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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 12

Метод Флетчера-Ривса
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 Флетчера-Ривса.

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

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

2 Описание метода

Стратегия метода Флетчера-Ривса состоит в построении последовательности точек { xk}, k = 0, 1, ..., таких, что f (xk+ 1) < f (xk), k = 0, l , ... Точки последовательности { xk} вычисляются по правилу:

$$x^{k+1} = x^k + t_k d^k, \; k = 0,1,\ldots; \qquad (6.7)$$

$$d^k = -\nabla f\left(x^k\right) + \beta_{k-1} d^{k-1}; \qquad (6.8)$$

$$d^0 = -\nabla f\left(x^0\right); \qquad (6.9)$$

$$\beta_{k-1} = \frac{\left\| \nabla f\left(x^k\right) \right\|^2}{\left\| \nabla f\left(x^{k-1}\right) \right\|^2}. \qquad (6.10)$$

Точка x 0задается пользователем, величина шага t_k определяется для каждого значения k из условия

$$\varphi(t_k) = f\left(x^k + t_k d^k\right) \rightarrow \min_{t_k}. \qquad (6.11)$$

Решение задачи одномерной минимизации (6.11) может осуществляться либо из условия $\frac{d\varphi}{dt_k} = 0, \frac{d^2\varphi}{dt_k^2} > 0,$ либо численно, с использованием методов одномерной минимизации, когда решается задача

$$\varphi(t_k) \rightarrow \min_{t_k \in [a,b]}. \qquad (6.12)$$

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.

Вычисление величины βk–1по формуле (6.10) обеспечивает для квадратичной формы $f(x)=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}x_ix_j$ построение последовательности H-сопряженных направлений $d_0, d_1, ..., d_k, ...$, для которых $(d_j, H_{di}) = 0$, $\forall\ i, j = 0, 1, ..., k$; $i \neq j$. При этом в точках последовательности $\{x_k\}$ градиенты функции $f(x)$ взаимно перпендикулярны, т. е. $\left(\nabla f\left(x^{k+1}\right), \nabla f\left(x^k\right)\right)=0,\ k=0,1,\ldots$

Для квадратичных функций $f(x)$ с матрицей $H > 0$ метод Флетчера-Ривса является конечным и сходится за число шагов, не превышающее n-размерность вектора x.

При минимизации неквадратичных функций метод не является конечным, при этом следует отметить, что погрешности в решении задачи (6.11) приводят к нарушению не только перпендикулярности градиентов, но и H-сопряженности направлений. Для неквадратичных функций, как правило, используется *алгоритм Лолака -Рибьера*, когда в формулах (6.7)–(6.9) величина β_{k-1}вычисляется следующим образом:

$$\beta_{k-1}=\begin{cases}\dfrac{\left(\nabla f\left(x^k\right),\left[\nabla f\left(x^k\right)-\nabla f\left(x^{k-1}\right)\right]\right)}{\left\|\nabla f\left(x^{k-1}\right)\right\|^2}, & k\notin J,\\ 0, \quad k\in J,\end{cases}$$

где $J = \{0, n, 2n, ...\}$. В отличие от алгоритма Флетчера-Ривса алгоритм Полака-Рибьера предусматривает использование итерации наискорейшего градиентного спуска через каждые n шагов. Построение последовательности $\{x_k\}$ заканчивается в точке, 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 Флетчера-Ривса

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

t = Symbol('t')


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


def steepest_gradient_descent_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

        if k != 0:
            prev_gradient = f.gradient_value(x_list[k - 1])
            beta = np.linalg.norm(gradient)**2 / np.linalg.norm(prev_gradient)**2
            d = -gradient + beta * d_list[k - 1]
        else:
            d = -gradient

        d_list.append(d)

        step = float(calculate_step(f, x_list[k], d_list[-1]))

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

        if np.linalg.norm(x_list[k + 1] - x_list[k]) < epsilon2 \
```

Окончание листинга 1

```
                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(steepest_gradient_descent_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.

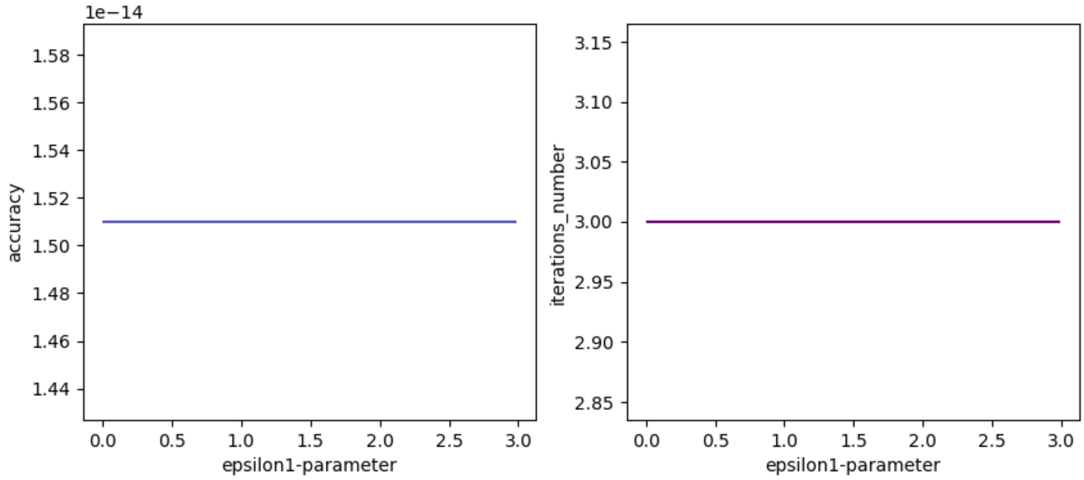

Рисунок 1 – Влияние параметра ε_1 на точность и производительность

Из рисунка 2 можно провести аналогичные рассуждения касательно параметра ε_2.

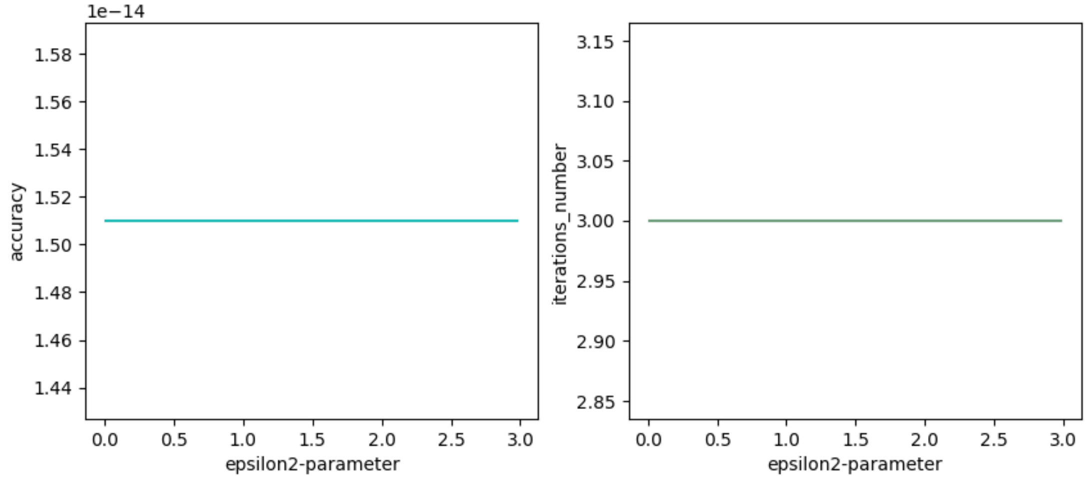

Рисунок 2 - Влияние параметра ε_2 на точность и производительность

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.

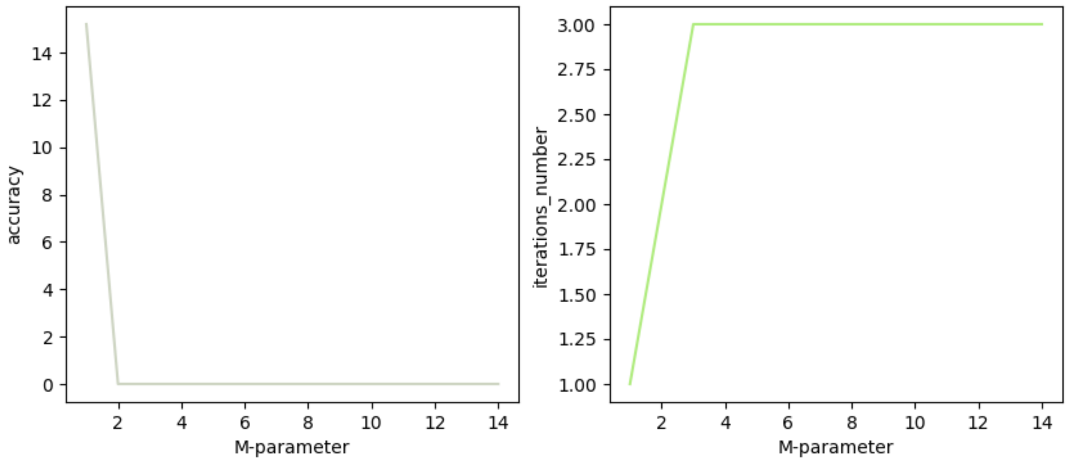

Рисунок 3 - Влияние параметра максимального количества шагов на точность и производительность

5 Вывод

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