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 № 3
«Методы многомерной 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

Содержание

Постановка задачи

“Оптимизировать затраты заказчика на переработку отходов производства предприятия и на покупку земли под застройку”

Необходимо в среде программирования Python реализовать функции, которые будут находить минимальные денежные затраты на переработку отходов производства предприятия и на покупку земли под застройку.

Условия заказчика:

Заказчик занимается строительством коттеджей. После проведения всех работ осталось 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 $4x^2$ денежных единиц за переработку x тонн мусора. Второму заводу надо заплатить за работу по утилизации и переработке x тонн мусора x^2 денежных единиц.

Также у заказчика появилась возможность приобрести землю под застройку новых коттеджей в двух поселках. С учетом всех денежных и трудовых затрат заказчик может приобрести не более 10-ти соток земли в двух поселках. Также немаловажным фактором является цена земли. В первом поселке она составляет $9y^2$ денежных единиц за сотку, а во втором - $5y^2$.

Заказчику необходимо распределить весь объем производственных отходов между двумя заводами и определиться с количеством покупаемой земли в каждом из поселков так, чтобы денежные затраты были минимальными. Стоит отметить, что заводам можно передавать не целое количество тонн производственных отходов и что площадь покупаемой земли в одном поселке тоже может быть не целым числом.

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 $f_1(x) = 4x^2$ денежных единиц, а на втором заводе - $f_2(x) = (24 - x)^2 = 576 - 48x + x^2$. Получается, что заказчик должен будет заплатить $f(x) = f_1(x) + f_2(x) = 5x^2 - 48x + 576$ денежных единиц.

Всего заказчик может приобрести 100 соток земли. Пусть в первом поселке будет куплено y соток (значение y может быть не целым), тогда во втором поселке можно будет приобрести $(100 - y)$ соток земли. Для покупки земли в первом поселке необходимо $f_1(y) = 9y^2$ денежных единиц за сотку, а во втором - $f_2(y) = 5(100 - y)^2 = 10000 - 200y + y^2$ за сотку. Получается, что заказчик должен будет заплатить $f(y) = f_1(y) + f_2(y) = 10y^2 - 200y + 10000$ денежных единиц. Необходимо найти минимум функции $f(x, y)$, которая имеет вид:

$$f(x, y) = 5x^2 - 48x + 10y^2 - 200y + 10576$$

Другими словами, необходимо найти минимальные затраты заказчика. Также необходимо найти количество тонн мусора, которые будут переданы на первый завод, т.е. найти значение x и найти количество соток земли, которые будут куплены в первом поселке, т.е. найти значение y.

Математическая модель

Для решения выше поставленной задачи

$$f(x, y) = 5x^2 - 48x + 10y^2 - 200y + 10576$$

, где необходимо найти минимум $f(x, y)$, будем использовать методы многомерной оптимизации. В данной работе реализовано 4 метода оптимизации: метод градиентного спуска с постоянным шагом, метод градиентного спуска с дроблением шага, метод наискорейшего градиентного спуска, алгоритм Ньютона-сопряженного градиента. Приведем математические модели 4 методов в общем виде.

2.1 Математическая модель метода градиентного спуска с постоянным шагом.

Градиентом $\nabla f(x)$ непрерывно дифференцируемой функции $f(x)$ в точке x называется вектор-столбец частных производных, вычисленных в этой точке:

$$\nabla f(x) = \begin{pmatrix} \frac{df(x)}{dx^1} \\ \frac{df(x)}{dx^2} \\ \cdots \\ \frac{df(x)}{dx^n} \end{pmatrix} \quad (1)$$

Градиент направлен перпендикулярно касательной плоскости, проведенной в точке , в сторону возрастания функции.

Пусть дана функция $f(x)$, ограниченная снизу на множестве R^n и имеющая непрерывные частные производные во всех его точках.

$$f(x) \to min, x \in R^n \quad (2)$$

Алгоритм представляет собой последовательность шагов постоянной величины от начальной точки в направлении антиградиента, т.е. в сторону убывания функции.

Начальные точки:

$$x_0 = (x_1^0, x_2^0, \ldots, x_n^0), x_1 = (x_1^1, x_2^1, \ldots, x_n^1) \quad (3)$$

Длина шага h, число $0 < \xi < 1$ для остановки алгоритма.
Условие остановки алгоритма:

$$||x^1 - x^0|| \leq \xi \tag{4}$$

При выполнении условия остановки (4):

$$minf(x) = f(x^1) \tag{5}$$

Если условие остановки не выполнено, то вычисляем следующую точку x^2, т.е. делаем шаг от точки x^1:

$$x^2 = x^1 - h\nabla f(x^1) \tag{6}$$

Условие остановки :

$$||x^2 - x^1|| \leq \xi \tag{7}$$

При выполнении условия остановки (7):

$$minf(x) = f(x^2) \tag{8}$$

Если условие остановки не выполнено, то делаем шаг от точки x^2 и так далее, пока точка локального минимума не будет найдена.

2.2 Математическая модель метода градиентного спуска с дроблением шага.

Задаются параметр дробления $0 < \delta < 1$, начальная величина шага λ_0 и значение параметра оценки $0 < \epsilon < 1$. Для каждого k полагают $\lambda_k = \lambda_0$.

Величина шага λ_k на каждой итерации выбирается из условия выполнения неравенства:

$$f(x^{k+1}) = f\left(x^k - \lambda^k f'(x^k)\right) \leq f(x^k) - \epsilon\lambda^k \left|\left|f'(x^k)\right|\right|^2 \tag{9}$$

Если условие (9) выполняется, то переходим к следующему шагу, иначе

$$\lambda_k = \lambda_k\delta \tag{10}$$

Условие остановки алгоритма:

$$||x^k - x^{k-1}|| \leq \xi \tag{11}$$

При выполнении условия выводится x^k.

2.3 Математическая модель метода наискорейшего градиентного спуска.

Этот вариант градиентного метода основывается на выборе шага из следующего соображения. Из точки x^k будем двигаться в направлении антиградиента до тех пор пока не достигнем минимума функции f на этом направлении, т. е. на луче

$$L = \{x = x^k - \lambda f'(x^k)\} \tag{12}$$

Найдем с помощью метода золотого сечения λ, при котором будет достигаться минимум функции:

$$f(\lambda) = f(x^k - \lambda f'(x^k)) \tag{13}$$

Найдем x с помощью формулы (11) при найденой λ.

Условие остановки алгоритма:

$$||x^k - x^{k-1}|| \leq \xi \tag{14}$$

При выполнении условия остановки (14):

$$minf(x) = f(x^k) \tag{15}$$

Если условие остановки не выполнено, то делаем шаг от точки x^k и так далее, пока точка локального минимума не будет найдена.

2.4 Математическая модель алгоритма Ньютона-сопряженного градиента.

Вычисление градиента функции:

$$\nabla f(x) = \begin{pmatrix} \frac{df(x)}{dx^1} \\ \frac{df(x)}{dx^2} \\ \cdots \\ \frac{df(x)}{dx^n} \end{pmatrix} \tag{16}$$

Весовой коэффициент не рассчитывается на первом шаге, на последующих с помощью формулы:

$$\beta_k = \frac{\left|\left|\nabla f(x_1, x_2, ..., x_n)_k\right|\right|^2}{\left|\left|\nabla f(x_1, x_2, ..., x_n)_{k-1}\right|\right|^2} \tag{17}$$

Для первого шага единичный вектор сопряженных направлений определяется следующим образом:

$$P_0 = \nabla f(X_0) \tag{18}$$

Для последующих шагов единичный вектор сопряженных направлений определяется:

$$P_k = \nabla f(X_k) + \beta_k P_{k-1} \tag{19}$$

Шаг расчета из условия поиска экстремума для следующей функции $F\big(x_k - \lambda P_k(x_k)\big)$

$$\lambda_k F\big(x_k - \lambda P_k(x_k)\big) \to extr \tag{20}$$

Определение нового значения аргументов функции после выполнения k-го шага расчета:

$$x_{k+1} = x_k - \lambda_k P_k \tag{21}$$

Условие остановки алгоритма:

$$||x^k - x^{k-1}|| \leq \xi \tag{22}$$

Вычислительный процесс заканчивается, когда будет достигнута точка, в которой оценка градиента будет равна нулю.

Алгоритмы

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

3.1 Алгоритм метода градиентного спуска с постоянным шагом

3.1.1 Описание входных данных

На вход подаются минимизируемая функция в аналитическом виде, функция градиента в аналитическом виде, константа шага (λ) и дополнительные параметры: максимальное число итераций, точность оптимизации по аргументу для критерия Останова, флаг «вывод промежуточных результатов» и флаг «запись промежуточных результатов в датасет».

3.1.2 Описание алгоритма решения

Алгоритм представляет собой последовательность шагов постоянной величины от начальных точек (3) в направлении антиградиента, т.е. в сторону убывания функции.

При выполнении условия остановки (4) необходимо перейти к (5). Если условие остановки не выполняется, то вычисляется следующую точку x^2, т.е. делается шаг от точки x^1 с помощью формулы (6).

Далее повторяем все действия с проверки условия остановки уже для новой точки до тех пор, пока не будет найден локальный минимум функции.

3.1.3 Описание выходных данных

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

3.2 Алгоритм метода градиентного спуска с дроблением шага

3.2.1 Описание входных данных

На вход подаются минимизируемая функция в аналитическом виде, функция градиента в аналитическом виде, начальный шаг (λ_0), значение параметра оценки (ϵ), значение параметра дробления (δ) и дополнительные параметры:

максимальное число итераций, точность оптимизации по аргументу для критерия Останова, флаг «вывод промежуточных результатов» и флаг «запись промежуточных результатов в датасет».

3.2.2 Описание алгоритма решения

После задания необходимых параметров для вычислений, переходим к выбору величины шага по неравенству (9).

При выполнении условия (9) переходим к следующему шагу. При невыполнении условия (9) переходим к равенству (10).

На новом шаге алгоритма проверяется условие остановки (11). Если оно выполняется, то выводим значение x^k.

3.2.3 Описание выходных данных

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

3.3 Алгоритм метода наискорейшего градиентного спуска

3.3.1 Описание входных данных

На вход подаются минимизируемая функция в аналитическом виде, функция градиента в аналитическом виде и дополнительные параметры: максимальное число итераций, точность оптимизации по аргументу для критерия Останова, флаг «вывод промежуточных результатов» и флаг «запись промежуточных результатов в датасет».

3.3.2 Описание алгоритма решения

Сначала из точки x^k двигаемся в направлении антиградиента, пока не достигнем максимума функции f на этом направлении, т. е. на луче (12).

Затем с помощью метода золотого сечения находим значение λ, при котором будет достигаться минимум функции (13). После этого с помощью формулы (11) находим x по найденному ранее значению λ.

Далее проверяем условие остановки алгоритма по формуле (14). При его выполнении переходим к формуле (15). При невыполнении условия остановки делаем шаг от точки x^k и так далее, пока точка локального минимума не будет найдена.

3.3.3 Описание выходных данных

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

3.4 Алгоритм Ньютона-сопряженного градиента

3.4.1 Описание входных данных

На вход подается функция в аналитическом виде.

3.4.2 Описание алгоритма решения

Сначала вычисляется градиент заданной функции по формуле (16). Для первого шага единичный вектор сопряженных направлений определяется по формуле (18).

Перед выполнением последующих шагом необходимо вычислить весовой коэффициент по формуле (17). После выполнения этого расчета переходим к формуле (19) для определения единичного вектора сопряженных направлений.

Следующий шаг включает в себя поиск экстремума функции по формуле (20). Далее идет определение новых значений аргументов функции после выполнения k-го шага расчета по формуле (21).

Затем проверяется условие остановки алгоритма (22) и вычислительный процесс заканчивается, когда будет достигнута точка, в которой оценка градиента будет равна нулю.

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.

Варианты использования

4.1 Использование системы для метода градиентного спуска с постоянным шагом

При работе с методом градиентного спуска с постоянным шагом имеем следующий вариант использования системы (* - необязательный параметр для ввода):

1. Пользователь должен ввести функцию $f(x, y \dots)$.
   Пример: $x^2 + y^2$.

   - Пользователь должен вводить функцию в аналитическом виде с любыми аргументами из латинского алфавита.
   - Не допускаются такие аргументы, как sin, log и т.п.
   - log - натуральный логарифм. Для ввода логарифма с другим основанием необходимо разделить на натуральный логарифм с желаемым основанием. Пример ввода функции логарифма с десятичным основанием: $\frac{log(x)}{log(10)}$
   - Для ввода модуля используйте запись в формате Abs(функция с аргументом). Пример ввода: Abs(x)
   - Для ввода экспоненты в степени используйте exp (функция с аргументом).
   - Для ввода квадратного корня используйте sqrt (функция с аргументом). Для другой степени корня используйте возведение в степень.

2. Пользователь должен ввести константа шаг Lambda.
   Пример: 0.01.

   - Пользователь должен ввести число без лишних знаков.
   - Допускаются и десятичные, и целочисленные значения.
   - Недопустимо ввод дробей вида $\frac{1}{2}$. Для этого нужно перевести дробь в десятичный формат.
   - Для метода градиентного спуска с постоянным шагом наиболее оптимальным шагом является 0.01. При выборе более высокого значения, алгоритм может расходится.

3. Пользователь должен ввести параметр, отвечающий за дополнительные ограничения.
   Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет.

- Пользователь может выбрать вариант 0 (нет), тогда значения для решения задачи будут взяты по умолчанию.
- При вводе 1 (да) пользователь продолжает ввод данных.
- Допустимы только значения 0 и 1.

4. * Пользователь может ввести точность оптимизации.
   Пример: 0.0001.

   - Вводится десятичное число с десятичным разделителем точка. Чем меньше число, тем выше точность найденного минимума.
   - По умолчанию значение равно 0.00001

5. * Пользователь может ввести максимальное число итераций.
   Пример: 500.

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

6. * Пользователь может ввести параметр, отвечающий за промежуточные результаты на каждой итерации.

   - При вводе True пользователь будет видеть промежуточное решение и значение функции в найденной точке.
   - По умолчанию значение равно False.
   - Допустимы только значения True и False.

7. * Пользователь может ввести параметр, отвечающий за запись промежуточных результатов на каждой итерации в pandas dataframe.

   - При вводе True все промежуточные результаты будут записаны в dataset, который можно будет использовать для дальнейшего анализа и других действий.
   - По умолчанию значение равно False.
   - Допустимы только значения True и False.

8. Выходные данные представляют собой список введенных значений, которые используются в последующих алгоритмах.

4.2 Использование системы для метода градиентного спуска с дроблением шага

При работе с методом градиентного спуска с дроблением шага имеем следующий вариант использования системы (* - необязательный параметр для ввода):

1. Пользователь должен ввести функцию $f(x, y\dots)$.
   Пример: $x^2 + y^2$.

   - Пользователь должен вводить функцию в аналитическом виде с любыми аргументами из латинского алфавита.
   - Не допускаются такие аргументы, как sin, log и т.п.
   - log - натуральный логарифм. Для ввода логарифма с другим основанием необходимо разделить на натуральный логарифм с желаемым основанием. Пример ввода функции логарифма с десятичным основанием: $\frac{log(x)}{log(10)}$
   - Для ввода модуля используйте запись в формате Abs(функция с аргументом). Пример ввода: Abs(x)
   - Для ввода экспоненты в степени используйте exp (функция с аргументом).
   - Для ввода квадратного корня используйте sqrt (функция с аргументом). Для другой степени корня используйте возведение в степень.

2. Пользователь должен ввести начальный шаг Lambda0.
   Пример: 0.01.

   - Пользователь должен ввести число без лишних знаков.
   - Допускаются и десятичные, и целочисленные значения.
   - Недопустимо ввод дробей вида $\frac{1}{2}$. Для этого нужно перевести дробь в десятичный формат.

3. Пользователь должен ввести значение параметра оценки в пределах от 0 до 1.
   Пример: 0.01.

   - Пользователь должен ввести число без лишних знаков.
   - Недопустимо ввод дробей вида $\frac{1}{2}$. Для этого нужно перевести дробь в десятичный формат.

4. Пользователь должен ввести значение параметра дробления в пределах от 0 до 1.
   Пример: 0.1.

   - Пользователь должен ввести число без лишних знаков.
   - Недопустимо ввод дробей вида $\frac{1}{2}$. Для этого нужно перевести дробь в десятичный формат.

5. Пользователь должен ввести параметр, отвечающий за дополнительные ограничения.
   Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет.

- Пользователь может выбрать вариант 0 (нет), тогда значения для решения задачи будут взяты по умолчанию.
- При вводе 1 (да) пользователь продолжает ввод данных.
- Допустимы только значения 0 и 1.

6. * Пользователь может ввести точность оптимизации.
   Пример: 0.0001.

   - Вводится десятичное число с десятичным разделителем точка. Чем меньше число, тем выше точность найденного минимума.
   - По умолчанию значение равно 0.00001

7. * Пользователь может ввести максимальное число итераций.
   Пример: 500.

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

8. * Пользователь может ввести параметр, отвечающий за промежуточные результаты на каждой итерации.

   - При вводе True пользователь будет видеть промежуточное решение и значение функции в найденной точке.
   - По умолчанию значение равно False.
   - Допустимы только значения True и False.

9. * Пользователь может ввести параметр, отвечающий за запись промежуточных результатов на каждой итерации в pandas dataframe.

   - При вводе True все промежуточные результаты будут записаны в dataset, который можно будет использовать для дальнейшего анализа и других действий.
   - По умолчанию значение равно False.
   - Допустимы только значения True и False.

10. Выходные данные представляют собой список введенных значений, которые используются в последующих алгоритмах.

4.3 Использование системы для наискорейшего спуска и метода Ньютона

При работе с методом градиентного спуска с дроблением шага имеем следующий вариант использования системы (* - необязательный параметр для ввода):

1. Пользователь должен ввести функцию $f(x, y\dots)$.
   Пример: $x^2 + y^2$.

   - Пользователь должен вводить функцию в аналитическом виде с любыми аргументами из латинского алфавита.
   - Не допускаются такие аргументы, как sin, log и т.п.
   - log - натуральный логарифм. Для ввода логарифма с другим основанием необходимо разделить на натуральный логарифм с желаемым основанием. Пример ввода функции логарифма с десятичным основанием: $\frac{log(x)}{log(10)}$
   - Для ввода модуля используйте запись в формате Abs(функция с аргументом). Пример ввода: Abs(x)
   - Для ввода экспоненты в степени используйте exp (функция с аргументом).
   - Для ввода квадратного корня используйте sqrt (функция с аргументом). Для другой степени корня используйте возведение в степень.

2. Пользователь должен ввести параметр, отвечающий за дополнительные ограничения.
   Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет.

   - Пользователь может выбрать вариант 0 (нет), тогда значения для решения задачи будут взяты по умолчанию.
   - При вводе 1 (да) пользователь продолжает ввод данных.
   - Допустимы только значения 0 и 1.

3. * Пользователь может ввести точность оптимизации.
   Пример: 0.0001.

   - Вводится десятичное число с десятичным разделителем точка. Чем меньше число, тем выше точность найденного минимума.
   - По умолчанию значение равно 0.00001

4. * Пользователь может ввести максимальное число итераций.
   Пример: 500

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

5. * Пользователь может ввести параметр, отвечающий за промежуточные результаты на каждой итерации.

   - При вводе True пользователь будет видеть промежуточное решение и значение функции в найденной точке.

- По умолчанию значение равно False.
- Допустимы только значения True и False.

6. * Пользователь может ввести параметр, отвечающий за запись промежуточных результатов на каждой итерации в pandas dataframe.
   - При вводе True все промежуточные результаты будут записаны в dataset, который можно будет использовать для дальнейшего анализа и других действий.
   - По умолчанию значение равно False.
   - Допустимы только значения True и False.
7. Выходные данные представляют собой список введенных значений, которые используются в последующих алгоритмах.

4.4 Пример использования системы

```
Введите функцию в аналитическом виде f(x,y,..). Пример: x**2+y**2. Ввод:x**2
Введите константа шаг Lambda. Пример: 0.1. Ввод:0.01
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:0
Полученный минимум функции методом градиентного спуска с постоянным шагом:  {x: '0.00048'} f(x,y...) = 0.00000


Введите функцию в аналитическом виде f(x,y,..). Пример: x**2+y**2. Ввод:x**2
Введите константа шаг Lambda. Пример: 0.1. Ввод:0.01
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:1
Введите точность оптимизации по аргументу для критерия Останова. Пример: 0.0001. 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 = [0.98] y = 0.96040
Промежуточный результат на итерации 2 : x = [0.9604] y = 0.92237
Промежуточный результат на итерации 3 : x = [0.941192] y = 0.88584
Промежуточный результат на итерации 4 : x = [0.92236816] y = 0.85076
Промежуточный результат на итерации 5 : x = [0.9039208] y = 0.81707
Промежуточный результат на итерации 6 : x = [0.88584238] y = 0.78472
Промежуточный результат на итерации 7 : x = [0.86812553] y = 0.75364
Промежуточный результат на итерации 8 : x = [0.85076302] y = 0.72380
Промежуточный результат на итерации 9 : x = [0.83374776] y = 0.69514
Промежуточный результат на итерации 10 : x = [0.81707281] y = 0.66761
Промежуточный результат на итерации 11 : x = [0.80073135] y = 0.64117
Промежуточный результат на итерации 12 : x = [0.78471672] y = 0.61578


Введите функцию в аналитическом виде f(x,y,..). Пример: x**2+y**2. Ввод:x**2
Введите константа шаг Lambda. Пример: 0.1. Ввод:0.01
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:1
Введите точность оптимизации по аргументу для критерия Останова. Пример: 0.0001. Ввод:0.00001
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 градиентного спуска с постоянным шагом:  {x: '0.00048'} f(x,y...) = 0.00000
```

Рис. 1: Варианты использования системы для метода градиентного спуска с постоянным шагом

```
Введите функцию в аналитическом виде f(x,y,..). Пример: x**2+y**2. Ввод:x**2
Введите начальный шаг Lambda0. Пример: 0.1. Ввод:0.01
Введите значение параметра оценки в пределах от 0 до 1. Пример: 0.1. Ввод:0.1
Введите значение параметра дробления в пределах от 0 до 1. Пример: 0.1. Ввод:0.1
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:0
Полученный минимум функции методом градиентного спуска с дроблением шага:  {x: '0.00048'} f(x,y...) = 0.00000
```

Рис. 2: Варианты использования системы для метода градиентного спуска с дроблением шага

```
Введите функцию в аналитическом виде f(x,y,..). Пример: x**2+y**2. Ввод:x**2
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:0
Полученный минимум функции методом наискорейшего спуска:  {x: '0.00000'} f(x,y...) = 0.00000
```

Рис. 3: Варианты использования системы для метода наискорейшего спуска

```
Введите функцию в аналитическом виде f(x,y,..). Пример: x**2+y**2. Ввод:x**2
Хотите ввести дополнительные ограничения? Если нет, будут взяты параметры по умолчанию. 1- Да/ 0 - Нет. Ввод:0
Полученный минимум функции методом Ньютона:  {x: '0.00000'} f(x,y...) = 0.00000
```

Рис. 4: Варианты использования системы для метода Ньютона-сопряженного градиента

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-1():**
  '''
  Функция-метод для ввода данных для метода градиентного спуска с постоянным шагом
  Выходные данные: список введенных значений, list
  '''

- **def f-input-2():**
  '''
  Функция-метод для ввода данных для метода градиентного спуска с дроблением шага
  Выходные данные: список введенных значений, list
  '''

- **def f-input-3():**
  '''
  Функция-метод для ввода данных для метода наискорейшего спуска и метода Ньютона
  Выходные данные: список введенных значений, list
  '''

5.2 Функции обработки информации

- **def grad(y):**
  '''
  Функция-метод для нахождения градиента функции в аналитическом виде
  Входные данные:
  y - функция в аналитическом виде, str
  Выходные данные: градиент функции в аналитическом виде, str
  '''

- **def grad-with-constant-step (y, Lambda, eps = 0.00001, step-max = 500, options = 'False', dataset = 'False'):**
  '''
  Функция-метод для нахождения минимума методом градиентного спуска с постоянным шагом

Входные данные:
y - функция в аналитическом виде, str
Lambda - константа шаг, float
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, шлаг о работе алгоритма и значения в точке экстремума. Возвращает точку минимума и dataset, если был указан соответствующий параметр, set
”’

- **def grad-with-crush-step (y, Lambda0, e, delta, eps = 0.00001, step-max = 500, options = 'False', dataset = 'False'):**
”’

Функция-метод для нахождения минимума методом градиентного спуска с дроблением шага
Входные данные:
y - функция в аналитическом виде, str
Lambda0 - начальный шаг, float
e - значение параметра оценки в пределах от 0 до 1, float
delta - значение параметра дробления в пределах от 0 до 1, float
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, шлаг о работе алгоритма и значения в точке экстремума. Возвращает точку минимума и dataset, если был указан соответствующий параметр, set
”’

- **def grad-with-fast-descent (y, eps = 0.00001, step-max = 500, options = 'False', dataset = 'False'):**
”’

Функция-метод для нахождения минимума методом наискорейшего спуска
Входные данные:
y - функция в аналитическом виде, 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 соответствующий параметр, set
”’

- **def grad-of-newton (y, eps = 0.00001, step-max = 500, options = 'False', dataset = 'False'):**
  '''

  Функция-метод для нахождения минимума методом Ньютона. Входные данные:
  y - функция в аналитическом виде, 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 соответствующий параметр, set
  '''

5.3 Функции вывода информации

- **def graph(y,x-solve):**
  '''

  Функция-метод для построения графика одномерной функции с указанием точки минимума
  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-grad-with-constant-step():**
  '''

  Функция-метод для объединения ввода и нахождения минимума для метода градиентного спуска с постоянным шагом
  '''

- **def all-f-grad-with-crush-step():**
  '''

  Функция-метод для объединения ввода и нахождения минимума для метода градиентного спуска с дроблением шага
  '''

- **def all-f-grad-fast-descent():**
  '''

  ФФункция-метод для объединения ввода и нахождения минимума для ме-

тода наискорейшего спуска
”’

- **def all-f-grad-of-newton():**
”’

Функция-метод для объединения ввода и нахождения минимума для метода Ньютона
”’

Тестирование

В разделе приводится тестирование работы программы.
Оптимальный способ представления результатов тестирования – это следующая таблица:

Таблица 1. Результаты тестирования и сравнение алгоритмов

Входные параметры	Градиентный спуск с постоянным шагом	Градиентный спуск с дроблением шага	Метод наискорейшего градиентного спуска	Алгоритм Ньютона - сопряженного градиента	WolframAlpha
Функция 1	f(x) = 10*x1**2+x2**2				
Дополнительные данные для ввода	Константа шаг Lambda = 0.01	Начальный шаг Lambda0 = 0.1, значение параметра оценки = 0.1, значение параметра дробления = 0.95	-	-	-
Полученное решение	{x2: '0.00048', x1: '0.00000'} f(x,y...) = 0.00000	{x1: '-0.00000', x2: '0.00001'} f(x,y...) = 0.00000	{x1: '-0.00000', x2: '0.00000'} f(x,y...) = 0.00000	{x1: '-0.00000', x2: '-0.00000'} f(x,y...) = 0.00000	{x1: '0', x2: '0'} f(x,y...) = 0
Время выполнения	CPU times: user 1.55 s, sys: 15.5 ms, total: 1.57 s	CPU times: user 476 ms, sys: 461 µs, total: 477 ms	CPU times: user 1.44 s, sys: 8.31 ms, total: 1.45 s	CPU times: user 716 ms, sys: 2.6 ms, total: 719 ms	Информация отсутствует
	Wall time: 1.64 s	Wall time: 794 ms	Wall time: 2.21 s	Wall time: 1.22 s	
Количество итераций	379	58	12	5	Информация отсутствует
Среднее время выполнения одной	40.897 µs	7.948 µs	692.5 µs	520 µs	Информация отсутствует

Функция 2	f(x) = x**3+2*y**2-3*x-4*y				
Дополнительные данные для ввода	Константа шаг Lambda = 0.001	Начальный шаг Lambda0 = 1, значение параметра оценки = 0.1, значение параметра дробления = 0.1	-	-	-
Полученное решение	{y: '1.00000', x: '1.00000'} f(x,y...) = -4.00000	{y: '1.00000', x: '1.00000'} f(x,y...) = -4.00000	{y: '1.00000', x: '1.00000'} f(x,y...) = -4.00000	{y: '1.00000', x: '1.00000'} f(x,y...) = -4.00000	{y: '1', x: '1'} f(x,y...) = -4
Время выполнения	CPU times: user 44.3 ms, sys: 2.84 ms, total: 47.2 ms	CPU times: user 14.1 ms, sys: 0 ns, total: 14.1 ms	CPU times: user 14.1 ms, sys: 852 µs, total: 14.9 ms	CPU times: user 11.2 ms, sys: 1.87 ms, total: 13.1 ms	Информация отсутствует
	Wall time: 108 ms	Wall time: 19.2 ms	Wall time: 26.4 ms	Wall time: 22.5 ms	
Количество итераций	2	2	2	1	Информация отсутствует
Среднее время выполнения одной	1.42 ms	0 ns	426 µs	1.87 ms	Информация отсутствует

Функция 3	f(x) = (x+2*y-7)**2+(2*x+y-5)**2				
Дополнительные данные для ввода	Константа шаг Lambda = 0.01	Начальный шаг Lambda0 = 0.1, значение параметра оценки = 0.1, значение параметра дробления = 0.2	-	-	-
Полученное решение	{y: '2.99966', x: '1.00034'} f(x,y...) = 0.00000	{y: '3.00000', x: '1.00001'} f(x,y...) = 0.00000	{y: '3.00000', x: '1.00000'} f(x,y...) = 0.00000	{y: '3.00000', x: '1.00000'} f(x,y...) = 0.00000	{y: '3', x: '1'} f(x,y...) = 0
Время выполнения	CPU times: user 2.61 s, sys: 365 µs, total: 2.61 s	CPU times: user 747 ms, sys: 237 µs, total: 747 ms	CPU times: user 1.83 s, sys: 8.26 ms, total: 1.84 s	CPU times: user 1.35 s, sys: 4.9 ms, total: 1.36 s	Информация отсутствует
	Wall time: 2.77 s	Wall time: 793 ms	Wall time: 2.87 s	Wall time: 1.55 s	
Количество итераций	396	58	12	7	Информация отсутствует
Среднее время выполнения одной	921.717 ns	4.086 µs	688.333 µs	700 µs	Информация отсутствует

Функция 4	f(x) = 2*x**2-1.05*x**4+x**6/6+x*y+y**2				
Дополнительные данные для ввода	Константа шаг Lambda = 0.01	Начальный шаг Lambda0 = 1, значение параметра оценки = 0.2, значение параметра дробления = 0.2	-	-	-
Полученное решение	{y: '0.00057', x: '-0.00024'} f(x,y...) = 0.00000	{y: '-0.00002', x: '0.00001'} f(x,y...) = 0.00000	{y: '-0.00000', x: '0.00000'} f(x,y...) = 0.00000	{y: '-0.00000', x: '0.00000'} f(x,y...) = 0.00000	{y: '0', x: '0'} f(x,y...) = 0
Время выполнения	CPU times: user 2.57 s, sys: 3.76 ms, total: 2.57 s	CPU times: user 400 ms, sys: 1.31 ms, total: 401 ms	CPU times: user 6.17 s, sys: 12.2 ms, total: 6.18 s	CPU times: user 5.07 s, sys: 17.2 ms, total: 5.09 s	Информация отсутствует
	Wall time: 2.68 s	Wall time: 410 ms	Wall time: 6.4 s	Wall time: 5.28 s	
Количество итераций	393	30	16	12	Информация отсутствует
Среднее время выполнения одной	9.567 µs	43.667 µs	762.5 µs	1.433 ms	Информация отсутствует

Функция 5	f(x) = sin(x+y)+(x-y)**2-1.5*x+2.5*y+1				
Дополнительные данные для ввода	Константа шаг Lambda = 0.01	Начальный шаг Lambda0 = 0.4, значение параметра оценки = 0.1, значение параметра дробления = 0.3	-	-	Константа шаг Lambda = 0.01
Полученное решение	{y: '-1.54439', x: '-0.54439'} f(x,y...) = -1.91321	{y: '-1.54720', x: '-0.54720'} f(x,y...) = -1.91322	{y: '-1.54720', x: '-0.54719'} f(x,y...) = -1.91322	{y: '-4.68879', x: '3.68879'} f(x,y...) = -5.05482	{y: '-1.5472', x: '-0.547198'} f(x,y...) = -1.91322
Время выполнения	CPU times: user 6.93 s, sys: 13.7 ms, total: 6.94 s	CPU times: user 577 ms, sys: 1.38 ms, total: 578 ms	CPU times: user 9.47 s, sys: 24.2 ms, total: 9.49 s	CPU times: user 6.75 s, sys: 25.2 ms, total: 6.77 s	Информация отсутствует
	Wall time: 7.18 s	Wall time: 628 ms	Wall time: 10.8 s	Wall time: 7.05 s	
Количество итераций	500	25	16	11	Информация отсутствует
Среднее время выполнения одной	27.4 µs	55.2 µs	1.513 ms	2.291 ms	Информация отсутствует

Функция 6	f(x) = 5*x**2-48*x+10*y**2-200*y+10576				
Дополнительные данные для ввода	Константа шаг Lambda = 0.01	Начальный шаг Lambda0 = 0.1, значение параметра оценки = 0.1, значение параметра дробления = 0.1	-	-	-
Полученное решение	{x: '4.79992', y: '10.00000'} f(x,y...) = 9460.80000	{x: '4.80000', y: '10.00003'} f(x,y...) = 9460.80000	{x: '4.80000', y: '10.00000'} f(x,y...) = 9460.80000	{x: '4.80000', y: '10.00000'} f(x,y...) = 9460.80000	{x: '4.8', y: '10'} f(x,y...) = 9460.8
Время выполнения	CPU times: user 476 ms, sys: 4.81 ms, total: 481 ms	CPU times: user 343 ms, sys: 3.21 ms, total: 346 ms	CPU times: user 1.87 s, sys: 10.1 ms, total: 1.88 s	CPU times: user 812 ms, sys: 5.8 ms, total: 818 ms	Информация отсутствует
	Wall time: 522 ms	Wall time: 364 ms	Wall time: 1.95 s	Wall time: 843 ms	
Количество итераций	103	57	10	5	Информация отсутствует
Среднее время выполнения одной	46.699 µs	56.316 µs	1.01 ms	1.16 ms	Информация отсутствует

По результатам тестирования алгоритмов можно сделать следующие выводы:

- Практически все ответы, полученные в результате работы алгоритмов, совпали с результатами WolframAlpha (в Функции 5 алгоритм Ньютона - сопряженного градиента дал иной результат);
- Быстрее всего система выполняла алгоритм градиентного спуска с дроблением шага;
- Наибольшее количество итераций наблюдалось в алгоритме градиентного спуска с постоянным шагом;
- Наименьшее количество итераций наблюдалось в алгоритме Ньютона - сопряженного градиента;
- Наименьшее среднее время выполнения системой итерации наблюдалось у алгоритма градиентного спуска с постоянным шагом (кроме Функции 2);
- Наибольшее среднее время выполнения системой итерации наблюдалось у алгоритма Ньютона - сопряженного элемента;

- Дополнительные данные для решения функции необходимо было вводить в алгоритмах градиентного спуска с постоянным шагом и градиентного спуска с дробление шага.

Таблица 2. Сравнение производительности приближенных алгоритмов и точных алгоритмов

Выбранный алгоритм	Метод наискорейшего спуска	Точный алгоритм	WolframAlpha
Функция 1	f(x,y) = x**3+2*y**2-3*x-4*y		
Полученное решение	x = 1	x = 1	x = 1
	y = 1	y = 1	y = 1
	f (x,y) = - 4	f (x,y) = - 4	f (x,y) = - 4
Время выполнения	Wall time: 41.2 ms	Wall time: 42 ms	Информация отсутствует
Функция 2	f(x,y) = (x+2*y-7)**2+(2*x+y-5)**2		
Выбранный алгоритм	Метод наискорейшего спуска	Точный алгоритм	WolframAlpha
Полученное решение	x = 1	x = 1	x = 1
	y = 3	y = 3	y = 3
	f(x,y) = 0	f(x,y) = 0	f(x,y) = 0
Время выполнения	Wall time: 3.08 s	Wall time: 134 ms	Информация отсутствует

Функция 3	f(x,y) = 10*x1**2+x2**2		
Выбранный алгоритм	Метод наискорейшего спуска	Точный алгоритм	WolframAlpha
Полученное решение	x = 0	x = 0	x = 0
	y = 0	y = 0	y = 0
	f(x,y) = 0	f(x,y) = 0	f(x,y) = 0
Время выполнения	Wall time: 1.82 s	Wall time: 47.8 ms	Информация отсутствует

Функция 4	f(x,y) = (x**2+y-11)**2+(x+y**2-7)**2		
Выбранный алгоритм	Метод наискорейшего спуска	Точный алгоритм	WolframAlpha
Полученное решение	x = 3	x = 3	x1=3, y1=2;
	y = 2	y = 2	x2=-3.77931, y2=-3.28319;
	f(x,y)= 0	f(x,y) = 0	x3=-2.80512, y3=3.13131;
			x4=3.58443, y4=-1.84813
			f(x,y)=0
Время выполнения	Wall time: 4.54 s	Wall time: 16.5 s	Информация отсутствует

Функция 5	$f(x, y) = 5*x^2 - 48*x + 10*y^2 - 200*y + 10576$		
Выбранный алгоритм	Метод наискорейшего спуска	Точный алгоритм	WolframAlpha
Полученное решение	x = 4.8	x = 4.8	x = 4.8
	y = 10	y = 10	y = 10
	f(x,y) = 9459.8	f(x,y) = 9459.8	f(x,y) = 9459.8
Время выполнения	Wall time: 2.37 s	Wall time: 35.8 ms	Информация отсутствует

По результатам сравнения производительности приближенных алгоритмов и точных алгоритмов можно сделать следующие выводы:

- Все ответы, полученные в результате работы алгоритмов, совпали с результатами WolframAlpha;
- На выполнение точного алгоритма было затрачено меньше времени, чем для выполнения метода наискорейшего спуска.
- Быстрее всего система выполняла точный алгоритм;
- Наименьшее среднее время выполнения системой итерации наблюдалось у точного алгоритма (кроме Функции 4);
- Наибольшее среднее время выполнения системой итерации наблюдалось у точного алгоритма.
- На основе полученных данных по тестированию, мы можем сделать соответствующие выводы. Результатом проведенных исследований может стать следующее заключение.

Заключение

Подводя итоги, можно констатировать следующее: согласно требованиям заказчика в среде программирования Python были реализованы функции, которые находят минимальные денежные затраты на переработку отходов производства предприятия и на покупку земли под застройку с помощью методов многомерной оптимизации.

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

Таблица 3. Сравнение алгоритмов.

Критерий	Градиентный спуск с постоянным шагом	Градиентный спуск с дроблением шага	Метод наискорейшего градиентного спуска	Алгоритм Ньютона - сопряженного градиента	WolframAlpha
Возможные параметры для ввода	Функция в аналитическом виде, константа шага, точность оптимизации, максимальное количество итераций, флаг для вывода промежуточных результатов, флаг для сохранения промежуточных результатов в dataframe.	Функция в аналитическом виде, начальный шаг, значение параметра оценки в пределах от 0 до 1, значение параметра дробления в пределах от 0 до 1, точность оптимизации, максимальное количество итераций, флаг для вывода промежуточных результатов, флаг для сохранения промежуточных результатов в dataframe.	Функция в аналитическом виде, константа шага, точность оптимизации, максимальное количество итераций, флаг для вывода промежуточных результатов, флаг для сохранения промежуточных результатов в dataframe.	Функция в аналитическом виде, константа шага, точность оптимизации, максимальное количество итераций, флаг для вывода промежуточных результатов, флаг для сохранения промежуточных результатов в dataframe.	Функция в аналитическом виде, границы области оптимизации
Точность выводимых значений	До пяти знаков после запятой	До пяти знаков после запятой	До пяти знаков после запятой	До пяти знаков после запятой	До пяти знаков после запятой
Полнота вывода ответа	Выводится в печатном виде локальный минимум. Выводятся координаты точки локального минимума. При запросе пользователя выводятся промежуточные результаты на каждой итерации.	Выводится в печатном виде локальный минимум. Выводятся координаты точки локального минимума. При запросе пользователя выводятся промежуточные результаты на каждой итерации.	Выводится в печатном виде локальный минимум. Выводятся координаты точки локального минимума. При запросе пользователя выводятся промежуточные результаты на каждой итерации.	Выводится в печатном виде локальный минимум. Выводятся координаты точки локального минимума. При запросе пользователя выводятся промежуточные результаты на каждой итерации.	Выводятся в печатном и графическом виде все локальные минимумы. Выводятся координаты точек экстремума.
Формат вывода ответа	Значение минимума функции и координаты точки минимума выводятся в форме десятичного числа. При запросе пользователя выводятся промежуточные результаты на каждой итерации в виде координат точки и значения функции на определенной итерации с подписью номера итерации.	Значение минимума функции и координаты точки минимума выводятся в форме десятичного числа. При запросе пользователя выводятся промежуточные результаты на каждой итерации в виде координат точки и значения функции на определенной итерации с подписью номера итерации.	Значение минимума функции и координаты точки минимума выводятся в форме десятичного числа. При запросе пользователя выводятся промежуточные результаты на каждой итерации в виде координат точки и значения функции на определенной итерации с подписью номера итерации.	Значение минимума функции и координаты точки минимума выводятся в форме десятичного числа. При запросе пользователя выводятся промежуточные результаты на каждой итерации в виде координат точки и значения функции на определенной итерации с подписью номера итерации.	Некоторые значения выводятся в формате дробей (в том числе с корням), остальные в форме десятичного числа

Время выполнения	Время выполнения для разных функций:	Время выполнения для разных функций:	Время выполнения для разных функций:	Время выполнения для разных функций:	Время выполнения не выводится
	1.64 s	749 ms	2.21 s	1.22 s	
	108 ms	19,2 ms	26.4 ms	22.5 ms	
	2.77 s	793 ms	2.87 s	1.55 s	
	2.68 s	410 ms	6.4 s	5.28 s	
	7.18 s	628 ms	10.8 s	7.05 s	
	522 ms	364 ms	364 ms	843 ms	
Количество итераций	379	58	12	5	Количество итераций не выводится
	2	2	2	1	
	396	58	12	7	
	393	30	16	12	
	500	25	16	11	
	103	57	10	5	

На основе таблиц 1, 2, 3 мы делаем вывод о том, что в качестве оптимального решения поставленной задачи лучше всего подходят методы наискорейшего градиентного спуска и алгоритм Ньютона - сопряженного градиента, т. к. у них наиболее оптимальные шаги и наиболее приближенные к реальным значениям минимумы, по сравнению с остальными алгоритмами.

Также мы вычислили, что на первый завод будет передано 4.8 тонн мусора, а на второй 19.2. Затраты на мусор будут составлять 460.8 д.е.

После этого было подсчитано, что в первом поселке будет куплено 10 соток земли, во втором - 90 соток. Затраты на землю будут составлять 9000 д.е.

По приведенным выше результатам можно сделать вывод, что минимальные затраты заказчика составят 9460.8 д.е.

Уважаемый заказчик, наша команда создала модель, которая позволяет решить эту поставленную задачу и удовлетворяет всем условиям. Мы рассчитываем на дальнейшее сотрудничество. В случае заключения договора, на долгосрочной основе, мы готовы отслеживать процессы затрат на вашем производстве, и производить плановый пересчет.