**Федеральное государственное образовательное**

**бюджетное учреждение**

**высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ**

**ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ**

**ФЕДЕРАЦИИ»**

**(Финансовый университет)**

**Факультет**

**информационных технологий и анализа больших данных**

**Направление «Прикладная математика и информатика»**

**Домашнее задание № 2**

«Методы одномерной оптимизации»

Студенты группы ПМ19-3

Караваев Артем Евгеньевич

Лазаренко Владлена Владимировна

Минаков Артем Дмитриевич

Пластун Екатерина Сергеевна

Голомысов Даниил Владиславович

Руководитель:

Аксенов Дмитрий Андреевич

**Москва 2022**

Оглавление.

1. Математическая модель (постановка задачи)
2. Алгоритмы
   1. Алгоритм 1
      1. Описание входных данных
      2. Описание алгоритма решения
      3. Описание выходных данных
   2. Алгоритм 2
      1. Описание входных данных
      2. Описание алгоритма решения
      3. Описание выходных данных
   3. Алгоритм 3
      1. Описание входных данных
      2. Описание алгоритма решения
      3. Описание выходных данных
3. Варианты использования системы
   1. ВИ 1
   2. ВИ 2
4. Архитектура решения
   1. Функции считывания информации
   2. Функции обработки информации
   3. Функции вывода информации
5. Тестирование
6. Выводы и заключение
7. **Постановка задачи (физическая модель)**

К нам пришел заказчик с новым проектом по открытию фирмы. В его планируемом производстве настольные лампы будут продаваться по цене 1200 рублей каждая. Затраты при производстве 30 ламп будут составлять 48 тысяч рублей, при производстве 70 ламп – 80 тысяч рублей. Заказчик может производить не более 90 и не менее 30 ламп. Цель заказчика минимизировать издержки на производство в его новой фирме по производству настольных ламп.

1. **Математическая модель**

Обозначим х - количество изготавливаемых и реализуемых настольных ламп. Тогда выручка от их продажи составит:

**R(x) = p\*x = 1200x**

Воспользуемся уравнением прямой для определения функции издержек:

1. **Алгоритмы**
   1. **Алгоритм 1**

**Поиск экстремума функции одной переменной методом золотого сечения.**

Метод основан на делении текущего отрезка [a; b], где содержится искомый экстремум, на две неравные части, подчиняющиеся правилу золотого сечения, для определения следующего отрезка, содержащего максимум.

* + 1. **Описание входных данных**

**Обязательные параметры:**

а) Функция в аналитическом виде (например: x^2);

б) Границы области оптимизации [a; b];

**Обязательные параметры:**

а) Точность оптимизации (по умолчанию 0,00001)

б) Максимальное количество шагов (по умолчанию 500)

* + 1. **Описание алгоритма решения**

1. Определяем начальные значения х1 и х2.
2. Вычисляем значения *Y1 = f(x1), Y2 = f(x2).*

2.1. Если Y1 < Y2, то задаем новые значения: b = x2, x2 = x1,  *, Y2 = Y1, Y1 = f(x1).*

2. 2. Если Y1 >= Y2, то задаем новые значения: a = x1, x1 = x2,  *, Y1 = Y2, Y2 = f(x2).*

1. **Пункт 2.** повторяется до тех пора, пока модуль |b – a| не станет меньше ԑ. Когда это условие будет достигнуто, то х = а, f(x) = *Y1*.

**Вывод: х, f(x).**

* + 1. **Описание выходных данных**

а) Найденное значение координаты точки экстремума (например: 0,1);

б) Значение функции в точке экстремума (например: 0,897);

в) Отчет о работе алгоритма (Отчет о работе алгоритма, например, флаг: 0- найдено значение с заданной точностью; 1-достигнуто максимальное количество итераций; 2-выполнено с ошибкой).

* 1. **Алгоритм 2**

**Поиск экстремума функции одной переменной методом парабол.**

Метод парабол является представителем группы методов, основанных на аппроксимации целевой функции некоторой более простой функцией (как правило полиномом), минимум которой можно легко найти. Точка минимума этой аппроксимируещей функции и принимается за очередное приближение точки минимума целевой функции.

* + 1. **Описание входных данных**

**Обязательные параметры:**

а) Функция в аналитическом виде (например: x^2);

б) Границы области оптимизации (например: -1 1);

* + 1. **Описание алгоритма решения**

1. Выбираем три точки из отрезка [a;b] так, чтобы выполнялись условия:
2. Вводим обозначения:

**,**

и вычисляем:

1. Вычисляем:

1. Находим

где – номер итерации.

Процедура повторяется, пока || > ε.

Если Δ (||) < ε (заданная точность), то поиск заканчиваем и **.** Если это условие не выполняется, то вся процедура повторяется с пункта **1)**, то есть мы выбираем новые точки х1, х2, х3и все заново рассчитываем.

1. **Вывод: х, f(x).**
   * 1. **Описание выходных данных**

а) Найденное значение координаты точки экстремума;

б) Значение функции в точке экстремума;

в) Отчет о работе алгоритма (например флаг: 0 – найдено значение с заданной точностью; 1 – достигнуто максимальное количество итераций; 2 – выполнено с ошибкой).

* 1. **Алгоритм 3**

**Поиск экстремума функции одной переменной комбинированным методом Брента.**

Метод Брента комбинирует в себе метод парабол и метод золотого сечения, совмещая в себе высокую скорость сходимость одного и стабильность другого метода. В малой окрестности точки минимума функции метод парабол показывает высокую скорость сходимости, превосходящую скорости сходимости методов деления отрезка пополам и золотого сечения.

* + 1. **Описание входных данных**

**Обязательные параметры:**

а) Функция в аналитическом виде (например: x^2);

б) Границы области оптимизации (например: -1 1);

* + 1. **Описание алгоритма решения**

1. Считаем начальные значения:

x, w, v = a + r\*(b – a), где .

, .

1. Пока не превышен лимит шагов выполняем следующее:
   1. Если max{x - a, b – x} < ԑ, то x – найденное решение, иначе g = ,
   2. Строим параболу по трем точкам x, w, v и находим ее вершину u.
   3. Если u не найдена или u [a,b] или |u – x| > g, то:
      1. Если x < , то u = x + r\*(b – x), В противном случае u = x - r\*(x – a),
   4. = |u – x|.
   5. f(u) > f(x), то:
      1. Если u < x, то a = u. В противном случае b = u.
      2. Если f(u) <= f(w) или w = x, то v = w, w = u. В противном случае, если f(u) <= f(v) или v = x или v = w, то v = u.

В противном случае:

* + 1. Если u < x, то b = x. В противном случае a = x, v = w, w = x, x = u.

**Вывод: х, f(x).**

* + 1. **Описание выходных данных**

а) Найденное значение координаты точки экстремума;

б) Значение функции в точке экстремума;

в) Отчет о работе алгоритма (например флаг: 0 – найдено значение с заданной точностью; 1 – достигнуто максимальное количество итераций; 2 – выполнено с ошибкой).

* 1. **Алгоритм 4**

**Алгоритм Бройдена-Флетчера-Гольдфарба-Шанно.**

Данный алгоритм относится к классу так называемых квазиньютоновских методов. В отличие от ньтоновских методов в квазиньютоновских не вычисляется напрямую гессиан функции, то есть нет необходимости находить частные производные второго порядка. Вместо этого гессиан вычисляется приближенно, исходя из сделанных до этого шагов.

* + 1. **Описание входных данных**

**Обязательные параметры:**

а) Функция в аналитическом виде;

б) Начальная точка;

**Необязательные параметры:**

в) Параметр для первого условия Вольфе (по умолчанию: 10^(-4));

г) Параметр для второго условия Вольфе (по умолчанию: 0.1);

д) Максимально возможное значение аргумента функции (по умолчанию 100);

е) Максимальное число итераций (по умолчанию: 500);

* + 1. **Описание алгоритма решения**

1. Инициализируем начальную точку х0. Задаем точность ԑ. Определяем начальное приближение, , где – обратный гессиан функции.  
   В качестве начального приближения можно взять гессиан функции, вычисленный в начальной точке.
2. Находим точку, в направлении которой будем производить поиск, она определяется следующим образом:
3. Вычисляем через рекуррентное соотношение:

Коэффициент k находится при помощи линейного поиска, где k удовлетворяет следующим условиям Вольфе:

0 <= c1 <= c2 <= 1.

1. Определяем вектора:

Где – шаг алгоритма на итерации, – изменение градиента на итерации.

1. Обновляем гессиан функции по следующей формуле:

Где , I – единичная матрица.

1. Алгоритм выполняется до тех пор, пока || > ԑ.

* + 1. **Описание выходных данных**

а) Найденное значение координаты точки экстремума;

б) Значение функции в точке экстремума;

г) Отчет о работе алгоритма (например флаг: 0 – точка удовлетворяющая условию Вольфе найдена; 1 – точка удовлетворяющая условию Вольфе найдена с заданной точностью; 2 – достигнуто максимальное количество итераций; 3 – Достигнуто ограничение на максимально возможное значение аргумента; 4 – выполнено с ошибкой).

1. **Варианты использования системы**
   1. **ВИ 1 – Метод золотого сечения.**
      1. **Ввод данных.**
         1. **Ввод названия переменной.**

Вы можете ввести любую английскую или русскую строчную букву, после чего нажмите «Enter». Пример:



* + - 1. **Ввод функции в аналитическом виде.**

Вводи функцию в аналитическом виде, используя то же название переменной, что и в **4.2.1.1.** Можно использовать тригонометрические функции, такие как sin, cos, tg. Также можно использовать ln (натуральный логарифм), log (десятичный логарифм), exp(экспонента). В качестве знака умножения используем \*, деления /, возведения в степень \*\*.

Функцию вводится без пробелов.



или



или



* + - 1. **Ввод границ производимой продукции.**

Отрезок [a; b] в программе вводится поочередно. Сначала программа попросит вас ввести левую границу (а), потом правую границу (b). Десятичное число пишется через точку.





* + - 1. **Вывод полученных результатов.**

****

**Первая строка** выводит найденный минимум функции.

**Вторая** строка вывод значение в найденной точке минимума.

**Третья** строка вывод количество итераций, которые потребовались для нахождения результатов.

**Четвертая** строка показывает, сколько времени потребовалось для выполнения алгоритма.

График показывает зависимость значения функции от номера итерации.

* 1. **ВИ 2 – Метод парабол.**
     1. **Ввод данных.**
        1. **Ввод названия переменной.**

Вы можете ввести любую английскую или русскую строчную букву, после чего нажмите «Enter». Пример:



* + - 1. **Ввод функции в аналитическом виде.**

Вводи функцию в аналитическом виде, используя то же название переменной, что и в **4.2.1.1.** Можно использовать тригонометрические функции, такие как sin, cos, tg. Также можно использовать ln (натуральный логарифм), log (десятичный логарифм), exp(экспонента). В качестве знака умножения используем \*, деления /, возведения в степень \*\*.

Функцию вводится без пробелов.



или



или

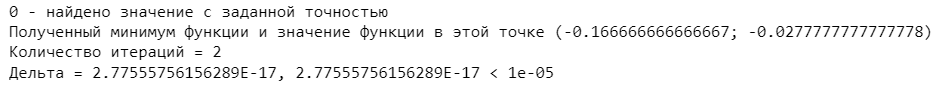


* + - 1. **Ввод границ производимой продукции.**

Отрезок [a; b] в программе вводится двумя числами через пробел, где первое число – это a, второе – b. Десятичное число пишется через точку. Функция на этом отрезке должна быть непрерывной и унимодальной.



* + - 1. **Вывод полученных результатов.**

****

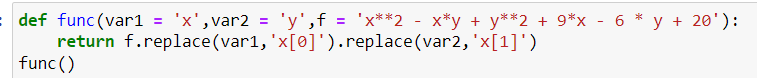
**Первая строка** – результат о работе алгоритм. Вывод может быть в трех вариантах: 0 – найдено значение с заданной точностью, 1 – достигнуто максимальное количество итераций, 3 – удачную тройку чисел найти не удалось.

**Вторая** строка выводит найденный минимум функции и ее значений в этой точке.

**Третья** строка вывод количество итераций, которые потребовались для нахождения результатов.

**Четвертая** строка показывает, что точность работы алгоритма меньше заданной.

* 1. **ВИ 4**
     1. **Ввод данных.**
        1. **Ввод названия переменных и функции в аналитическом виде.**
           1. Для ввода данных для решения, вам нужно выполнить следующее:



* В функции func в скобках присвоить (написать через равно) var1 название первой переменной в кавычках
* присвоить var2 название второй переменной в кавычках,
* присвоить f функцию в кавычках в аналитическом виде.

\*В качестве знака умножения используем \*, деления /, возведения в степень \*\*.

На выводе получим функцию, которая нам понадобится для следующих шагов: 

* + - * 1. По аналогии заполняем данные для следующей функции gradFunc():

****

На выводе получим список из 2 функций, которые нам понадобятся для следующих шагов: 

* + - * 1. Полученные функции раннее теперь введем в следующие 2 функции f(x), f1(x):

****

**После return вводим ф-цию, полученную в пункте 4.4.1.1.1:**



****

**После return np.array вводим ф-цию в ([]), полученную в пункте 4.4.1.1.2:**



* + 1. **Вывод полученных результатов.**

**Изображение выглядит как текст

Автоматически созданное описание**

**Первая строка** выводит координаты минимума функции.

**Вторая** строка вывод значение в найденной точке минимума.

**Третья** строка выводит информацию о нахождении результатов.

1. **Архитектура решения**
   1. **Метод золотого сечения.**
      1. **Функция считывания информации.**

**def inputzolotoe():**

"""

Функция для ввода данных, с помощью которого будет осуществляться метод золотого сечения

Parameters

----------

param: str

Строка на вход с одной переменной, которая может быть задана любой русской или английской буквой

f: str

Функция в аналитическом виде в формате строки

a: str

Левая граница оптимизации в формате строки

b: str

Правая граница оптимизации в формате строки

Final: dictionary

Словарь со всеми переменными и функциями, которые ранее были введены

"""

* + 1. **Функция обработки и вывода информации.**

**def zolotoe(data):**

"""

Функция, которая осуществляет алгоритм метода золотого сечения, а также построения графика зависимости значения функции от номера итерации, описанный в пункте 3.1.2.

Parameters

----------

data: dictionary

Словарь, со всеми переменными, которые были заданы в функции ввода

f: str

Функция в аналитическом виде в формате строки

a: float

Левая граница оптимизации

b: float

Правая граница оптимизации

e: float

Точность алгоритма

alpha: float

Константа, используемая для расчетов

n: int

Количество итераций алгоритма

k: list

Список со значением функции на каждой итерации

x1, x2, y1, y2: float

Переменные, используемые для осуществления метода золотого сечения.

Returns

---------

a: float

Найденная точка минимума

y1: float

Значение функции в точке минимума

i: int

Количество итераций

"""

* 1. **Метод парабол.**
     1. **Функция считывания информации.**

**def inputParabol():**

"""

Функция для ввода данных, с помощью которого будет осуществляться метод парабол, описанный в пункте 3.2.2.

Parameters

----------

params: str

Строка на вход с одной переменной, которая может быть задана любой русской или английской буквой

F: str

Функция в аналитическом виде в формате строки

lim1: str

Границы оптимизации через пробел в формате строки

Final: dictionary

Словарь со всеми переменными и функциями, которые ранее были введены

"""

* + 1. **Функция обработки и вывода информации.**

**def parabola(dictionary, eps=0.00001, maxk=500):**

"""

Функция, которая осуществляет алгоритм метода парабол, описанный

Parameters

----------

eps: float

точность оптимизации

maxk: int

максимальное количество шагов алгоритма

dictionary: dict

Словарь, со всеми переменными, которые были заданы в функции ввода

f: str

Функция в аналитическом виде в формате строки

lim1: list

Границы области оптимизации

Returns

---------

x\_: float

Найденная точка минимума

f\_: float

Значение функции в точке минимума

k: int

Количество итераций

"""

* 1. **Алгоритм Бройдена-Флетчера-Гольдфарба-Шанно**
     1. **Функции считывания информации**

**def f()**

"""

Функции для перевода заданной функции из аналитического вида в нужный для дальнейших вычислений.

Parameters

-----------

x: str

Переменная х.

Returns

-----------

Заданная функция для дальнейших вычислений.

"""

**def f1()**

"""

Функции для перевода градиента заданной функции из аналитического вида в нужный для дальнейших вычислений.

Parameters

-----------

x: str

Переменная х.

Returns

-----------

Массив из 2 строк с градиентами заданной функция для дальнейших вычислений.

"""

**def func()**

"""

Функция для ввода аналитической функции, для которой будут находиться экстремум.

Parameters

-----------

Var1: str

Строка на вход c переменной х

Var2: str

Строка на вход c переменной y

f: str

Строка на вход c функцией f в аналитическом виде.

Returns

-----------

f: str

Функция f в аналитическом виде, где вместо х,у х[0], x[1].

"""

**def gradFunc()**

"""

Функция для вычисления градиента функции двух переменных.

Parameters

-----------

Var1: str

Строка на вход c переменной х

Var2: str

Строка на вход c переменной y

f: str

Строка на вход c функцией f в аналитическом виде.

Returns

-----------

f: str

Градиенты функции f в аналитическом виде по первой и второй переменным.

"""

* + 1. **Функция обработки информации и вывода информации**

**def BFGS()**

"""

Функция для нахождения координаты точки экстремума и значения ф-ции в этой точке.

Parameters

-----------

func:

Функция.

grad: np.array

Градиенты функции.

x0: list

Список с координатами точки.

v1: float

Параметр для 1 условия Вольфа.

v2: float

Параметр для 2 условия Вольфа.

xmax : int

Максимально возможное значение аргумента ф-ции.

interv: float

Порог выхода по длине интервала поиска

maxiter: int

Максимальное количество итераций.

p1: bool

Флаг «вывод промежуточных результатов»

p2: bool

Флаг «запись промежуточных результатов в датасет»

Returns

-----------

xk: list

Список координат точки экстремума

func(xk)

Значение ф-ции в точке экстремума

0/1/2/3/4

Отчет о работе алгоритма

"""

1. **Тестирование**

Таблица 1. Результаты тестирования метода парабол

|  |  |  |
| --- | --- | --- |
| Функция | Метод парабол | Стороннее решение |
|  | x = 0.1098  F(x) = 0.8976 | x = 0.11  F(x) = 0.898 |
|  | x = 8.50158  F(x) = 2.1338 | x = 8.621  F(x) = 2.368 |
|  | x = 2.70647  F(x) = -7.2743 | x = 2.706  F(x) = -7.2706 |
|  | x = 0.2407  F(x) = 5.1483 | x = 0.241  F(x) = 5.14834107 |
|  | x = 2.2174  F(x) = 0.3682 | x = 2.2175  F(x) = 0.36828 |

Таблица 2. Результаты тестирования метода золотого сечения

|  |  |  |
| --- | --- | --- |
| Функция | Золотое сечение | Стороннее решение |
|  | x = 0.1098  F(x) = 0.8976 | x = 0.11  F(x) = 0.898 |
|  | x = 8.50158  F(x) = 2.1338 | x = 8.621  F(x) = 2.368 |
|  | x = 2.70647  F(x) = -7.2743 | x = 3.012  F(x) = -0.35 |
|  | x = 0.2407  F(x) = 5.1483 | x = 0.241  F(x) = 5.14834107 |
|  | x = 2.2174  F(x) = 0.3682 | x = 2.5  F(x) = 0.558 |

Таблица 3. Результаты тестирования **алгоритма Бройдена-Флетчера-Гольдфарба-Шанно**

|  |  |  |
| --- | --- | --- |
| Функция | Метод Бройдена-Флетчера-Гольдфарба-Шанно. | Стороннее решение |
| x/(x^2 + y) | Выполнено с ошибкой, альфа не может быть найдена с такими условиями | Вопрос о наличии экстремума остается открытым |
| (x+y)^5 - 2 \* (x+y)^4 | X=1.29999973 y=0.29999973  F(x,y)= -2.621439999996918  (при заданном начальном условии x0 =[1,0]) | Вопрос о наличии экстремума остается открытым |
| x^3+x\*y^2+x^2+y^2 | X=1.29999973 y=0.29999973  F(x,y)= -2.621439999996918  (при заданном начальном условии x0 =[1,0]) | x = 0  y=0  F(x,y) = 0 |
| x^3+3x\*y^2-15x-12y | X=1.99999751 y=0.99999747  F(x,y)=-27.999999999886388 | x = 2  y= 1  F(x,y) = -28 |
| x^2\*y^2+0.5x^2+0.5y^2+x\*y+1 | X=-0.0076573 y=0.00762688  F(x,y)=1.0000000038733976 | x = 0  y= 0  F(x,y) = 1 |

Таблица 5. Время/колво итераций выполнения в секундах.

|  |  |  |
| --- | --- | --- |
| Функция | Золотое сечение | Метод парабол |
|  | 0,0295/24 | 0,0254/8 |
|  | 0,0502/27 | 0,032/21 |
|  | 0,0350/28 | 0,0206/6 |
|  | 0,0425/24 | 0,0126/5 |
|  | 0,0408/26 | 0,0557/8 |

Таблица 6. Время выполнения в секундах.

|  |  |
| --- | --- |
| Функция | BGFS |
| x/(x^2 + y) | 0.0009963512420654297 |
| (x+y)^5 - 2 \* (x+y)^4 | 0.0019664764404296875 |
| x^3+x\*y^2+x^2+y^2 | 0.0019958019256591797 |
| x^3+3x\*y^2-15x-12y | 0.0019958019256591797 |
| x^2\*y^2+0.5x^2+0.5y^2+x\*y+1 | 0.0029914379119873047 |

По результатам вычисления скорости, самый эффективный метод для использования - метод BGFS.

1. **Заключение**

**Решение методом золотого сечения:**

Х = 30

Y = 48000,002

Точность – 0,00001

**Решение методом парабол:**

3 - Не удалось найти удачную тройку чисел

По результатам сравнительного анализа методов, оптимальный для заказчика, оказался алгоритм золотого сечения.