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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 16

Метод Ньютона-Рафсона
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

Разработать программу, реализующую метод Ньютона-Рафсона.

Найти безусловный экстремум функции, выбранной в соответствии с заданием, с использованием разработанной программы.

Функция: $(x_2 + x_1 - 1)^2 + 2(x_1 - 2)^2 \to min$

2 Описание метода

Стратегия метода Ньютона-Рафсона состоит в построении последовательности точек $\{x_k\}$, $k = 0, 1, ...$, таких, что $f(x_{k+1}) < f(x_k)$, $k = 0, 1, ...$ Точки последовательности вычисляются по правилу

$x_{k+1} = x_k - t_k H -1(x_k) \nabla f(x_k)$, $k = 0, 1, ...$, (7.4)

где x_0 задается пользователем, а величина шага tk определяется из условия

Стратегия метода Ньютона состоит в построении последовательности точек $\{x_k\}$, $k = 0, 1, ...$, таких, что $f(x_{k+1}) < f(x_k)$, $k = 0, 1, ...$ Точки последовательности вычисляются по правилу

$x_{k+1} = x_k + d_k$, $k = 0, 1, ...$, (7.4)

где x_0 — задается пользователем, а направление спуска d_k определяется для каждого значения k по формуле

$$\varphi(t_k) = f\left(x^k - t_k H^{-1}\left(x^k\right) \nabla f\left(x^k\right)\right) \to \min_{t_k}. \quad (7.5)$$

Задача (7.5) может решаться либо аналитически с использованием необходимого условия минимума $\frac{d\varphi}{dt_k} = 0$ с последующей проверкой достаточного условия $\frac{d^2\varphi}{dt_k^2} > 0$, либо численно как задача.

$$\varphi(t_k) \to \min_{t_k \in [a,b]}, \quad (7,6)$$

где интервал $[a, b]$ задается пользователем.

Если функция $\varphi(t_k)$ достаточно сложна, то возможна ее замена полиномом $P(t_k)$ второй или третьей степени и тогда шаг t_k может быть определен из условия $\frac{dP}{dt_k} = 0$ при выполнении условия $\frac{d^2P}{dt_k^2} > 0$.

При численном решении задачи определения величины шага степень близости найденного значения t_k к оптимальному значению t_k^*, удовлетворяющему условиям $\frac{d\varphi}{dt_k} = 0, \frac{d^2\varphi}{dt_k^2} > 0,$ зависит от задания интервала $[a, b]$ и точности методов одномерной минимизации.

Построение последовательности $\{x_k\}$ заканчивается в точке x_k, для которой $\| \nabla f(x_k) \| < \varepsilon_1$, где ε_1— заданное число, или при $k \geq M$ (M — предельное число итераций), или при двукратном, одновременном выполнении двух неравенств $\| x_{k+1} - x_k\| < \varepsilon_2$, $|f(x_{k+1}) - f(x_k)| < \varepsilon_2$, где ε_2— малое положительное число. Вопрос о том, может ли точка x_k рассматриваться как найденное приближение искомой точки минимума, решается путем проведения дополнительного исследования, которое описано ниже.

3 Исходные тексты программ

На листинге 1 представлен код программы, реализующий задание.

Листинг 1 – Метод Ньютона-Рафсона

```
import numpy as np
from sympy.solvers import solve
from sympy import Symbol
from const import func

t = Symbol('t')


def matrix_positivity_check(arr):
    if arr.shape[0] != arr.shape[1]:
        raise ValueError('Matrix must be square!')

    return  all([np.linalg.det(arr[:i  +  1,  :i  +  1])  >  0  for  i  in
range(arr.shape[0])])


def calculate_step(f, x, d):
    t_function = f.calc(x + t * d)
    return solve(t_function.diff(t), t)[0]


def newton_method(f, x0, epsilon1=0.1, epsilon2=0.1, M=100):
    x_list = [np.array(x0).astype(float)]
    d_list = []
    k = 0
    while k < M:
        gradient = f.gradient_value(x_list[k])

        if np.linalg.norm(gradient) < epsilon1:
            return x_list[k], k + 1

        hessian = f.hessian(x_list[k])
        inv_hessian = np.linalg.inv(hessian)

        if matrix_positivity_check(inv_hessian):
            d_list.append(-inv_hessian    @    gradient.reshape((len(gradient),
1)).squeeze())
```

Окончание листинга 1

```
            step = 1
        else:
            d_list.append(-gradient)
            step = calculate_step(f, x_list[k], d_list[k])

        x_list.append(x_list[k] + step * d_list[k])

        if np.linalg.norm(x_list[k + 1] - x_list[k]) < epsilon2 \
                and abs(f.calc(x_list[k + 1]) - f.calc(x_list[k])) < epsilon2 \
                and len(x_list) > 2 \
                and np.linalg.norm(x_list[k] - x_list[k - 1]) < epsilon2 \
                and abs(f.calc(x_list[k]) - f.calc(x_list[k - 1])) < epsilon2:
            return x_list[k + 1], k + 1

        k += 1

    return x_list[-1], k


if __name__ == '__main__':
    print(newton_method(func, [0.5, 1], epsilon1=0.1, epsilon2=0.15, M=10))
```

4 Исследование влияния параметров метода на точность и скорость нахождения решения

В качестве гиперпараметров по умолчанию использовались следующие значения:

а) x0 = (-10, 10);

б) ε_1 = 0.1;

в) ε_2 = 0.1;

г) M = 100.

Из рисунка 1 можно заключить, что параметр epsilon в данном случае не влияет на работоспособность программы из-за малого количества итераций.

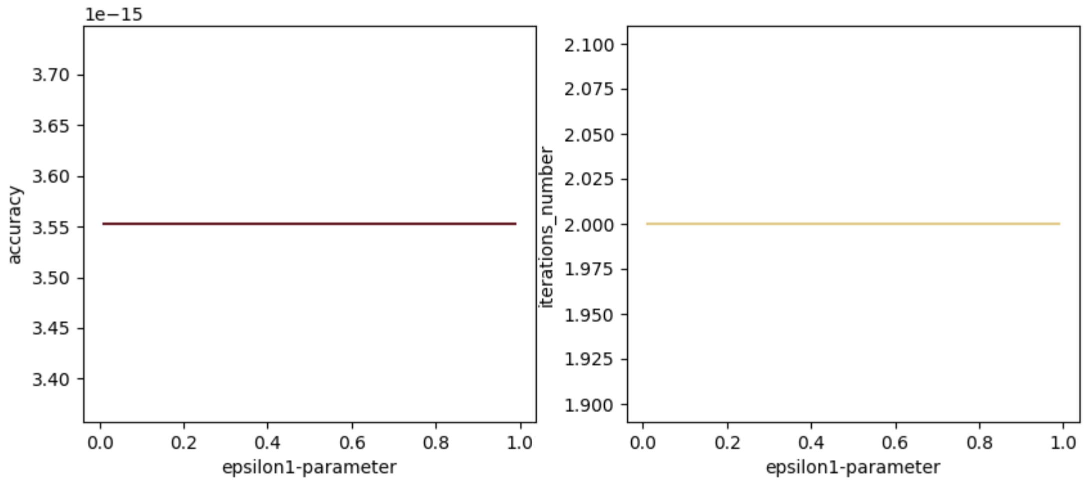

Рисунок 1 – Влияние параметра ε_1 на точность и производительность

Из рисунка 2 можно провести аналогичные рассуждения касательно параметра ε_2.

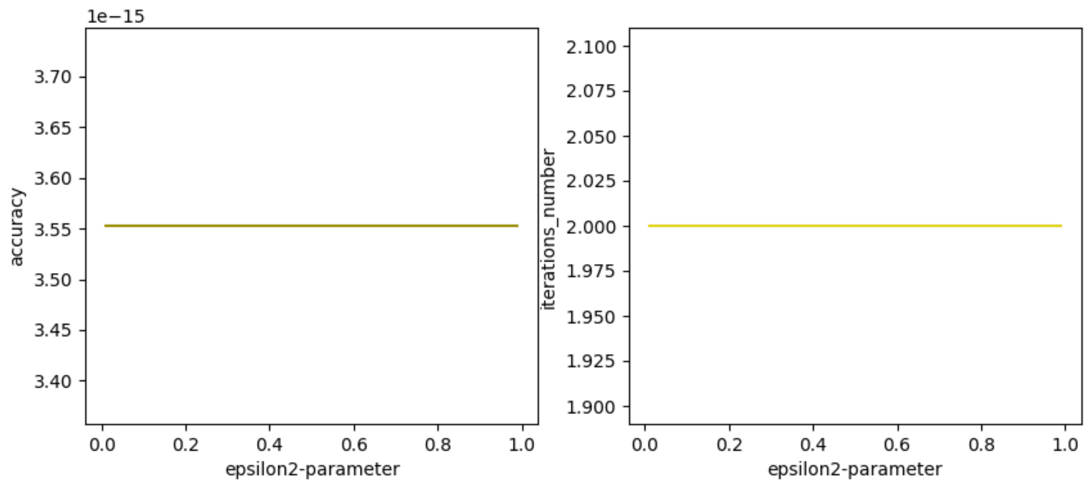

Рисунок 2 - Влияние параметра ε_2 на точность и производительность

Из рисунка 3 можно заключить, что параметр максимального количества шагов лучше не делать слишком небольшим, и что чем он больше, тем точнее, но менее производительнее, программа.

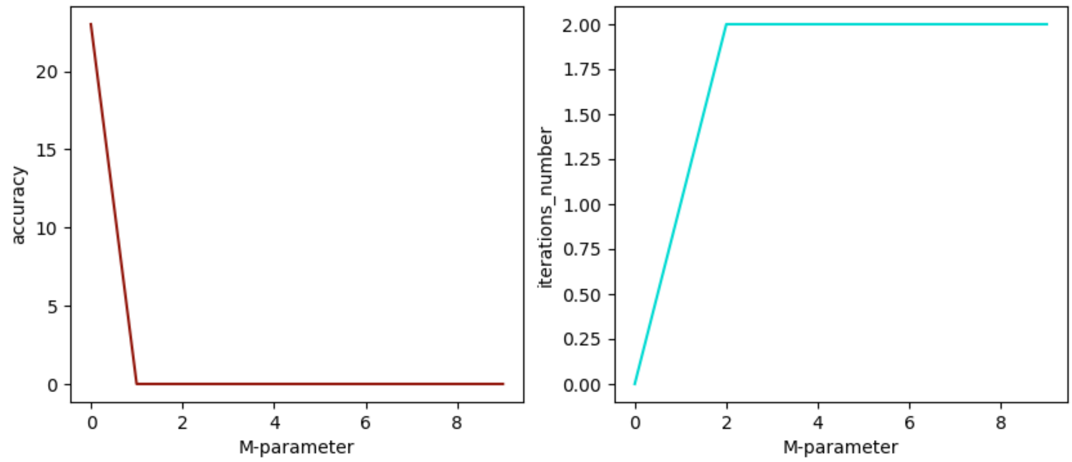

Рисунок 3 - Влияние параметра максимального количества шагов на точность и производительность

5 Вывод

В результате данной работы был реализован и проанализирован метод Ньютона-Рафсона для поиска локального минимума многомерной функции. Также были проанализированы гиперпараметры метода и их влияние на работу функции.