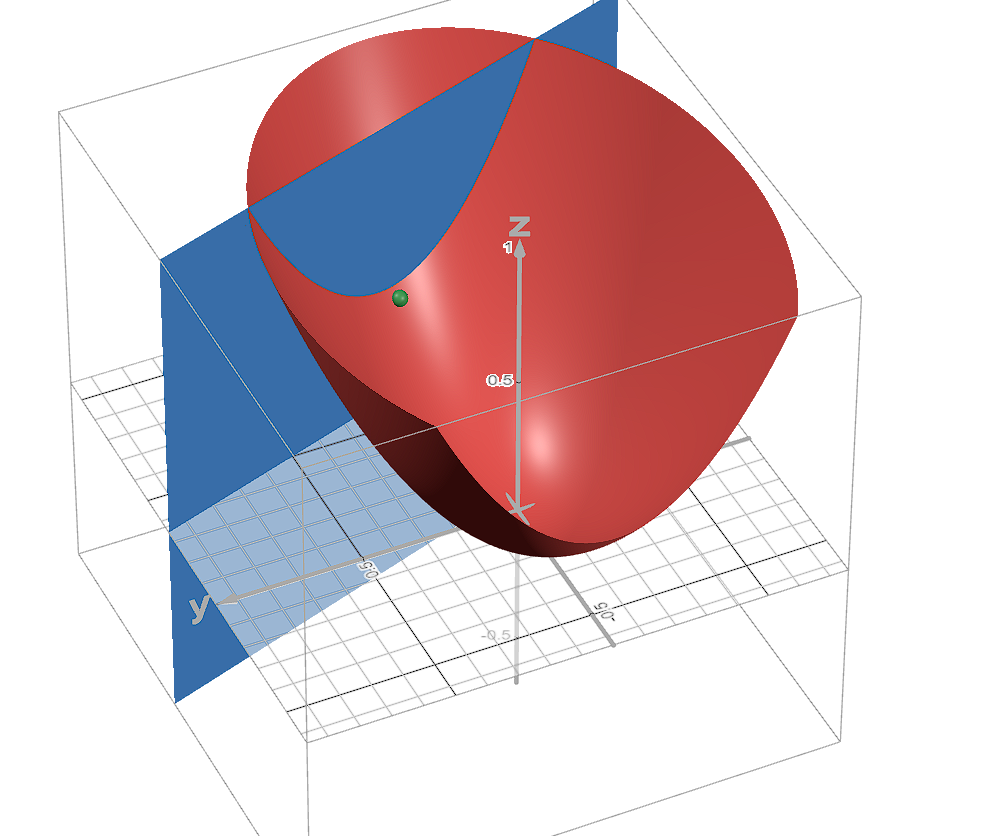
|  |  |  |  |
| --- | --- | --- | --- |
| Отчет сдан «20» мая 2025 г. | | | |
|  |  |  |  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  | | | |
| Преподаватель | Волкова О. Р., к.т.н. |  |  |
|  | *(Ф.И.О., должность, степень, звание.)* |  | *(подпись)* |

МОСКВА 2025

1. Найти локальный минимум целевой функции при заданном ограничении из индивидуального задания методом штрафов.

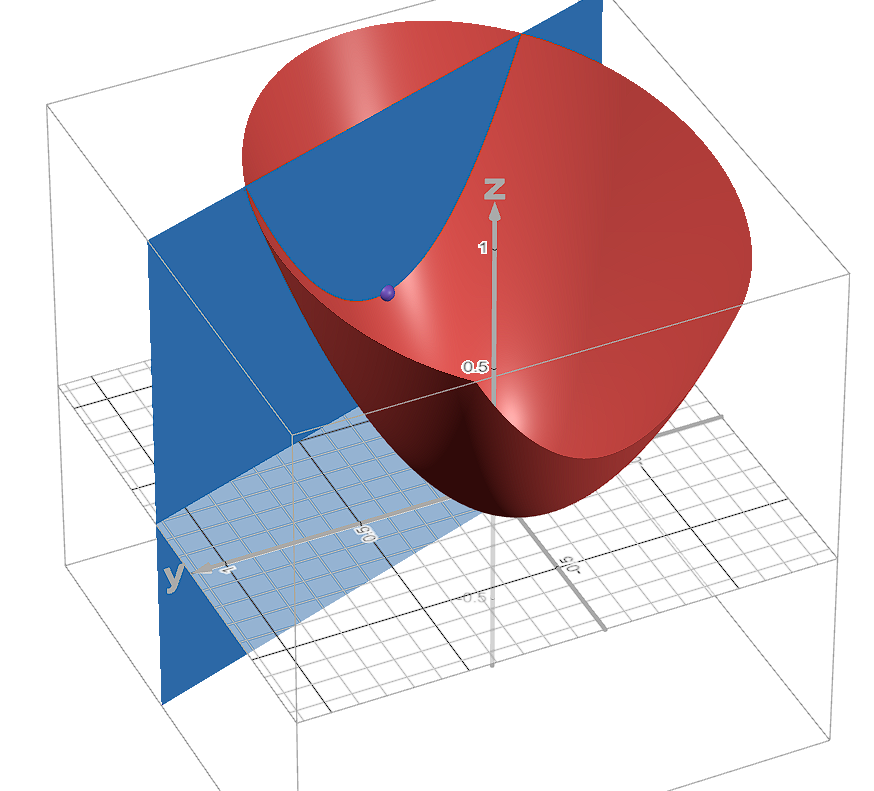


1. Построить графическое изображение целевой функции и ограничения, отметить найденное решение.



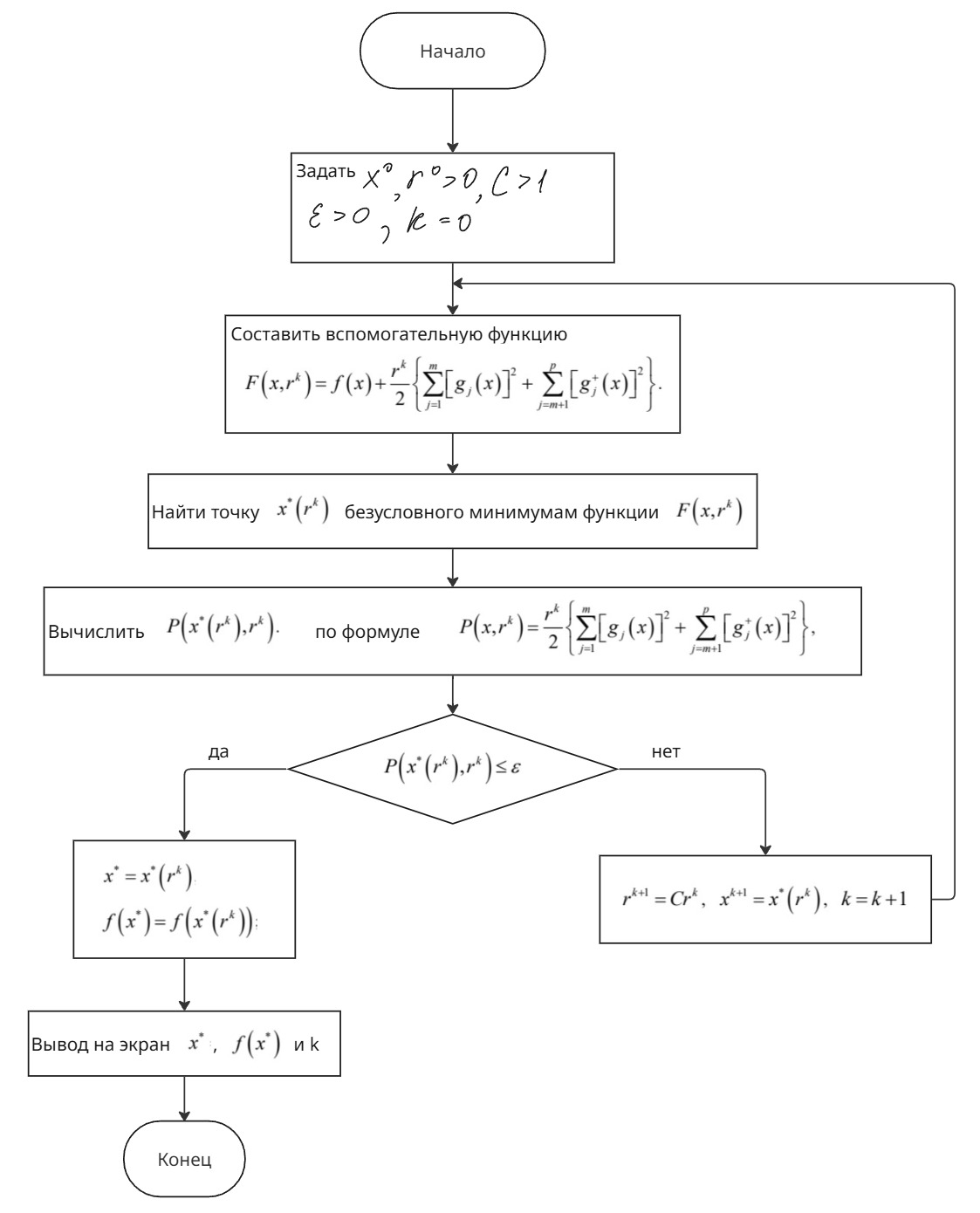
Красным выделен график целевой функции, синим – график ограничения, зеленым найденная точка из предыдущего задания.

Для сравнения, на следующем графике будет показана точка, которая действительно является условным минимумом функции (точка отмечена фиолетовым).

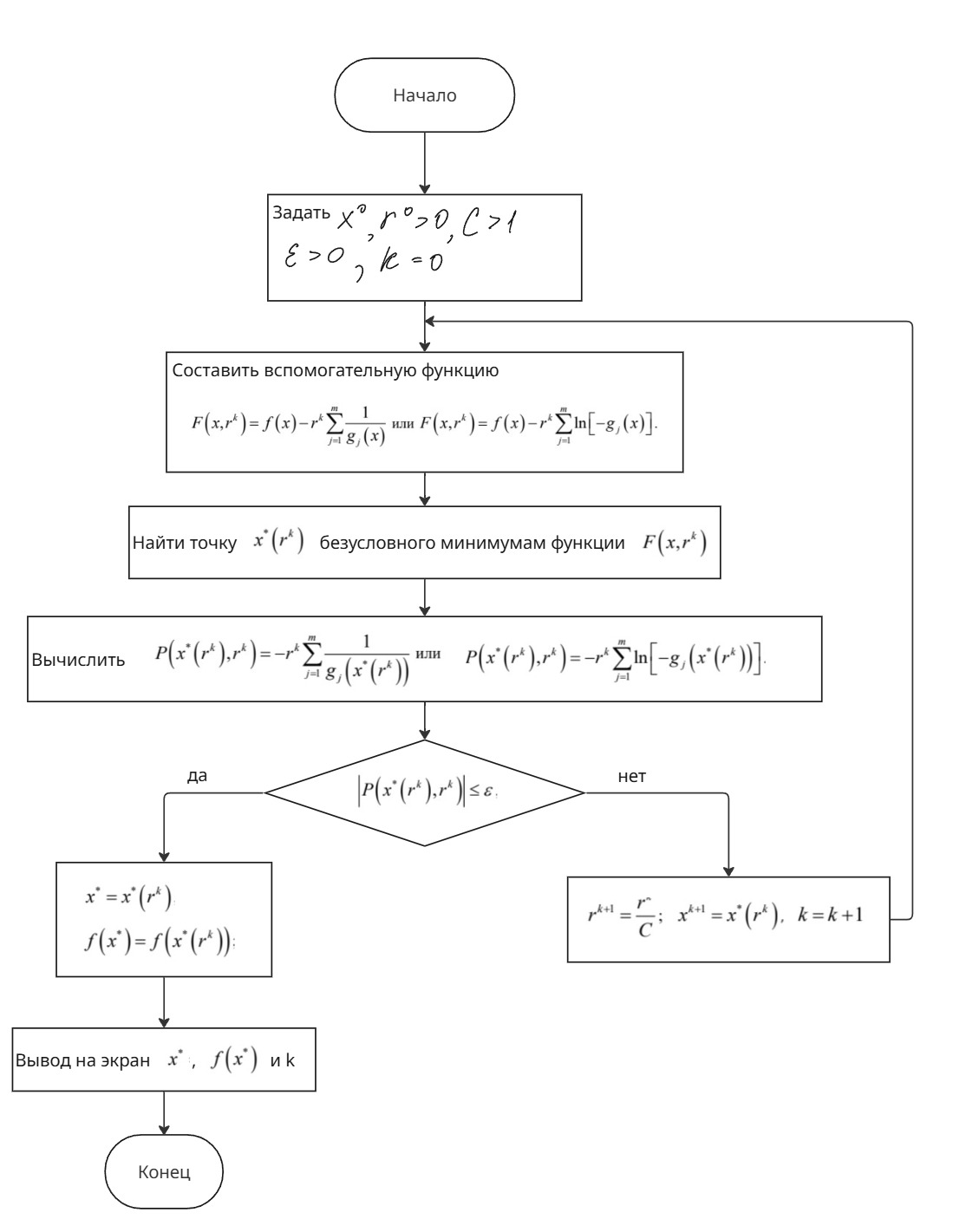


Расхождение возникло из-за ограничений метода (конечное *rk* и малое число итераций). Для точного решения нужно увеличить точность метода штрафов. Например, При r0=100, C=5, ε=10^−6 метод сойдется к (12/19; 2/19​)

1. Построить блок-схему алгоритма поиска локального минимума целевой функции для метода штрафов.



1. Найти локальный минимум целевой функции из индивидуального задания методом барьерных функций (индивидуальное задание взять у преподавателя).
2. Построить блок-схему алгоритма поиска локального минимума целевой функции для метода барьерных функций.



Создать веб-приложение для поиска минимума функции с помощью методов условной оптимизации.

**import** numpy **as** np

**import** sympy **as** sp

**from** string **import** ascii\_letters

**from** scipy**.**optimize **import** minimize

class ExtremumFinder**:**

**def** \_\_init\_\_(*self*)**:**

*self***.**funcF **=** None

*self***.**funcG **=** None

**def** SetFuncF(*self***,** **func**)**:**

        variable **=** list()

**if** '=' **in** func**:**

            func **=** func[func**.**index('=') **+** 1**:**]

        func **=** func**.**replace("^"**,** "\*\*")

        func **=** func**.**replace("e"**,** "2.71828182845904")

*self***.**funcF **=** func

**for** letter **in** ascii\_letters**:**

**if** letter **in** func**:**

                variable**.**append(letter)

**if** len(variable) **==** 2**:**

**break**

*self***.**funcF **=** *self***.**funcF**.**replace(variable[0]**,** "(x)")

*self***.**funcF **=** *self***.**funcF**.**replace(variable[1]**,** "(y)")

**def** SetFuncG(*self***,** **func**)**:**

        variable **=** list()

**if** '=' **in** func**:**

            func **=** func[func**.**index('=') **+** 1**:**]

        func **=** func**.**replace("^"**,** "\*\*")

        func **=** func**.**replace("e"**,** "2.71828182845904")

*self***.**funcG **=** func

**for** letter **in** ascii\_letters**:**

**if** letter **in** func**:**

                variable**.**append(letter)

**if** len(variable) **==** 2**:**

**break**

*self***.**funcG **=** *self***.**funcG**.**replace(variable[0]**,** "(x)")

*self***.**funcG **=** *self***.**funcG**.**replace(variable[1]**,** "(y)")

**def** penalty\_method\_search(*self***,** **x0,** **e,** **r0,** **C,** **M=**100)**:**

        x\_syms **=** sp**.**symbols('x y')

        f\_expr **=** sp**.**sympify(*self***.**funcF)

        g\_expr **=** sp**.**sympify(*self***.**funcG)

        f\_func **=** sp**.**lambdify(x\_syms**,** f\_expr**,** **modules=**['numpy'])

        g\_func **=** sp**.**lambdify(x\_syms**,** g\_expr**,** **modules=**['numpy'])

**def** f(**x**)**:**

**return** f\_func(**\***x)

**def** g(**x**)**:**

**return** g\_func(**\***x)

**def** penalty(**x,** **r**)**:**

**return** r **\*** g(x)**\*\***2

**def** F(**x,** **r**)**:**

**return** f(x) **+** penalty(x**,** r)

        r **=** r0

        x **=** np**.**array(x0)

**for** k **in** range(M)**:**

            result **=** minimize(F**,** x**,** **args=**(r**,**)**,** **method=**'BFGS')

            x\_new **=** result**.**x

            P\_val **=** penalty(x\_new**,** r)

**if** P\_val **<=** e**:**

**break**

            r **\*=** C

            x **=** x\_new

**return** x\_new**.**tolist()**,** round(float(f(x\_new))**,** 5)**,** k **+** 1

**def** barrier\_method\_search(*self***,** **x0,** **e,** **r0,** **C,** **method,** **m=**100)**:**

        x\_syms **=** sp**.**symbols('x y')

        f\_expr **=** sp**.**sympify(*self***.**funcF)

        g\_expr **=** sp**.**sympify(*self***.**funcG)

        f\_func **=** sp**.**lambdify(x\_syms**,** f\_expr**,** **modules=**'numpy')

        g\_func **=** sp**.**lambdify(x\_syms**,** g\_expr**,** **modules=**'numpy')

**def** f(**x**)**:**

**return** f\_func(**\***x)

**def** g(**x**)**:**

**return** g\_func(**\***x)

**def** barrier(**x,** **r**)**:**

            gx **=** g(x)

**if** method **==** "inverse"**:**

**if** gx **>=** 0**:**

**return** np**.**inf

**return** **-**r **/** gx

**elif** method **==** "log"**:**

**if** gx **>=** 0**:**

**return** np**.**inf

**return** **-**r **\*** np**.**log(**-**gx)

**def** F(**x,** **r**)**:**

**return** f(x) **+** barrier(x**,** r)

        r **=** r0

        x **=** np**.**array(x0)

**for** k **in** range(m)**:**

            result **=** minimize(F**,** x**,** **args=**(r**,**)**,** **method=**'BFGS')

            x\_new **=** result**.**x

            penalty\_value **=** abs(barrier(x\_new**,** r))

**if** penalty\_value **<=** e**:**

**break**

            r **/=** C

            x **=** x\_new

**return** x\_new**,** round(f(x\_new)**,** 5)**,** k **+** 1

**def** test()**:**

    finder **=** ExtremumFinder()

    finder**.**SetFuncF("x\*\*2 + y\*\*2 - 6\*x + 2\*y + 25")

    finder**.**SetFuncG("x + 10\*y - 7")

    x **=** [0**,** 0]

    e **=** 8

    r0 **=** 0.05

    C **=** 0.5

    result **=** finder**.**penalty\_method\_search(x**,** e**,** r0**,** C)

    print(result)

*# test()*

Скриншоты

