Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Теоретическая информатика и компьютерные технологии»

Отчёт о лабораторной работе № 6

по курсу «Численные методы»

МЕТОД НАИСКОРЕЙШЕГО СПУСКА ПОИСКА МИНИМУМА ФУНКЦИИ МНОГИХ ПЕРЕМЕННЫХ.

Студент: К. Лозовска

Группа: ИУ9И-64Б

Преподаватель: А.Б.Домрачева

Москва, 2023

СОДЕРЖАНИЕ

ЦЕЛЬ И ПОСТАНОВКА ЗАДАЧИ

Цель: Аппроксимировать заданную функцию алгебраическими многочленами, применив метод наименьших квадратов.

Постановка задачи:

найти минимум функции двух переменных с точностью $\varepsilon = 0{,}001$, начиная итерации из точки X^o. Найти минимум аналитичности. Сравнить полученные результаты.

Тестовый пример:

$$f(x) = x_1^2 + 4x_1x_2 + 17x_2^2 + 5x_2, \qquad X^o = (0{,}0)$$

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Метод наискорейшего спуска является итерационным. Пусть для функции $f(x_1, \dots, x_n)$ на k-ом шаге имеем некоторое приближение к минимуму $X^k = (x_1^k, \dots x_n^k)$. Рассмотрим функцию одной переменной:

$$\varphi_k(t) = f\left(x_1^k - t\frac{\partial f}{\partial x_1}(X^k), \dots, x_n^k - t\frac{\partial f}{\partial x_n}(X^k)\right) = f\left(X^k - tgradf(x^k)\right),$$

где вектор $gradf(x^k) = \left(\frac{\partial f}{\partial x_1}(X^k), \dots, \frac{\partial f}{\partial x_n}(X^k)\right)$- градиент функции f в точке X^k. Функция $\varphi_k(t)$ представляет собой ограничение исходной функции $f(x)$ на прямую градиентного (наискорейшего) спуска, проходящую через точку k-ого приближения X^k.

Минимум функции $\varphi_k(t)$ можно найти любым методом одномерной оптимизации. Обозначим эту точку минимума через t^*. Теперь для следующего приближения к точке экстремума полагаем:

$$X^{k+1} = X^k - t^*gradf(X^k) = \left(x_1^k - t^*\frac{\partial f}{\partial x_1}(X^k), \dots, x_n^k - t^*\frac{\partial f}{\partial x_n}(X^k)\right).$$

Процесс поиска минимума продолжаем до тех пор, пока $\left\|gradf(X^k)\right\| = \max\limits_{1 \le i \le n} |\frac{\partial f}{\partial x_i}(X^k)|$ не станет меньше допустимой погрешности ε.

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Данный код реализует метод оптимизации для поиска минимального значения заданной функции. Сначала импортируется необходимая библиотека ***sympy*** и задается точность ***eps*** и переменные ***x*** и ***y***. Затем определяется функция ***f***, которая возвращает значение заданной функции. Далее определяется функция ***analytical_min***, которая находит координаты минимума функции ***f***, используя частные производные и решение системы уравнений.

```
from sympy import *

eps = 0.001
x, y = symbols('x y')

def f():
    return x**2 + 4*x*y + 17*y**2 + 5*y

def analytical_min(f, variables):
    # Получаем частные производные функции f по переменным variables
    partial_derivatives = [diff(f, variable) for variable in variables]
    # Решаем систему уравнений, приравнивая частные производные к нулю
    solution = solve(partial_derivatives, variables)
    # Возвращаем координаты минимума
    return tuple(solution[variable] for variable in variables)
```

Затем выводятся значения функции ***f*** и ее частных производных по ***x*** и ***y***, а также вторых производных по ***x*** и ***y***. Далее задаются начальные значения переменных ***xk*** и ***yk***.

```
print('f: ', f())
fx = diff(f(), x)
fy = diff(f(), y)
print('df/dx: ', fx)
print('df/dy: ', fy)
print('d^2f/dx^2: ',diff(fx, x))
print('d^2f/dxdy: ',diff(fx, y))
print('d^2f/dy^2: ',diff(fy, y))
print()
k = 0
xk, yk = 0.0, 0.0
```

Запускается цикл ***while***, который будет выполняться до тех пор, пока максимальное значение модуля частных производных не станет меньше ***eps***.

Внутри цикла вычисляются значения ***phi1*** и ***phi2***, используя частные производные функции ***f***, и находим ***t_start***. Затем обновляются значения переменных ***xk*** и ***yk***, используя найденное значение ***t_start*** и частные производные функции ***f***.

После завершения цикла выводятся координаты минимума, найденные методом наискорейшего спуска и аналитически, а также разницу между ними.

```
while(max(abs(fx.subs({x: xk, y: yk})), abs(fy.subs({x: xk, y:yk}))) >= eps):
    phi1 = - (fx.subs({x: xk, y: yk}))**2 - (fy.subs({x: xk, y: yk}))**2
    phi2 = diff(fx, x).subs({x: xk, y: yk}) * (fx.subs({x: xk, y: yk}))**2 + \
           2 * diff(fx , y).subs({x: xk, y: yk}) * fx.subs({x: xk, y: yk}) * fy.subs({x: xk, y: yk}) \
           + diff(fy, y).subs({x: xk, y: yk}) * (fy.subs({x: xk, y: yk}))**2
    t_start = - phi1 / phi2
    xk = xk - t_start * fx.subs({x: xk, y: yk})
    yk = yk - t_start * fy.subs({x: xk, y: yk})
    k +=1

print(f'methods min {xk, yk}')
print(f'analytical min: {analytical_min(f(), [x,y])}')
print(f'delta: {abs(xk -analytical_min(f(), [x,y])[0]), abs(yk -analytical_min(f(), [x,y])[1])}')
```

РЕЗУЛЬТАТЫ

Для примера была использована следующая функция двух переменных:

$$f(x) = x_1^2 + 4x_1x_2 + 17x_2^2 + 5x_2, \qquad X^o = (0,0).$$

В результате работы реализованного метода были получены следующие значения:

- **Аналитический минимум:** *(5/13, -5/26) = (0.38461538, -0,19230769)*
- **Минимум, полученный методом наискорейшего спуска**: *(0.385073602103277, -0.192354289753889).*
- **Погрешность вычисления**: *(0.000458217487891877, 0.0000465974461962 8).*

ВЫВОДЫ

Для данной тестовой функции метод наискорейшего спуска может дать достаточно точный результат, если точка минимума находится достаточно близко к начальной точке (0,0) и значение eps достаточно мало. Однако, если точка минимума находится далеко от начальной точки, метод может сойтись к локальному минимуму вместо глобального минимума.

В данном случае, точность метода наискорейшего спуска зависит от того, насколько близко начальная точка (0,0) к точке минимума (5/13, -5/26). С учетом того, что delta (разница между методом и аналитическим решением) достаточно мала, можно сделать вывод, что метод наискорейшего спуска дал достаточно точный результат для данной функции и начальной точки.

В целом метод наискорейшего спуска может быть эффективным для функций, которые имеют гладкие и выпуклые поверхности, но может быть неэффективным для функций, которые имеют множество локальных минимумов или сильно вытянутые формы..