Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу «Численные методы»
«Численный метод градиентного спуска»

Студент: Ф.М. Шавандрин
Преподаватель: Д.Е. Пивоваров
Группа: М8О-308Б-19
Дата: 05.06.2022
Оценка:
Подпись:

Москва, 2022

Задача: создать алгоритм, который найдет максимальное значение по модулю минимума на заданном радиусе. Реализовать отображение результата на графике.
Вариант: метод градиентного спуска.

Описание

Градиентный спуск — метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента. Алгоритм: рабочая область функции (заданный интервал) разбивается на несколько точек, затем выбираются точки локальных минимумов. После этого все координаты передаются функции в качестве аргументов и выбирается аргумент, дающий наименьшее значение. Затем применяется метод градиентного спуска.

Основные понятия

Использовал приращение аргумента:

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}.$$

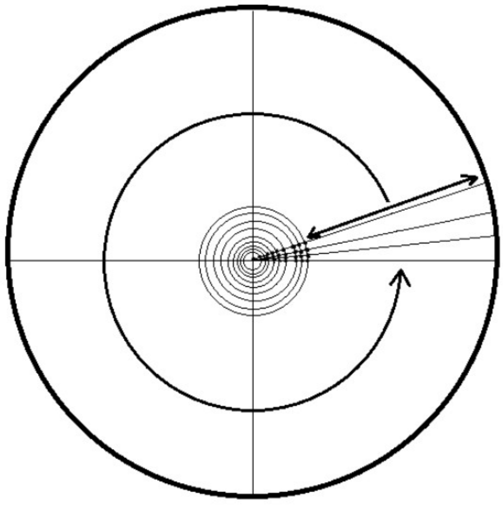
Для дальнейшего разбиения плоскости необходим поворот вектора:

$$R\mathbf{v} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}.$$

В градиентном спуске используется, как ни странно, градиент. Градиент — это

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} + \frac{\partial f}{\partial z}\mathbf{k}$$

Визуальная схема генерации точек



360 * arr_shape dots

, где arr_shape dots – число точек для разбиения рабочей области (заданного интервала).

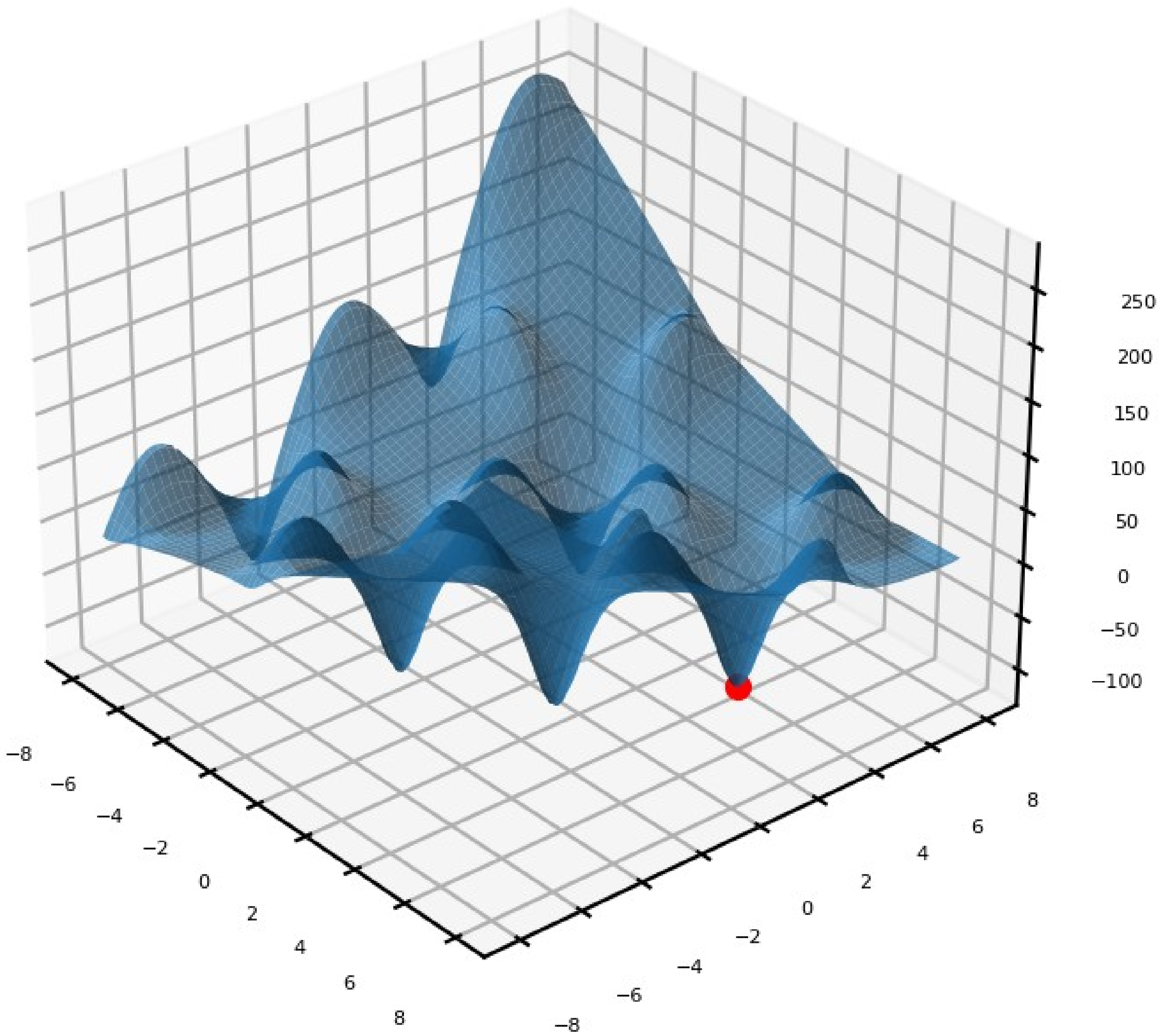
Исходный код

```
import numpy as np
import matplotlib.pyplot as plot


radius = 8                                        # радиус рабочей области
global_epsilon = 0.000000001
centre = (global_epsilon, global_epsilon)   # центр окружности
n = 100                                  # число точек для разбиения рабочей области
step = radius / n                        # шаг между двумя точками


# исходная функция
def differentiable_function(x, y):
    return np.sin(x) * np.exp((1 - np.cos(y)) ** 2) + \
        np.cos(y) * np.exp((1 - np.sin(x)) ** 2) + (x - y) ** 2


# поворот векторв
def rotate_vector(length, a):
    return length * np.cos(a), length * np.sin(a)


# производная по x
def derivative_x(epsilon, arg):
    return (differentiable_function(global_epsilon + epsilon, arg) -
            differentiable_function(epsilon, arg)) / global_epsilon


# производная по y
def derivative_y(epsilon, arg):
    return (differentiable_function(arg, epsilon + global_epsilon) -
            differentiable_function(arg, epsilon)) / global_epsilon


# функция для подсчёта массива приблизительных минимумов функции
def calculate_flip_points():
    flip_points = np.array([0, 0])
    points = np.zeros((360, n), dtype=bool)
    cx, cy = centre

    for i in range(n):
        for alpha in range(360):
            x, y = rotate_vector(step, alpha)
            x = x * i + cx
            y = y * i + cy
            points[alpha][i] = derivative_x(x, y) + derivative_y(y, x) > 0
            if not points[alpha][i - 1] and points[alpha][i]:
                flip_points = np.vstack((flip_points, np.array([alpha, i - 1])))

    return flip_points


# функция для выбора точки, значение функции в которой минимально
def pick_estimates(positions):
    vx, vy = rotate_vector(step, positions[1][0])
    cx, cy = centre
    best_x, best_y = cx + vx * positions[1][1], cy + vy * positions[1][1]

    for index in range(2, len(positions)):
```

```
        vx, vy = rotate_vector(step, positions[index][0])
        x, y = cx + vx * positions[index][1], cy + vy * positions[index][1]
         if differentiable_function(best_x, best_y) > differentiable_function(x,
y):
            best_x = x
            best_y = y

    for index in range(360):
        vx, vy = rotate_vector(step, index)
        x, y = cx + vx * (n - 1), cy + vy * (n - 1)
         if differentiable_function(best_x, best_y) > differentiable_function(x,
y):
            best_x = x
            best_y = y

    return best_x, best_y


# метод градиентного спуска
def gradient_descent(best_estimates, is_x):
    derivative = derivative_x if is_x else derivative_y
    best_x, best_y = best_estimates
    descent_step = step
    value = derivative(best_y, best_x)

    while abs(value) > global_epsilon:
        descent_step *= 0.95
        best_y = best_y - descent_step \
            if derivative(best_y, best_x) > 0 else best_y + descent_step
        value = derivative(best_y, best_x)

    return best_y, best_x


# функция нахождения точки минимума
def find_minimum():
    flip_points = calculate_flip_points()
    estimates = pick_estimates(flip_points)
    first_grad_desc = gradient_descent(estimates, False)
    return gradient_descent(first_grad_desc, True)


# функция сетки точек для построения
def get_grid(grid_step):
    samples = np.arange(-radius, radius, grid_step)
    x, y = np.meshgrid(samples, samples)
    return x, y, differentiable_function(x, y)


# функция отрисовки графика
def draw_chart(point, grid):
    point_x, point_y, point_z = point
    grid_x, grid_y, grid_z = grid
    plot.rcParams.update({
        'figure.figsize': (4, 4),
        'figure.dpi': 200,
        'xtick.labelsize': 4,
        'ytick.labelsize': 4
    })
    ax = plot.figure().add_subplot(111, projection='3d')
    ax.scatter(point_x, point_y, point_z, color='red')
    ax.plot_surface(grid_x, grid_y, grid_z, rstride=5, cstride=5, alpha=0.7)
    plot.show()
```

```
if __name__ == '__main__':
    # исходная функция задается в функции differentiable_function(x, y)
    min_x, min_y = find_minimum()
    minimum = (min_x, min_y, differentiable_function(min_x, min_y))
    draw_chart(minimum, get_grid(0.05))
```

Результат работы



Выводы

Выполнив курсовую работу по курсу «Численные методы», я познакомился с методом градиентного спуска, который находит точку локального минимума функции с помощью движения вдоль градиента. Освежил в памяти понятия приращение аргумента, поворот вектора, градиента. В процессе написания кода столкнулся с проблемой реализации функций для отображения заданной функции. Параллельно с методом градиентного спуска познакомился и с другими методами нахождения локального минимума, такие как метод наискорейшего градиентного спуска, метод покоординатного спуска, метод Гаусса-Зейделя и другие.

Литература

1. https://www.resolventa.ru/spr/matan/derivative.htm. [Электронный ресурс] Дата обращения: 04.06.2022

2. https://habr.com/ru/post/520090/. [Электронный ресурс] Дата обращения: 05.06.2022
3. http://www.machinelearning.ru/wiki/index.php?title=Метод_градиентного_спуска. [Электронный ресурс] Дата обращения: 05.06.2022