Федеральное государственное образовательное
бюджетное учреждение
высшего образования

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ
ФЕДЕРАЦИИ»

(Финансовый университет)

Факультет
информационных технологий и анализа больших данных
Кафедра «Прикладная математика и информатика »

Домашнее задание № 2

«Методы одномерной оптимизации»

Студенты группы ПМ19-3:
Филимонова Ю.М.
Корнева Т.А.
Косовская Т.П.
Дубровская А.А.
Кривоносова Д.В.

Руководитель:
Аксенов Дмитрий Андреевич

Москва 2022

Оглавление.

1. Постановка задачи

“Оптимизировать затраты заказчика на переработку отходов производства предприятия”

Необходимо в среде программирования Python реализовать функции, которые будут находить минимальные денежные затраты на переработку отходов производства предприятия

Обеспечить визуализацию с помощью 2-D графика функции с отмеченными точками локальных экстремумов.

Условия заказчика:

Заказчик занимается строительством коттеджей. После проведения всех работ ежегодно остаётся 24 тонны производственных отходов. Заказчику необходимо распределить производственные отходы между двумя заводами, которые могут утилизировать или переработать мусор от строительства коттеджей. Первому заводу необходимо заплатить $4*x^2$ денежных единиц за переработку x тонн мусора. Второму заводу надо заплатить за работу по утилизации и переработке x тонн мусора x^2 денежных единиц. Заказчику необходимо распределить весь объем производственных отходов между двумя заводами так, чтобы денежные затраты были минимальными. Стоит отметить, что заводам можно передавать не целое количество тонн производственных отходов.

Постановка задачи:

Всего после проведения строительных работ осталось 24 тонны производственных отходов. Пусть на первый завод будет направлено x тонн мусора (значение x может быть не целым), тогда на второй завод будет направлено (24 - x) тонн производственных отходов. За переработку

и утилизацию мусора на первом заводе необходимо заплатить $f_1(x) = 4*x^2$ денежных единиц, а на втором заводе - $f_2(x) = (24 - x)^2 = 576 – 48*x + x^2$. Получается, что заказчик должен будет заплатить
$f(x) = f_1(x) + f_2(x) = 5*x^2 - 48*x + 576$ денежных единиц. Необходимо найти минимум функции f(x) при $0 < x < 24$. Также необходимо найти количество тонн мусора, переданных на первый завод, т.е. найти значение x.

2. Математическая модель

Для решения выше поставленной задачи $f(x) = 5*x^2 - 48*x + 576$, где необходимо найти минимум f(x) при условии $0 < x < 24$, будем использовать методы одномерной оптимизации. В данной работе реализовано 4 метода оптимизации: метод золотого сечения, метод парабол, метод Брента, метод неточной оптимизации или алгоритм Бройдена-Флетчера-Гольдфарба-Шанно.

- **Метод золотого сечения** - метод поиска экстремума действительной функции одной переменной на заданном отрезке. В основе метода лежит принцип деления отрезка в пропорциях золотого сечения.
- **Метод парабол** является представителем группы методов, основанных на аппроксимации целевой функции некоторой более простой функцией (как правило полиномом), минимум которой можно легко найти. Данный метод обладает супер линейной скоростью сходимости. На каждой итерации этого метода строится квадратичный трехчлен, график которого (парабола) проходит через 3 выбранные точки графика функции f(x).

- **Метод Брента** является линейным поиском, который является гибридом поиска золотого сечения и квадратичной интерполяции. Метод Брента пытается сочетать лучшие функции обоих подходов.
- **Алгоритм Бройдена - Флетчера - Гольдфарба - Шанно** - итерационный метод численной оптимизации, предназначенный для нахождения локального максимума/минимума нелинейного функционала без ограничений. Один из наиболее широко применяемых квазиньютоновских методов. BFGS определяет направление спуска путем предварительного кондиционирования градиента информацией о кривизне.

3. Алгоритмы

В разделе описываются различные алгоритмы, выбранные для воплощения математической модели в виде законченного продукта.

3.1. Алгоритм 1

Поиск экстремума функции одной переменной методом золотого сечения.

3.1.1. Описание входных данных

На вход подается названия функция в аналитическом виде, границы области оптимизации и дополнительные ограничения: точность оптимизации по аргументу, максимальное число итераций, флаг «вывод промежуточных результатов» и флаг «запись промежуточных результатов в датасет».

3.1.2. Описание алгоритма решения

Сначала каждая из точек x_1 и x_2 делит исходный интервал (a,b) на две части так, что отношение целого к большей части равно отношении большей части к меньшей, т.е. равно так называемому "золотому отношению".

Получается, что длины интервалов (a, x_1) и (x_2, b) одинаковы и составляют 0,382 от длины (a,b). Другими словами, сначала вычисляются значения функций $f(x_1)$ и $f(x_2)$, где $x_1=a+0,382(b-a)$, $x_2=b+0,382(b-a)$.

Затем определяется новый интервал (a, x_1) или (x_2, b), в котором локализован минимум. Внутри полученного интервала находится новая точка (x_1 в случае 1) или (x_2 в случае 2), отстоящая от его конца на расстоянии, составляющем 0,382 от его длины. В этой точке рассчитывается значение f(x).

Затем вычисления повторяются, начиная с определения нового интервала, до тех пор, пока величина интервала неопределенности станет меньше или равна ε, где ε - заданное сколь угодно малое положительное число.

3.1.3 Описание выходных данных

Выходные данные представляют собой найденное значение координаты точки экстремума, значение функции в точке экстремума и график функции с точкой минимума.

3.2. Алгоритм 2

Поиск экстремума функции одной переменной методом парабол.

3.2.1. Описание входных данных

На вход подается названия функция в аналитическом виде, границы области оптимизации и дополнительные ограничения: точность оптимизации по аргументу, максимальное число итераций, флаг «вывод промежуточных результатов» и флаг «запись промежуточных результатов в датасет».

3.2.2. Описание алгоритма решения

Сначала раскладываем заданную функцию в ряд Тейлора в некоторой точке x_k, ограничиваясь тремя членами разложения.

Другими словами, аппроксимируем заданную функцию в точке x_k параболой. Выбираются три точки, в интервал которых входит точка минимума функции (границами интервала являются минимальная и максимальная точки).

Затем находятся коэффициенты аппроксимирующей параболы путем решения системы линейных уравнений.

После этого находится минимум параболы и проверяются неравенства для подтверждения попадания точки минимума в интервал.

3.2.3. Описание выходных данных

Выходные данные представляют собой найденное значение координаты точки экстремума, значение функции в точке экстремума и график функции с точкой минимума.

3.3. Алгоритм 3

Поиск экстремума функции одной переменной комбинированным методом Брента.

3.3.1. Описание входных данных

На вход подается названия функция в аналитическом виде, границы области оптимизации и дополнительные ограничения: точность оптимизации по аргументу, максимальное число итераций, флаг «вывод промежуточных результатов» и флаг «запись промежуточных результатов в датасет».

3.3.2. Описание алгоритма решения

В данном методе на каждой итерации отслеживаются значения в шести точках (не обязательно различных): a, c, x, w, v, u. Точки a, c задают текущий интервал поиска решения, x – точка, соответствующая наименьшему значению функции, w – точка, соответствующая второму снизу значению функции, v – предыдущее значение w. Аппроксимирующая парабола строится с помощью трех наилучших точек x, w, v.

Затем в качестве следующей точки оптимизационного процесса принимается минимум аппроксимирующей параболы (u) при выполнении следующих условий: u попадает внутрь интервала [a, c] и u отстоит от точки x не более, чем на половину от длины предпредыдущего шага. Если точка u отвергается, то следующая точка находится с помощью золотого сечения большего из интервалов [a, x] и [x, c].

3.3.3. Описание выходных данных

Выходные данные представляют собой найденное значение координаты точки экстремума, значение функции в точке экстремума и график функции с точкой минимума.

3.4. Алгоритм 4

Алгоритм неточной одномерной минимизации (Алгоритм Бройдена — Флетчера — Гольдфарба — Шанно).

3.4.1. Описание входных данных

На вход подается названия функция в аналитическом виде, начальная точка и дополнительные ограничения: параметр для первого условия Вольфе, параметр для второго условия Вольфе, максимально возможное значение аргумента, порог выхода по длине интервала поиска, максимально количество итераций, флаг для вывода промежуточных результатов, флаг для сохранения промежуточных результатов и флаг сохранения промежуточных результатов в датасете.

3.4.2. Описание алгоритма решения

Определяются условия, характеризующие «значимое уменьшение» значения функции на текущей итерации. Рассматривается новая функция и ее линейное приближение раскладывается в ряд Тейлора. Затем находится точка неточного решения задачи по условиям Голдштайна и Флетчера.

3.4.3. Описание выходных данных

Выходные данные представляют собой найденное значение координаты точки экстремума, значение функции в точке экстремума и график функции с точкой минимума.

4. Варианты использования системы

В разделе перечисляются названия предусматриваемых вариантов использования системы пользователем.

4.1. ВИ 1

При работе с алгоритмами 3.1-3.3 имеем функцию ввода со следующими входными параметрами (* - необязательный параметр для ввода):

1. Введите функцию y=f(x). Пример: x**2. Ввод:
    a. Пользователь вводит функцию в аналитическом виде с любым аргументом из латинского алфавита.
    b. Не допускаются такие аргументы, как sin, log и т.п.
    c. Предпочтительно, использовать функцию y=f(x).
    d. log - натуральный логарифм. Для ввода логарифма с другим основанием необходимо разделить на натуральный логарифм с желаемым основанием. Пример ввода функции логарифма с десятичным основанием: log(x)/log(10)
    e. Для ввода модуля используйте запись в формате Abs(функция с аргументом). Пример ввода: Abs(x)

    f. Для ввода экспоненты в степени используйте exp(функция с аргументом).
    g. Для ввода квадратного корня используйте sqrt(функция с аргументом). Для другой степени корня используйте возведение в степень.
2. Введите границы области оптимизации в формате отрезка. Пример: [-1,1]. Ввод:
    a. Пользователь вводит отрезок в формате [a,b].
    b. Обязательно написание со скобками. Скобки типа ‘()’ недопустимы. Между числами запятая.
    c. Допустим ввод только десятичных дробей (десятичный разделитель точка). Дроби типа a/b не допускаются. Например, вместо ½ при вводе необходимо написать 0.5.
    d. При наличии в отрезке значения pi, значение в отрезке писать через умножение. Пример: [0,2*pi],[0,1*pi].
3. Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:
    a. Пользователь может выбрать вариант 0 (нет), тогда значения для решения задачи будут взяты по умолчанию.
    b. При вводе 1 (да) пользователь продолжает ввод данных.
    c. Допустимы только значения 0 и 1.
4. * Введите точность оптимизации. Пример: 0.0001. Ввод:
    a. Вводится десятичное число с десятичным разделителем точка. Чем меньше число, тем выше точность найденного минимума.
    b. По умолчанию значение равно 0.00001
5. * Введите максимальное число итераций. Пример: 500. Ввод:

a. Вводится целое число без других лишних знаков.
b. По умолчанию значение равно 500.

6. * Хотите видеть промежуточные результаты на каждой итерации? False/True. Ввод:
    a. При вводе True пользователь будет видеть промежуточное решение и значение функции в найденной точке.
    b. По умолчанию значение равно False.
7. * Записывать промежуточные результаты на каждой итерации в pandas dataframe? False/True. Ввод:
    a. При вводе True все промежуточные результаты будут записаны в dataset, который можно будет использовать для дальнейшего анализа и других действий.
    b. По умолчанию значение равно False.

4.2. ВИ 2

При работе с алгоритмом 3.4 имеем функцию ввода со следующими входными параметрами (* - необязательный параметр для ввода):

1. Введите функцию y=f(x). Пример: x**2. Ввод:
    a. Пользователь вводит функцию в аналитическом виде с любым аргументом из латинского алфавита.
    b. Не допускаются такие аргументы, как sin, log и т.п.
    c. Предпочтительно, использовать функцию y=f(x).
    d. log - натуральный логарифм. Для ввода логарифма с другим основанием необходимо разделить на натуральный логарифм с желаемым основанием. Пример ввода функции логарифма с десятичным основанием: log(x)/log(10)

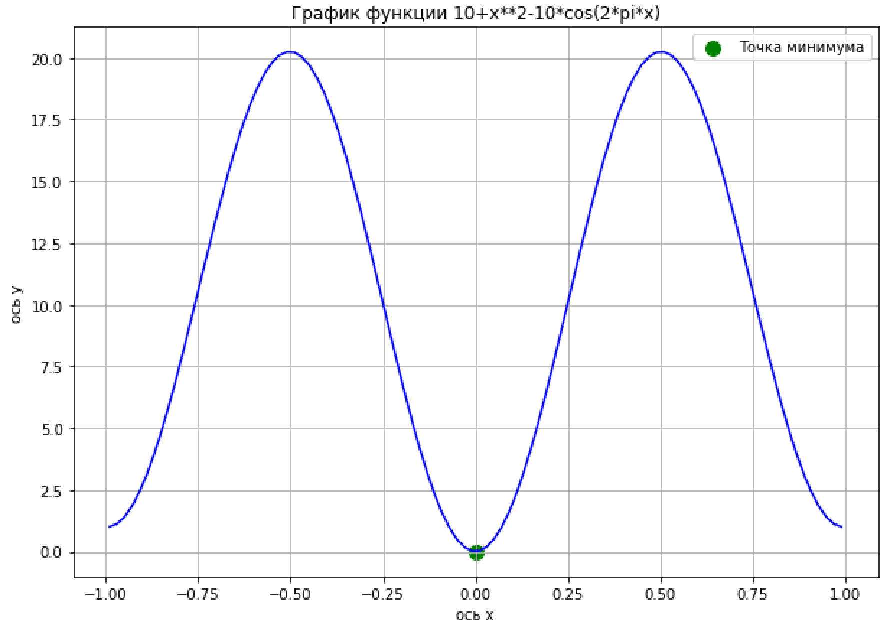
    e. Для ввода модуля используйте запись в формате Abs(функция с аргументом). Пример ввода: Abs(x)
    f. Для ввода экспоненты в степени используйте exp(функция с аргументом).
    g. Для ввода квадратного корня используйте sqrt(функция с аргументом). Для другой степени корня используйте возведение в степень.
2. Введите начальную точку. Пример: 0. Ввод:
    a. Пользователь вводит начальную точку в виде целого числа или десятичного.
    b. Допустим ввод только десятичных дробей (десятичный разделитель точка). Дроби типа a/b не допускаются. Например, вместо ½ при вводе необходимо написать 0.5.
3. Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:
    a. Пользователь может выбрать вариант 0 (нет), тогда значения для решения задачи будут взяты по умолчанию.
    b. При вводе 1 (да) пользователь продолжает ввод данных.
    c. Допустимы только значения 0 и 1.
4. *Введите параметр для первого условия Вольфе. Пример: 0.0001. Ввод:
    a. Вводится десятичное число с десятичным разделителем точка.
    b. По умолчанию значение равно 10**(-4)
5. * Введите параметр для второго условия Вольфе. Пример: 0.1. Ввод:

    a. Вводится десятичное число с десятичным разделителем точка.
    b. По умолчанию значение равно 0.1
6. * Введите максимально возможное значение аргумента. Пример: 100. Ввод:
    a. Вводится десятичное число с десятичным разделителем точка.
    b. По умолчанию значение равно 1000.
7. * Введите порог выхода по длине интервала поиска. Пример: 0.00001. Ввод:
    a. Вводится десятичное число с десятичным разделителем точка.
    b. По умолчанию значение равно 10**(-8)
8. * Введите максимальное число итераций. Пример: 500. Ввод:
    a. Вводится целое число без других лишних знаков.
    b. По умолчанию значение равно 500.
9. * Хотите видеть промежуточные результаты на каждой итерации? False/True. Ввод:
    a. При вводе True пользователь будет видеть промежуточное решение и значение функции в найденной точке.
    b. По умолчанию значение равно False.
10. * Записывать промежуточные результаты на каждой итерации в pandas dataframe? False/True. Ввод:
    a. При вводе True все промежуточные результаты будут записаны в dataset, который можно будет использовать для дальнейшего анализа и других действий.
    b. По умолчанию значение равно False.

4.3. Примеры работы системы с пользователем.

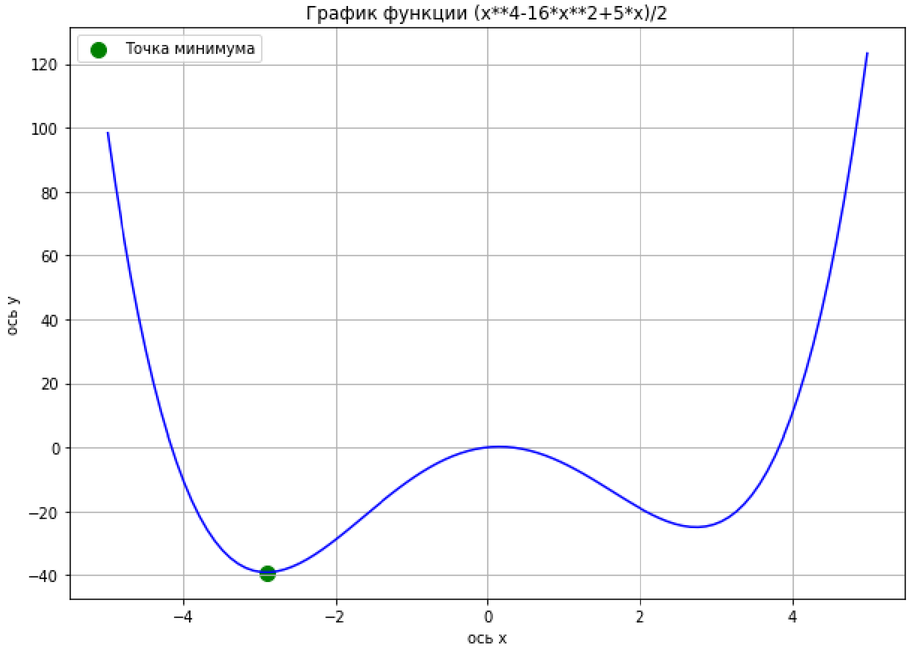
4.3.1. ВИ1

```
Введите функцию y=f(x). Пример: x**2. Ввод:10+x**2-10*cos(2*pi*x)
Введите границы области оптимизации в формате отрезка. Пример: [-1,1]. Ввод:[-1,1]
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:0
Полученный минимум функции методом золотого сечения: x = 0.00000 y = 0.00000
```



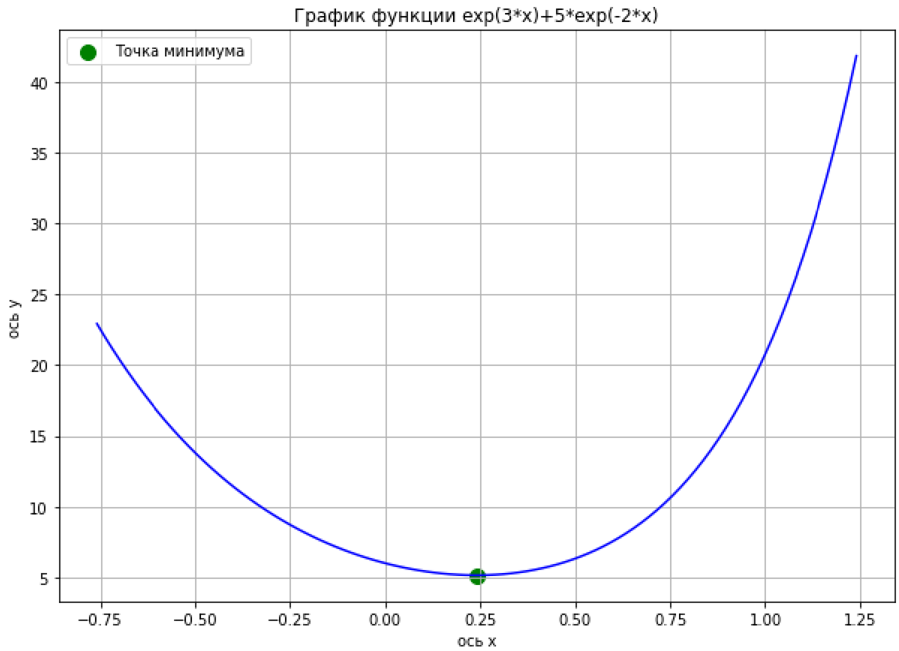

```
Введите функцию y=f(x). Пример: x**2. Ввод:10+x**2-10*cos(2*pi*x)
Введите границы области оптимизации в формате отрезка. Пример: [-1,1]. Ввод:[-1,1]
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:1
Введите точность оптимизации. Пример: 0.0001. Ввод:0.0001
Введите максимальное число итераций. Пример: 500. Ввод:500
Хотите видеть промежуточные результаты на каждой итерации? False/True. Ввод:True
Записывать промежуточные результаты на каждой итерации в pandas dataframe? False/True. Ввод:False
Промежуточный результат на итерации 1 : x = -0.38197 y = 17.51959
Промежуточный результат на итерации 2 : x = -0.14590 y = 3.93690
Промежуточный результат на итерации 3 : x = -0.00000 y = 0.00000
Промежуточный результат на итерации 4 : x = 0.09017 y = 1.57058
Промежуточный результат на итерации 5 : x = 0.03444 y = 0.23443
Промежуточный результат на итерации 6 : x = -0.00000 y = 0.00000
Промежуточный результат на итерации 7 : x = 0.02129 y = 0.08976
Промежуточный результат на итерации 8 : x = 0.00813 y = 0.01311
Промежуточный результат на итерации 9 : x = -0.00000 y = 0.00000
Промежуточный результат на итерации 10 : x = 0.00502 y = 0.00501
Промежуточный результат на итерации 11 : x = 0.00192 y = 0.00073
Промежуточный результат на итерации 12 : x = -0.00000 y = 0.00000
Промежуточный результат на итерации 13 : x = 0.00119 y = 0.00028
Промежуточный результат на итерации 14 : x = 0.00045 y = 0.00004
Промежуточный результат на итерации 15 : x = -0.00000 y = 0.00000
Промежуточный результат на итерации 16 : x = 0.00028 y = 0.00002
Промежуточный результат на итерации 17 : x = 0.00011 y = 0.00000
Промежуточный результат на итерации 18 : x = -0.00000 y = 0.00000
Промежуточный результат на итерации 19 : x = 0.00007 y = 0.00000
Промежуточный результат на итерации 20 : x = 0.00003 y = 0.00000
Промежуточный результат на итерации 21 : x = -0.00000 y = 0.00000
Полученный минимум функции методом золотого сечения: x = -0.00000 y = 0.00000
```

```
Введите функцию y=f(x). Пример: x**2. Ввод:(x**4-16*x**2+5*x)/2
Введите границы области оптимизации в формате отрезка. Пример: [-1,1]. Ввод:[-5,5]
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:1
Введите точность оптимизации. Пример: 0.0001. Ввод:0.0001
Введите максимальное число итераций. Пример: 500. Ввод:500
Хотите видеть промежуточные результаты на каждой итерации? False/True. Ввод:False
Записывать промежуточные результаты на каждой итерации в pandas dataframe? False/True. Ввод:True
Полученный минимум функции методом золотого сечения: x = -2.90355 y = -39.16617
```



4.3.2. ВИ2

```
Введите функцию y=f(x). Пример: x**2. Ввод:exp(3*x)+5*exp(-2*x)
Введите начальную точку. Пример: 0. Ввод:0
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:0
Полученный минимум функции методом неточной оптимизации: x = 0.24106 y = 5.14834
```



```
Введите функцию y=f(x). Пример: x**2. Ввод:-x/(x**2+2)
Введите начальную точку. Пример: 0. Ввод:0
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:1
Введите параметр для первого условия Вольфе. Пример: 0.0001. Ввод:0.00001
Введите параметр для второго условия Вольфе. Пример: 0.1. Ввод:0.3
Введите максимально возможное значение аргумента. Пример: 100. Ввод:100
Введите порог выхода по длине интервала поиска. Пример: 0.00001. Ввод:0.00001
Введите максимальное число итераций. Пример: 500. Ввод:500
Хотите видеть промежуточные результаты на каждой итерации? False/True. Ввод:True
Записывать промежуточные результаты на каждой итерации в pandas dataframe? False/True. Ввод:False
Промежуточный результат на итерации 1 : x = 50.01485 y = -0.01998
Промежуточный результат на итерации 2 : x = 25.03683 y = -0.03981
Промежуточный результат на итерации 3 : x = 12.57603 y = -0.07852
Промежуточный результат на итерации 4 : x = 6.39850 y = -0.14901
Промежуточный результат на итерации 5 : x = 3.40199 y = -0.25063
Промежуточный результат на итерации 6 : x = 2.04175 y = -0.33098
Промежуточный результат на итерации 7 : x = 1.35366 y = -0.35322
Промежуточный результат на итерации 8 : x = 1.40769 y = -0.35355
Промежуточный результат на итерации 9 : x = 1.41408 y = -0.35355
Промежуточный результат на итерации 10 : x = 1.41421 y = -0.35355
Полученный минимум функции методом неточной оптимизации: x = 1.41421 y = -0.35355
```

5. Архитектура решения

В разделе описываются создаваемые для решения задачи методы (функции), разделенные по 4-м принципиальным блокам.

5.1. Функции считывания информации

```
# def f_input():
  '''
  Функция для ввода данных для алгоритмов 1-3
  Выходные данные: список введенных значений
  '''
```

```
# def f_input_new():
  '''
  Функция для ввода данных для алгоритма 4
  Выходные данные: список введенных значений
  '''
```

5.2. Функции обработки информации

```
# def gold_ratio(y,g,eps=0.00001,step_max=500,options='False',dataset='False'):
  '''
  Функция для нахождения минимума методом золотого сечения
  Входные данные:
  y - функция в аналитическом виде, str
  g - границы области оптимизации, str
  eps - точность оптимизации, float
  step_max - максимально количество итераций, int
  options - вывод промежуточных результатов, str
  dataset - сохранение промежуточных результатов в dataframe, str
  Выходные данные: печать минимума и значения в точке экстремума. Возвращает точку минимума и dataset, если был указан соответсвующий параметр
  '''
```

```
# def method_parabola(y,g,eps=0.00001,step_max=500,options='False',dataset='False'):
  '''
  Функция для нахождения минимума методом парабол
  Входные данные:
  y - функция в аналитическом виде, str
  g - границы области оптимизации, str
  eps - точность оптимизации, float
  step_max - максимально количество итераций, int
  options - вывод промежуточных результатов, str
  dataset - сохранение промежуточных результатов в dataframe, str
  Выходные данные: печать минимума и значения в точке экстремума. Возвращает точку минимума и dataset, если был указан соответсвующий параметр
  '''
```

```
# def method_Brenta(y,g,eps=0.00001,step_max=500,options='False',dataset='False'):
  '''
  Функция для нахождения минимума методом Брента
  Входные данные:
  y - функция в аналитическом виде, str
  g - границы области оптимизации, str
  eps - точность оптимизации, float
  step_max - максимально количество итераций, int
  options - вывод промежуточных результатов, str
  dataset - сохранение промежуточных результатов в dataframe, str
  Выходные данные: печать минимума и значения в точке экстремума. Возвращает точку минимума и dataset, если был указан соответсвующий параметр
  '''
```

```
# def method_inaccur_optim(y,x_k,rho=10**(-4),sigma=0.1,alpha_max=1000,len_x=10**(-8),step_max=500,options='False',dataset='False'):
  '''
  Функция метода неточной одномерной минимизации
  Входные данные:
  y - функция в аналитическом виде, str
  x_k - начальная точка, float
  rho - параметр для первого условия Вольфе,float
  sigma - параметр для второго условия Вольфе,float
  alpha_max - максимально возможное значение аргумента, float
  len_x - порог выхода по длине интервала поиска,float
  step_max - максимально количество итераций, int
  options - вывод промежуточных результатов, str
  dataset - сохранение промежуточных результатов в dataframe, str
  Выходные данные: печать минимума и значения в точке экстремума. Возвращает точку минимума и dataset, если был указан соответсвующий параметр
  '''
```

5.3. Функции вывода информации

```
# def graph(y,g,x_solve):
  '''
  Функция для построения графика для алгоритмов 1-3
  Входные данные:
  y - функция в аналитическом виде, str
  g - границы области оптимизации, str
  x_solve - найденный минимум, float
  Выходные данные: печать графика с функцией и точкой минимума
  '''
```

```
# def graph_new(y,g,x_solve):
  '''
  Функция для построения графика для алгоритма 4
  Входные данные:
  y - функция в аналитическом виде, str
  x_solve - найденный минимум, float
  Выходные данные: печать графика с функцией и точкой минимума
  '''
```

5.4. Вспомогательные функции

```
# def all_f_for_gold_ratio():
  '''
  Функция для объединения ввода, нахождения минимума и построения графика
  '''
```

```
# def all_f_for_method_parabola():
  '''
  Функция для объединения ввода, нахождения минимума и построения графика
  '''
```

```
# def all_f_for_method_Brent():
  '''
  Функция для объединения ввода, нахождения минимума и построения графика
  '''
```

```
# def all_f_for_method_inaccur_optim():
  '''
  Функция для объединения ввода, нахождения минимума и построения графика
  '''
```

6. Тестирование

В разделе приводится тестирование работы программы. Оптимальный способ представления результатов тестирования – это следующие таблицы:

Таблица 1. Результаты тестирования и сравнение алгоритмов

Входные параметры	**Метод золотого сечения**	**Метод парабол**	**Метод Брента**	**Методом неточной оптимизаци и**	**Wolframalpha**
Функция 1	f(x) = exp(3*x)+5*exp(-2*x) на отрезке [-1,1]				
Полученное решение	x = 0.24079 y = 5.14834	x = 0.24079 y = 5.14834	x = 0.24079 y = 5.14834	x = 0.24079 y = 5.14834	x = 0.24079 y = 5.1483
Время выполнения	CPU times: user 99.5 ms, sys: 0 ns, total: 99.5 ms Wall time: 118 ms	CPU times: user 69.6 ms, sys: 0 ns, total: 69.6 ms Wall time: 78.2 ms	CPU times: user 36.5 ms, sys: 0 ns, total: 36.5 ms Wall time: 57.4 ms	CPU times: user 36 ms, sys: 0 ns, total: 36 ms Wall time: 70.5 ms	Время выполнения не выводится
Количество итераций	27	16	17	6	Количество итераций не выводится
Функция 2	f(x) = -x/(x**2+2) на отрезке [0,2]				

Полученное решение	x = 1.41422 y = -0.35355	x = 1.41422 y = -0.35355	x = 1.41422 y = -0.35355	x = 1.41422 y = -0.35355	x = 1.4142 y = -0.35355
Время выполнения	CPU times: user 69.7 ms, sys: 760 µs, total: 70.5 ms Wall time: 75.9 ms	CPU times: user 56.5 ms, sys: 0 ns, total: 56.5 ms Wall time: 79 ms	CPU times: user 84.8 ms, sys: 0 ns, total: 84.8 ms Wall time: 96.5 ms	CPU times: user 24.1 ms, sys: 1.91 ms, total: 26 ms Wall time: 33.4 ms	Время выполнения не выводится
Количество итераций	27	11	19	13	Количество итераций не выводится
Функция 3	f(x) = 5*x**2-48*x+576 на отрезке [0, 24]				
Полученное решение	x = 4.80000 y = 460.80000	x = 4.80000 y = 460.80000	x = 4.80000 y = 460.80000	x = 4.80000 y = 460.80000	x = 4.8 y = 460.8
Время выполнения	CPU times: user 99.5 ms, sys: 800 µs, total: 100 ms Wall time: 212 ms	CPU times: user 20.1 ms, sys: 56 µs, total: 20.2 ms Wall time: 44.6 ms	CPU times: user 102 ms, sys: 0 ns, total: 102 ms Wall time: 214 ms	CPU times: user 32 ms, sys: 908 µs, total: 32.9 ms Wall time: 73.2 ms	Время выполнения не выводится
Количество итераций	32	2	11	2	Количество итераций не выводится

Таблица 2. Результаты тестирования и сравнение алгоритмов

Входные параметры	Метод золотого сечения	Метод парабол	Метод Брента	Wolframalpha
Функция 4	f(x) = -5*x**5 + 4*x**4 - 12*x**3 + 11*x**2 - 2*x+1 на отрезке [-0.5, 0.5]			
Полученное решение	x = 0.10986 y = 0.89763	x = 0.10986 y = 0.89763	x = 0.10986 y = 0.89763	x = 0.10986 y = 0.897633
Время выполнения	CPU times: user 187 ms, sys: 5.94 ms, total: 193 ms Wall time: 267 ms	CPU times: user 64.6 ms, sys: 0 ns, total: 64.6 ms Wall time: 77.1 ms	CPU times: user 173 ms, sys: 1.5 ms, total: 174 ms Wall time: 277 ms	Время выполнения не выводится
Количество итераций	25	10	23	Количество итераций не выводится
Функция 5	f(x) = log(x-2)**2 + log(10-x)**2 - x**0.2 на отрезке [6, 9.9]			
Полученное решение	x = 8.50159 y = 2.13384	x = 8.50149 y = 2.13384	x = 8.50159 y = 2.13384	x = 8.50159 y = 2.13384
Время выполнения	CPU times: user 278 ms, sys: 599 µs, total: 279 ms Wall time: 366 ms	CPU times: user 922 ms, sys: 6.72 ms, total: 929 ms Wall time: 1.46 s	CPU times: user 201 ms, sys: 1.46 ms, total: 203 ms Wall time: 330 ms	Время выполнения не выводится

Количество итераций	28	111	23	Количество итераций не выводится
Функция 6	f(x) = -3*x*sin(0.75*x) + exp(-2*x) на отрезке [0, 2*pi]			
Полученное решение	x = 2.70648 y = -7.27436	x = 2.70648 y = -7.27436	x = 2.70648 y = -7.27436	x = 2.70648 y = -7.27436
Время выполнения	CPU times: user 974 ms, sys: 6.57 ms, total: 981 ms Wall time: 1.72 s	CPU times: user 185 ms, sys: 3.56 ms, total: 189 ms Wall time: 301 ms	CPU times: user 401 ms, sys: 2.69 ms, total: 403 ms Wall time: 666 ms	Время выполнения не выводится
Количество итераций	29	9	27	Количество итераций не выводится
Функция 7	0.2*x*log(x) + (x-2.3)**2 на отрезке [0.5, 2.5]			
Полученное решение	x = 2.12464 y = 0.35098	x = 2.12464 y = 0.35098	x = 2.12464 y = 0.35098	x = 2.12464 y = 0.350978
Время выполнения	CPU times: user 216 ms, sys: 2.6 ms, total: 219 ms Wall time: 393 ms	CPU times: user 109 ms, sys: 0 ns, total: 109 ms Wall time: 212 ms	CPU times: user 224 ms, sys: 858 µs, total: 225 ms Wall time: 395 ms	Время выполнения не выводится
Количество итераций	27	4	39	Количество итераций не выводится

Также проведем сравнительный анализ точного алгоритма, который был создан в прошлой работе и модифицирован для работы с одномерной функцией, и приближенных алгоритмов, выбирая наиболее оптимальный для каждой функции по результатам тестирования выше. Например, можно заметить что со степенными функциями лучше всего работает метод парабол.

Таблица 3. Сравнение производительность приближенных алгоритмов и точных алгоритмов

Функция 1	f(x) = 5*x**2-48*x+576 на отрезке [0, 24]	
Выбранный алгоритм	Метод парабол	Точный алгоритм
Полученное решение	x = 4.80000 y = 460.80000	В точке x = 4.80000 функция имеет локальный минимум равный 460.80000
Время выполнения	Wall time: 44.6 ms	Wall time: 20.4 ms
Функция 2	f(x) = 0.2*x*log(x) + (x-2.3)**2 на отрезке [0.5, 2.5]	
Выбранный алгоритм	Метод парабол	Точный алгоритм
Полученное решение	x = 2.12464 y = 0.35098	В точке x = 2.12464 функция имеет локальный минимум равный 0.35098
Время выполнения	Wall time: 212 ms	Wall time: 195 ms
Функция 3	f(x) =3*x-log(x) на отрезке [0.1,1]	
Выбранный алгоритм	Метод золотого сечения	Точный алгоритм

Полученное решение	x = 0.33333 y = 2.09861	В точке x = 0.33333 функция имеет локальный минимум равный 2.09861
Время выполнения	Wall time: 4 ms	Wall time: 4 ms
Функция 4	f(x) = exp(3*x)+5*exp(-2*x) на отрезке [-1,1]	
Выбранный алгоритм	Метод неточной оптимизации	Точный алгоритм
Полученное решение	x = 0.24079 y = 5.14834	В точке x = 0.24079 функция имеет локальный минимум равный 5.14834
Время выполнения	Wall time: 70.5 ms	Wall time: 20.4 ms
Функция 5	f(x) = -5*x**5 + 4*x**4 - 12*x**3 + 11*x**2 - 2*x+1 на отрезке [-0.5, 0.5]	
Выбранный алгоритм	Метод парабол	Точный алгоритм
Полученное решение	x = 0.10986 y = 0.89763	В точке x = 0.10986 функция имеет локальный минимум равный 0.89763
Время выполнения	Wall time: 267 ms	Wall time: 1.03 s

7. Заключение

Итак, подводя итоги, можно констатировать следующее: согласно требованиям заказчика в среде программирования Python была реализована функция, которая находит минимальные денежные затраты на переработку отходов производства предприятия с помощью поиска экстремума функции одной переменной методом золотого сечения, методом парабол, комбинированным методом Брента и с помощью алгоритма неточной одномерной минимизации (Алгоритм Бройдена — Флетчера — Гольдфарба — Шанно). А также было обеспечено решение визуализацией с помощью двумерных графиков функции, с отмеченными точками локальных экстремумов.

Произведем сравнение выбранных алгоритмов по разным критериям. Оптимальный вид сравнения приведен в следующей таблице:

Таблица 4. Сравнение алгоритмов.

Критерий	**Метод золотого сечения**	**Метод парабол**	**Метод Брента**	**Методом неточной оптимизации**	**Wolframalpha**
Возможные параметры для ввода	функция в аналитическом виде, границы области оптимизации, максимально количество итераций, вывод промежуточных результатов, сохранение промежуточных результатов в dataframe	функция в аналитическом виде, границы области оптимизации, максимально количество итераций, вывод промежуточных результатов, сохранение промежуточных результатов в dataframe	функция в аналитическом виде, границы области оптимизации, максимально количество итераций, вывод промежуточных результатов, сохранение промежуточных результатов в dataframe	функция в аналитическом виде, границы области оптимизации, точность оптимизации, начальная точка, параметр для первого условия Вольфе, параметр для второго условия Вольфе, максимально возможное значение аргумента, порог выхода по длине интервала поиска, максимально	функция в аналитическом виде, границы области оптимизации

				количество итераций, вывод промежуточных результатов, сохранение промежуточных результатов в dataframe	
Точность выводимых значений	До пяти знаков после запятой	До пяти знаков после запятой	До пяти знаков после запятой	До пяти знаков после запятой	В некоторых случаях до шести знаков после запятой
Полнота вывода ответа	Выводятся в печатном и графическом виде все локальные минимумы и максимумы, а также пишутся седловые точки. Выводятся координаты точек экстремум. Дроби и корни переводятся в десятичные числа.	Выводятся в печатном и графическом виде все локальные минимумы и максимумы, а также пишутся седловые точки. Выводятся координаты точек экстремум. Дроби и корни переводятся в	Выводятся в печатном и графическом виде все локальные минимумы и максимумы, а также пишутся седловые точки. Выводятся координаты точек экстремум. Дроби и корни переводятся в	Выводятся в печатном и графическом виде все локальные минимумы и максимумы, а также пишутся седловые точки. Выводятся координаты точек экстремума. Дроби и корни переводятся в десятичные числа.	Выводятся в графическом виде все локальные минимумы и максимумы (без седловых точек), но в печатном формате не всегда пишутся все точки экстремума. Выводятся

		десятичные числа.	десятичные числа.		координаты точек экстремума.
Формат вывода ответа	Все значения экстремумов выводятся в форме десятичного числа	Все значения экстремумов выводятся в форме десятичного числа	Все значения экстремумов выводятся в форме десятичного числа	Все значения экстремумов выводятся в форме десятичного числа	Некоторые значения выводятся в формате дробей (в том числе с корнями), остальные в форме десятичного числа
Время выполнения	Время выполнения для разных функций: 118 ms 75.9 ms 212 ms 267 ms 366 ms 1.27 s 393 ms	Время выполнения для разных функций: 78.2 ms 79 ms 44.6 ms 77.1 ms 1.46 s 301 ms 212 ms	Время выполнения для разных функций: 57.4 ms 96.5 ms 214 ms 277 ms 330 ms 666 ms 395 ms	Время выполнения для разных функций: 70.5 ms 33.4 ms 73.2 ms - - - -	Время выполнения не выводится
Количество итераций	27 27 32 25 28 29	16 11 2 10 111 9	17 19 11 23 23 27	6 13 2 - - -	Количество итераций не выводится

	27	4	39	-	
Количество графиков	Один	Один	Один	Один	Один

Мы предлагаем в качестве оптимального решения поставленной задачи алгоритм 2 (метод парабол), так как он выводит ответ за наименьшее количество времени и наименьшее количество итераций, по сравнению с остальными алгоритмами.

Таким образом можно сделать вывод, что из 24 тонн мусора на первый завод нам надо поставить 4.8 тонн, соответственно на второй 19.2 тонны. При таком соотношении заказчик сможет заплатить минимальное количество денежных единиц, которое равно 460.8 д.е., что удовлетворяет его запросу.