Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 1

Тема «Оптимизация функции одной переменной»

по дисциплине «Теория принятия решений»

Выполнил: студент группы ИСТ-22-1б Петраков М.В.

Проверил: доцент каф. ВММБ Бояршинова И. Н.

Пермь, 2024

Постановка задания

Найти минимум функции одной переменной тремя методами: одним из методов нулевого порядка, одним из методов первого порядка и методом второго порядка с проверкой применимости метода к заданной функции 

1. Обоснование применимости методов

Для выяснения возможности применить методы оптимизации необходимо рассмотреть график функции

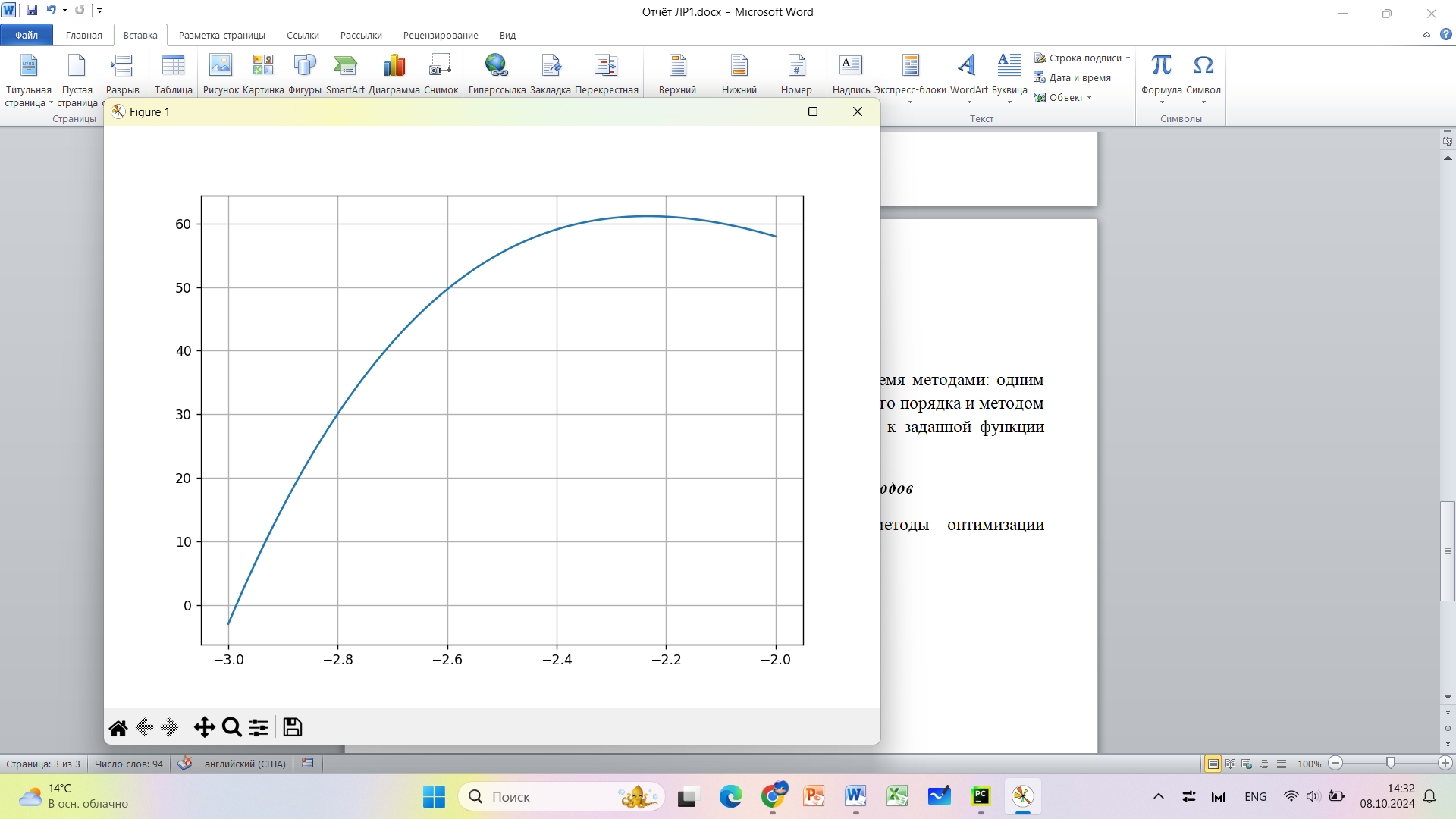


Рис. 1 – график функции 

Из графика видно, что функция принимает только одно минимальное и одно максимальное значение, а также не является выпуклой. Из этого следует невозможность применения методов первого и второго порядка для поиска минимума. Однако остаётся возможность найти максимум функции. Тогда можно переходить к описанию применяемых методов.

2. Краткое описание алгоритма методов

Применяются: метод золотого сечения, метод касательных и метод Ньютона.

Метод золотого сечения:

Применимость этого метода будет зависеть от унимодальности функции. У функции  есть только одна точка минимума, поэтому метод золотого сечения доступен для реализации. Метод состоит в том, чтобы найти две точки, которые делят отрезок так, чтобы длина меньшей части отрезка к длине большей части относилась так же, как длина большей части ко всему отрезку. Далее представлена реализация метода золотого сечения на языке программирования Python.

def gold\_glowing(a, b, epsilon):  
 # обозначаются параметры  
 ak = a  
 bk = b  
 eps = epsilon  
 # затем точки  
 x1 = ak + ((3 - math.sqrt(5)) / 2) \* (bk - ak)  
 x2 = ak + ((math.sqrt(5) - 1) / 2) \* (bk - ak)  
 # и задаётся количество итераций  
 iterations = 0  
 while abs(bk - ak) >= eps:  
 # новая итерация  
 iterations += 1  
 # print(f"b - a = {bk - ak}")  
 # Найти разницу между двумя переменными  
 # В следующей итерации она будет определять расстояние от начала отрезка до точки золотого сечения  
 c = x2 - x1  
 if f(x1) >= f(x2):  
 # здесь точка деления находится слева от середины  
 ak = x1  
 bk = bk  
 x1 = x2  
 x2 = bk - c  
 else:  
 # здесь точка деления находится справа  
 ak = ak  
 bk = x2  
 x2 = x1  
 x1 = ak + c  
 """после завершения поиска необходимого отрезка выбирается точка  
 посередине найденного отрезка, и от неё находится значение функции"""  
 xmin = (ak + bk) / 2  
 ymin = f(xmin)  
 return xmin, ymin, iterations

Результатом стал кортеж чисел (-2.9999995651470845, -2.99991085519299, 29), где: первое число, равное -2.9999995651470845, – точка минимума функции, второе число, равное 2.99991085519299, - минимум функции на отрезке, третье число, равное 29, - число итераций.

В случае поиска максимума функции метод золотого сечения также срабатывает. Результатом является кортеж (-2.234022780570286, 61.18062461689776, 29), где аналогично поиску минимума первое число – точка максимума, второе – максимум функции на отрезке, третье – число итераций.

Метод касательных:

Применимость этого метода будет зависеть от унимодальности и функции. У данной функции на отрезке [-3; -2] есть один минимум, но он находится в точке -3, а градиентные методы ищут конкретно точку экстремума. В данном случае метод касательных сможет найти максимум функции на отрезке. Метод заключается в том, чтобы находить точку пересечения касательных, затем точки пересечения касательной к найденной точке с первыми двумя, сравнить значения производной в первой найденной точке (она посередине) с нулем и в зависимости от этого уменьшить отрезок поиска. Далее представлена реализация метода касательных, использующая дополнительную функцию для поиска точки пересечения.

# функция поиска пересечения двух касательных (точки на входе - переменные,  
# от которых находится производная функции)  
def intersection(xl, xr):  
 xm = (df(xl) \* xl - df(xr) \* xr + f(xr) - f(xl)) / (df(xl) - df(xr))  
 return xm

def tangles(a, b, epsilon):  
 # найти точку пересечения касательных к функции в концах заданного отрезка  
 xm = intersection(a, b)  
 xm\_1 = xm \* 10000  
 # завести количество итераций для метода  
 iterations = 0  
 # в сущности необходима близость модуля производной искомой точки к нулю  
 # из-за использования импортируемых функций (numpy diff) точность ниже 10^(-4) вызывает ошибку  
 while abs(df(xm)) >= epsilon and abs(f(xm)-f(xm\_1)-df(xm\_1)\*(xm-xm\_1))>=epsilon:  
 # print(xm)  
 iterations += 1  
 # Найти правые и левые точки пересечения касательных к ф. в концах отрезка с оной в ранее найденной т.  
 xl = intersection(a, xm)  
 xr = intersection(xm, b)  
 # если производная в точке оказывается больше нуля, функция справа от этого значения только возрастает  
 # стало быть, необходимо уменьшить отрезок поиска, выбросив все значения после точки xr  
 if df(xm) > 0:  
 xr = xm  
 # если же производная оказывается меньше нуля, функция слева от точки только убывает  
 # тогда необходимо отбросить диапазон до xl  
 elif df(xm) < 0:  
 xl = xm  
 # после уменьшения отрезка находится новая точка пересечения касательных,  
 # которую можно проверить на близость к минимуму  
 a = xl  
 b = xr  
 xm\_1 = xm  
 xm = intersection(xl, xr)  
 return xm, f(xm), iterations

Минимум этой конкретной функции метод касательных не найдет, но найдет максимум. Результатом работы метода является кортеж (-2.4644147304374884, 56.99054882782992, 8), где первое число – точка максимума, второе – максимум функции, третье – число итераций. Уже здесь можно отметить отклонение примерно на 10% от настоящего максимума.

Метод Ньютона:

Применимость этого метода зависит от выпуклости функции, что является более строгим условием унимодальности. Данная функция является выпуклой вверх, так что метод Ньютона найдет точка минимума для функции –f(x), что будет являться точкой максимума для изначальной функции. Метод заключается в том, чтобы найти первую и вторую производные в точке, разницу между и точкой и отношением первой и второй производной в точке, после чего сравнить полученную точку с предыдущей и значения функции в этих точках.

# метод Ньютона  
def newton(a, epsilon):  
 # обозначить точки для итерации (текущая и следующая)  
 xk = a  
 xk\_1 = a+0.1  
 iterations = 0  
 while abs(xk - xk\_1) >= epsilon or abs(f(xk) - f(xk\_1)) >= epsilon:  
 # после перейти к вычислению следующего значения с сохранением предыдущего  
 iterations += 1  
 if iterations > 1:  
 xk = xk\_1  
 df1 = df(xk)  
 d2f1 = d2f(xk)  
 xk\_1 = xk - (df1 / d2f1)  
 return xk\_1, f(xk\_1), iterations

У функции  на отрезке [-3;-2] метод Ньютона не найдёт точку минимума, но сможет найти точку максимума. Результатом работы является кортеж (-2.234022892850585, 61.180624616898626, 6), где первое число – точка максимума, второе число – максимум на отрезке, третье – число итераций.

Выводы о работе методов

Метод нулевого порядка имеет большее количество действий в одной итерации в сравнении с остальными методами, а количество итераций для остановки работы метода в несколько раз больше. Из этого следует большее время работы. Метод первого порядка имеет меньше действий в одной итерации, чем метод нулевого порядка, но больше, чем метод второго порядка. Самым большим недостатком является меньшая точность решения. Достоинством можно посчитать относительно небольшое количество итераций, что даёт меньшее количество времени работы. Метод Ньютона самый точный и самый быстрый: малое количество действий и малое количество итераций. Ценой является меньшая применимость этого метода: только выпуклые функции.