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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 13

Метод Дэвидона-Флетчера-Пауэлла
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 Дэвидона-Флетчера-Пауэлла.

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 Дэвидона-Флетчера-Пауэлла (Д-Ф-П) состоит в построении последовательности $\{x_k\}$, $k = 0, 1, ...$, таких, что $f(x_{k+1}) < f(x_k)$, $k = 0$, 1 , ... Точки последовательности $\{x_k\}$ вычисляются по правилу:

$x_{k+1} = x_k - t_k A_k \nabla f(x_k)$, $k = 0, 1, ...,$ (6.13)

где A_k есть матрица размера $n \times n$, которая вычисляется по правилу

$$A^{k+1} = A^k + A^k{}_c, \quad A^0 = E, \qquad (6.14)$$

$$A^k{}_c = \frac{\Delta x^k (\Delta x^k)^T}{(\Delta x^k)^T \Delta g^k} - \frac{A^k \Delta g^k (\Delta g^k)^T A^k}{(\Delta g^k)^T A^k \Delta g^k}, \qquad (6.15)$$

где $\Delta x^k = x^{k+1} - x^k$, $\Delta g^k = \nabla f(x^{k+1}) - \nabla f(x^k)$.

Точка x_0 задается пользователем, величина шага t_k определяется из условия

$$\varphi(t_k) = f\left(x^k - t_k A^k \nabla f(x^k)\right) \to \min_{t_k}. \qquad (6.16)$$

Решение задачи (6.16) может осуществляться как из условий $\frac{d\varphi}{dt_k} = 0, \frac{d^2\varphi}{dt_k^2} > 0,$ или из условий $\frac{dP}{dt_k} = 0, \frac{d^2P}{dt_k^2} > 0,$ где $P(t_k)$ — полином, аппроксимирующий функцию $\varphi(t_k)$, так и численно, т. е. путем поиска решения задачи $\varphi(t_k) \to \min_{t_k \in [a, b]}$ методами одномерной минимизации.

Формулы (6.14), (6.15) при аналитическом решении задачи (6.16) обеспечивают построение последовательности $\{A_k\}$ положительно определенных матриц, таких, что $A_k \to H - 1(x^*)$ при $k \to \infty$. Следствием этого для квадратичной функции $f(x)=\frac{1}{2}(Hx,x)+(b,x),\ H>0,$ является тот факт, что направления d_k, $k = 0, 1, ...$, будут H -сопряженными и, следовательно, алгоритм Д-Ф-П сойдется не более чем за n шагов.

Для неквадратичных функций $f(x)$ алгоритм перестает быть конечным и его сходимость зависит от точности решения задачи (6.16). Глобальную сходимость алгоритма можно гарантировать лишь при его обновлении через каждые n шагов, т. е. когда в формуле (6.13)

$$A^k = \begin{cases} E,\ k \in J;\ J = \{0, n, 2n, \ldots\}, \\ A^{k-1} + A_c^{k-1},\ k \notin J. \end{cases}$$

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.

3 Исходные тексты программ

На листинге 1 представлен код программы, реализующий задание.

Листинг 1 – Метод Дэвидона-Флетчера-Пауэлла

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

t = Symbol('t')


def generate_n_basis_vectors(n):
    return np.array([[1. if k == j else 0. for k in range(n)] for j in range(n)])


def calculate_step(f, x, a, gradient):
    t_function = f.calc(x - t * np.matmul(a, gradient))
    return solve(t_function.diff(t), t)[0]


def calculate_a(f, x_list, a, k):
    prev_gradient, gradient = f.gradient_value(x_list[k - 1]),
f.gradient_value(x_list[k])
    dx = x_list[k] - x_list[k - 1]
    dg = gradient - prev_gradient

    a_first_part = np.matmul(np.transpose([dx]), [dx]) / np.matmul([dx],
np.transpose([dg]))
    a_second_part_numerator = np.matmul(np.matmul(np.matmul(a,
np.transpose([dg])), [dg]), a)
    a_second_part_denominator = np.matmul(np.matmul([dg], a), np.transpose([dg]))

    return a_first_part - a_second_part_numerator / a_second_part_denominator


def devidon_fletcher_powell_method(f, x0, epsilon1=0.1, epsilon2=0.1, M=100):
    x_list = [np.array(x0).astype(float)]
    k = 0
    a = generate_n_basis_vectors(2)
    while k < M:
```

Окончание листинга 1

```
        gradient = f.gradient_value(x_list[k])

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

        if k != 0:
            a += calculate_a(f, x_list, a, k)

        step = float(calculate_step(f, x_list[k], a, gradient))
        d = - np.matmul(a, gradient)

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

        # print(x_list[k], a, step, sep='\n', end='\n\n')

        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(devidon_fletcher_powell_method(assess_func, [-10, 10], 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) $\varepsilon_1 = 0.1$;

в) $\varepsilon_2 = 0.1$;

г) M = 100.

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

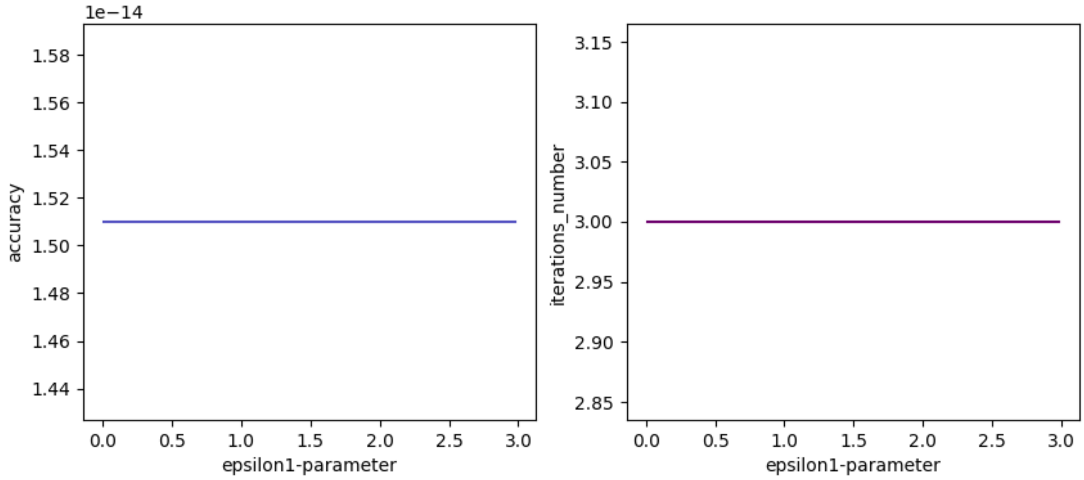

Рисунок 1 – Влияние параметра ε_1 на точность и производительность

Из рисунка 2 можно провести аналогичные рассуждения касательно параметра ε_2.

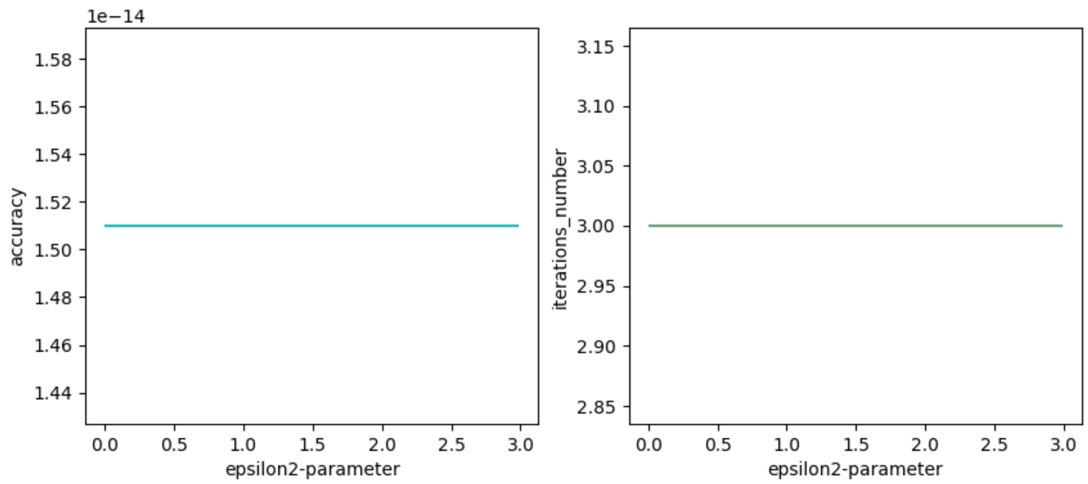

Рисунок 2 - Влияние параметра ε_2 на точность и производительность

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.

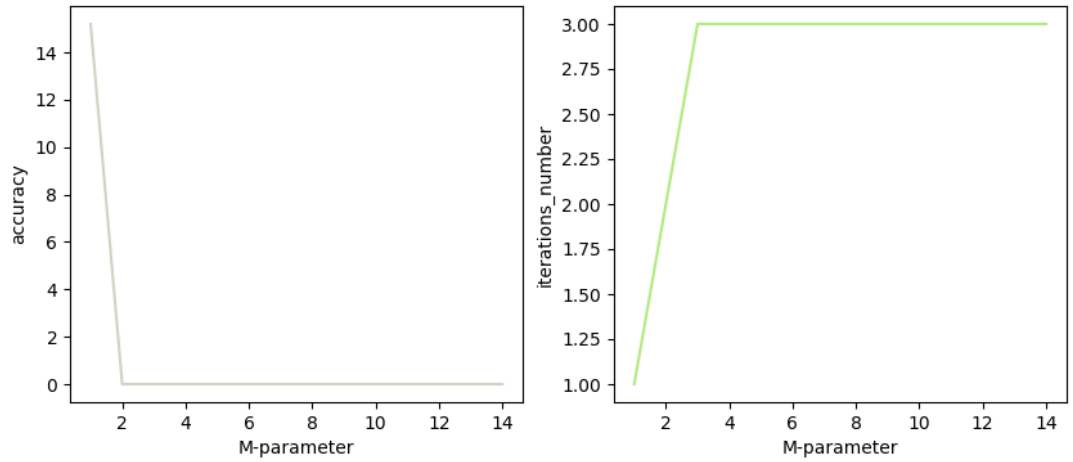

Рисунок 3 - Влияние параметра максимального количества шагов на точность и производительность

5 Вывод

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