Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 18

Сравнительный анализ эффективности численных методов второго порядка для поиска безусловного экстремума
Тема

Преподаватель			В. В. Тынченко
		Подпись, дата	Инициалы, Фамилия
Студент	КИ19-17/1Б, №031939174		А. К. Никитин
	Номер группы, зачетной книжки	Подпись, дата	Инициалы, Фамилия

Красноярск 2021

1 Постановка задачи

На основании результатов выполнения практических работ модуля "Численные методы второго порядка для поиска безусловного экстремума" сравнить реализованные алгоритмы по точности и скорости решения задач оптимизации, варьируя параметры алгоритмов. Для проведения вычислительных экспериментов самостоятельно выбрать 3 целевые функции и интервалы неопределенности, интересные с точки зрения исследования. Результаты вычислительных экспериментов представить в табличном виде, прокомментировать их и сделать обоснованный вывод об особенностях работы исследуемых алгоритмов и их эффективности на различных целевых функциях.

2 Функции для исследования

Ниже представлен список функций нескольких переменных.

1. $f(x_1, x_2) = 3x_1^2 + x_1 x_2 + 2x_2^2 - x_1 - 4 x_2$
2. $f(x_1, x_2) = (x_1 - 4)^2 + (x_2 - 1)^2$
3. $f(x_1, x_2) = 2x_1^2 -2x_1 + x_1x_2 - x_2 + x_2^2$

3 Исходные тексты программ

На листинге 1 представлен код программы, проводящий анализ функций.

Листинг 1 – Анализ функций нескольких переменных

```
from newton import newton_method
from newton_raphson import newton_raphson_method
from marquardt import marquardt_method
from matplotlib import pyplot as plt
from const import Function
from sympy import Symbol
import numpy as np
from time import time

x1 = Symbol('x1')
x2 = Symbol('x2')

param_start = 0.01
```

Продолжение листинга 1

```
param_end = 3
param_step = 0.03
x0 = [-10, 10]

plt.rcParams["figure.figsize"] = (10, 5)


def assess(method, *args, **kwargs):
    t1 = time()
    extremum = method(*args, **kwargs)[0]
    t2 = time()
    return extremum, t2 - t1


def compare(func, exact_extremum):
    deltas = {'newton': [], 'newton_raphson': [], 'marquardt': []}
    performances = {'newton': [], 'newton_raphson': [], 'marquardt': []}
    names = tuple(deltas.keys())

    epsilons = np.arange(param_start, param_end, param_step)

    for param in np.arange(param_start, param_end, param_step):
        methods = (newton_method, newton_raphson_method, marquardt_method)
        results = [assess(method, func, x0, epsilon1=param, epsilon2=param) for
method in methods]
        [deltas[key].append(sum([abs(exact_extremum[i] - result[0][i]) for i in
range(len(exact_extremum))]))
         for key, result in zip(names, results)]
        [performances[key].append(result[1]) for key, result in zip(names,
results)]

    fig, ax = plt.subplots(nrows=1, ncols=2)
    colors = [np.random.rand(3, ) for _ in range(len(methods))]
    [ax[0].plot(epsilons, delta, c=color) for delta, color in zip(deltas.values(),
colors)]

    ax[0].set_xlabel('epsilon')
    ax[0].set_ylabel('accuracy')

    [ax[1].plot(epsilons, iteration, c=color) for iteration, color in
zip(performances.values(), colors)]
```

Окончание листинга 1

```
    ax[1].set_xlabel('epsilon')
    ax[1].set_ylabel('time')

    ax[1].legend(names)
    plt.show()


if __name__ == '__main__':
    func1 = Function(3 * x1 ** 2 + x1 * x2 + 2 * x2 ** 2 - x1 - 4 * x2, (x1, x2))
    # func2 = Function((x1 - 4)**2 + (x2 - 1)**2, (x1, x2))
    # func3 = Function(2 * x1**2 - 2 * x1 + x1 * x2 - x2 + x2**2, (x1, x2))
    exact_extremum1 = [0., 1.]
    # exact_extremum2 = [4, 1]
    # exact_extremum3 = [3/7, 2/7]
    # compare(func1, exact_extremum1)
    # compare(func2, exact_extremum2)
    compare(func1, exact_extremum1)
```

4 Сравнительный анализ методов

На рисунках 1, 2, 3 представлены графики изменения точности и производительности методов Ньютона, Ньютона-Рафсона, Марквардта от параметра epsilon.

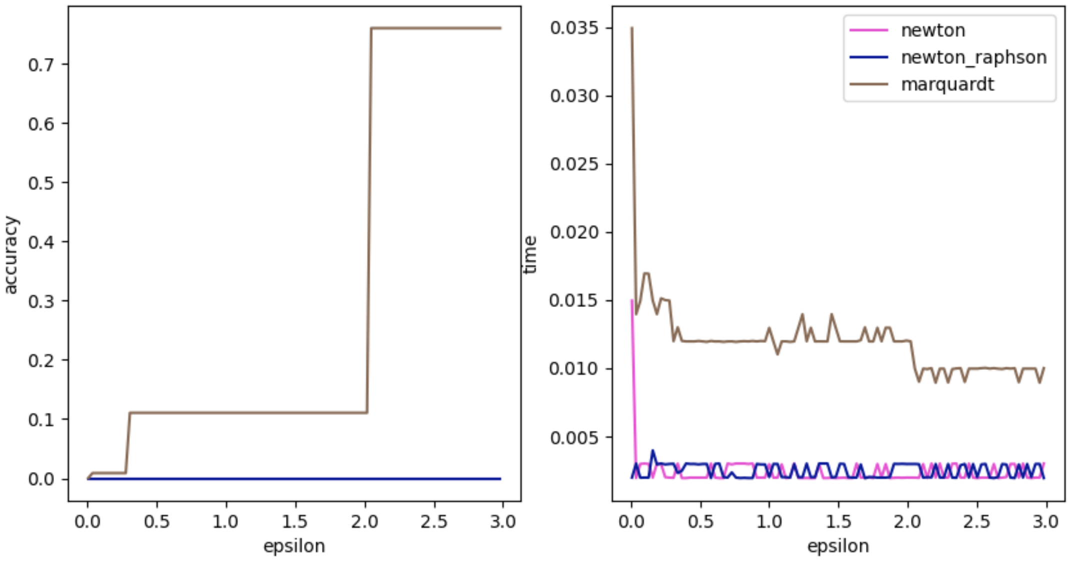

Рисунок 1 – Изменение точности алгоритмов от параметра epsilon функции $f(x_1, x_2) = 3x_1^2 + x_1 x_2 + 2x_2^2 - x_1 - 4 x_2$

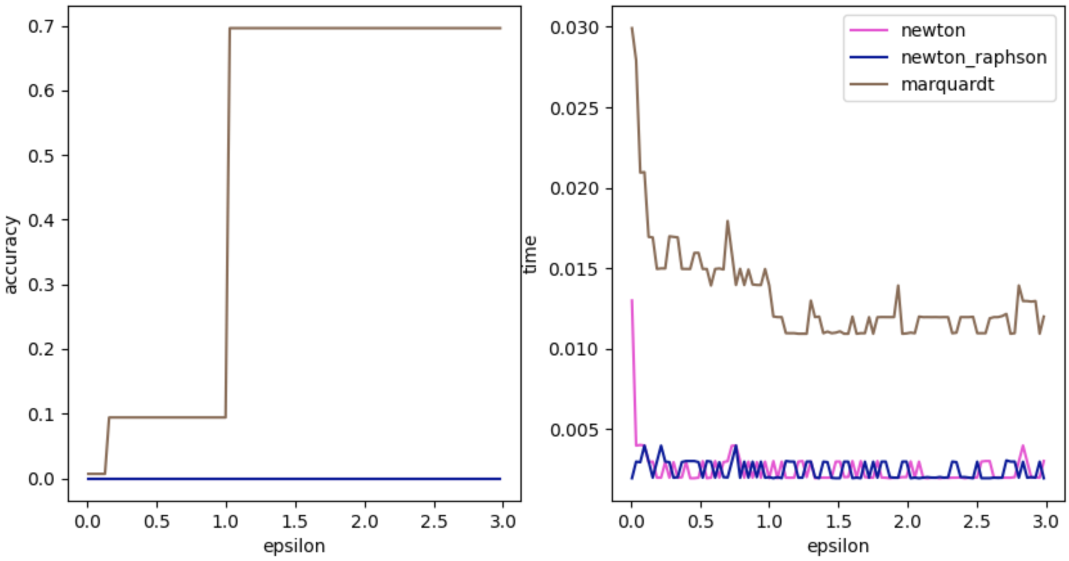

Рисунок 2 – Изменение точности алгоритмов от параметра epsilon функции $f(x_1, x_2) = (x_1 - 4)^2 + (x_2 - 1)^2$

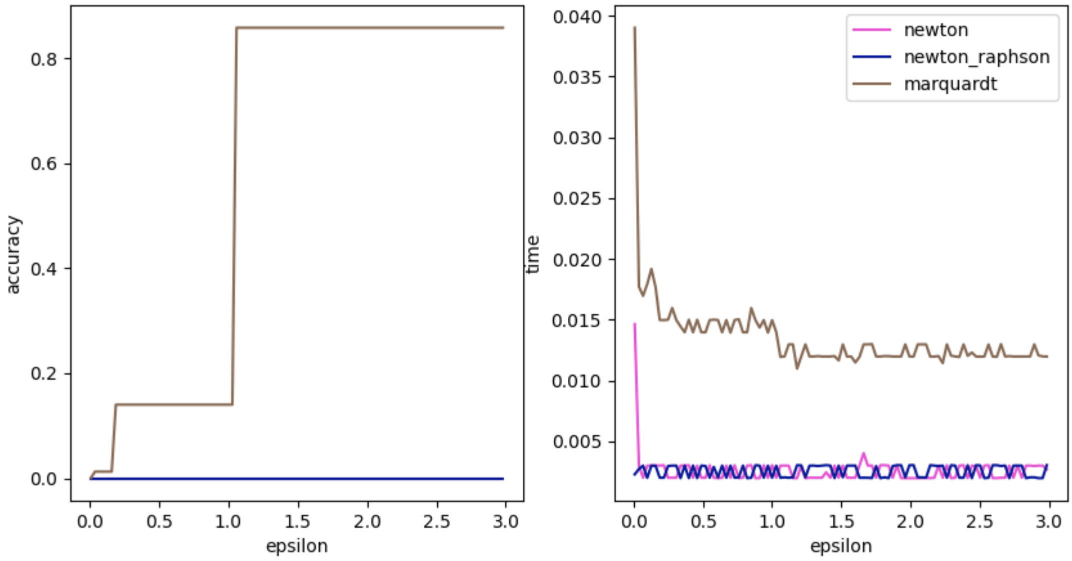

Рисунок 3 – Изменение точности алгоритмов от параметра epsilon функции $f(x_1, x_2) = 2x_1^2 - 2x_1 + x_1x_2 - x_2 + x_2^2$

5 Вывод

Таким образом, для минимизации функций нескольких переменных переменной наиболее эффективным по времени и точности оказался метод Ньютона -Рафсона.