МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»

(ННГУ)

Институт информационных технологий, математики и механики

Передовая инженерная школа

Направление подготовки: «Прикладная информатика»

Отчет по лабораторной работе

на тему:

**«Метод Нелдера – Мида»**

**Выполнили:**

студенты группы 3821Б1ПИмэ

Анненко М.Ю.

Мазникер А.А.

И студенты группы 3821Б1ПИсм

Шальнов Ю.Д.

Гончарова А.Н.

Соменкова Е.А.

**Преподаватель:**

Сморякова Валентина Михайловна

Нижний Новгород

2024

Оглавление

[Введение 3](#_Toc167486523)

[Цель работы: 4](#_Toc167486524)

[Теоретический материал 4](#_Toc167486525)

[Алгоритм 5](#_Toc167486526)

[Идентификация функций, проходящих через метод 7](#_Toc167486527)

[Функция Химмельблау 7](#_Toc167486528)

[Вывод 10](#_Toc167486529)

[Примечание 11](#_Toc167486530)

[Источники: 12](#_Toc167486531)

# Введение

Метод Нелдера-Мида, также известный как метод симплекс-метода, является одним из наиболее популярных методов оптимизации без градиента. Он был разработан в 1965 году Джоном Нелдером и Роджером Мидом и является простым и эффективным инструментом для нахождения минимума (или максимума) функции без использования производных.  
  
Основная идея метода Нелдера-Мида заключается в использовании алгоритма поиска по образцу, который использует набор точек (или симплекс) для приближенного вычисления оптимального значения функции. С помощью итеративного процесса, метод изменяет размер и форму симплекса, приближаясь к оптимальному решению.  
  
Этот метод хорошо подходит для задач оптимизации, где функция не имеет явного аналитического выражения или не может быть легко дифференцирована. Он также обладает свойством устойчивости к шуму в данных и способности обрабатывать функции с несколькими экстремумами.  
  
В целом, метод Нелдера-Мида является простым и эффективным инструментом для решения различных задач оптимизации и находится в широком использовании в различных областях науки и техники.

# Цель работы:

Изучить метод Нелдера-Мида, научиться применять его на практике посредством реализации на языке программирования C#.

# Теоретический материал

Метод Нелдера-Мида, или метод деформируемого многогранника, применяется для нахождения решения задачи оптимизации вещественных функций многих переменных

f(x) → min, x ∈ ,

причем функция f(x), вообще говоря, не является гладкой и может быть зашумленной. Другой особенностью метода является то, что на каждой итерации вычисляется значение функции f(x) не более чем в трех точках. Это особенно важно в случае сложно вычислимой функции f(x). Метод Нелдера-Мида прост в реализации и полезен на практике, но, с другой стороны, для него не существует теории сходимости — алгоритм может расходиться даже на гладких функциях.

Выпуклым множеством называется такое множество, содержащее вместе с любыми двумя точками соединяющий их отрезок. Выпуклой оболочкой множества называется его минимальное выпуклое надмножество. Симплексом (n-симплексом) называется выпуклая оболочка множества аффинно независимых (n + 1) точек (вершин симплекса).

# Алгоритм

Пусть требуется найти безусловный минимум функции n переменных  𝑓(𝑥(1),𝑥(2),…,𝑥(𝑛)). Предполагается, что серьёзных ограничений на область определения функции нет, то есть функция определена во всех встречающихся точках.

Параметрами метода являются:

* коэффициент отражения 𝛼>0α > 0, обычно выбирается равным 1 1.
* коэффициент сжатия 𝛽>0β > 0, обычно выбирается равным 0,5 0.5.
* коэффициент растяжения 𝛾>0 > 0, обычно выбирается равным 22.

1. «Подготовка». Вначале выбирается 𝑛+1n + 1 точка 𝑥𝑖=(𝑥𝑖(1),𝑥𝑖(2),…,𝑥𝑖(𝑛)),𝑖=1..𝑛+1, образующие симплекс n-мерного пространства. В этих точках вычисляются значения функции: 𝑓1=𝑓(𝑥1),𝑓2=𝑓(𝑥2),…,𝑓𝑛+1=𝑓(𝑥𝑛+1).
2. «Сортировка». Из вершин симплекса выбираем три точки: 𝑥ℎ с наибольшим (из выбранных) значением функции 𝑓ℎ, 𝑥𝑔 со следующим по величине значением 𝑓𝑔 и 𝑥𝑙 с наименьшим значением функции 𝑓𝑙. Целью дальнейших манипуляций будет уменьшение по крайней мере 𝑓ℎ.
3. Найдём центр тяжести всех точек, за исключением 𝑥ℎ: 𝑥𝑐=1𝑛∑𝑖≠ℎ𝑥𝑖. Вычислять 𝑓𝑐=𝑓(𝑥𝑐) не обязательно.
4. «Отражение». Отразим точку 𝑥ℎ относительно 𝑥𝑐 с коэффициентом 𝛼α (при 𝛼=1α = 1 это будет центральная симметрия, в общем случае — гомотетия), получим точку 𝑥𝑟 и вычислим в ней функцию: 𝑓𝑟=𝑓(𝑥𝑟). Координаты новой точки вычисляются по формуле:

𝑥𝑟=(1+𝛼)𝑥𝑐−𝛼𝑥ℎ.

1. Далее смотрим, насколько нам удалось уменьшить функцию, ищем место 𝑓𝑟 в ряду 𝑓ℎ,𝑓𝑔,𝑓𝑙.

Если 𝑓𝑟<𝑓𝑙, то направление выбрано удачное и можно попробовать увеличить шаг. Производим «растяжение». Новая точка 𝑥𝑒=(1−𝛾)𝑥𝑐+𝛾𝑥𝑟 и значение функции 𝑓𝑒=𝑓(𝑥𝑒).

Если  𝑓𝑒<𝑓𝑟, то можно расширить симплекс до этой точки: присваиваем точке 𝑥ℎ значение 𝑥𝑒 и заканчиваем итерацию (на шаг 9).

Если 𝑓𝑒<𝑓𝑟 𝑓𝑟<𝑓𝑒, то переместились слишком далеко: присваиваем точке 𝑥ℎ значение 𝑥𝑟 и заканчиваем итерацию (на шаг 9).

Если  𝑓𝑙<𝑓𝑟<𝑓𝑔, то выбор точки неплохой (новая лучше двух прежних). Присваиваем точке 𝑥ℎ значение 𝑥𝑟 и переходим на шаг 9.

Если 𝑓𝑔<𝑓𝑟<𝑓ℎ , то меняем местами значения 𝑥𝑟 и 𝑥ℎ. Также нужно поменять местами значения 𝑓𝑟 и 𝑓ℎ. После этого идём на шаг 6.

Если 𝑓ℎ<𝑓𝑟, то просто идём на следующий шаг 6.

В результате (возможно, после переобозначения) 𝑓𝑙<𝑓𝑔<𝑓ℎ<𝑓𝑟.

1. «Сжатие». Строим точку 𝑥𝑠=𝛽𝑥ℎ+(1−𝛽)𝑥𝑐  и вычисляем в ней значение 𝑓𝑠=𝑓(𝑥𝑠).
2. Если 𝑓𝑠<𝑓ℎ𝑓𝑒<𝑓𝑟 , то присваиваем точке 𝑥ℎ значение 𝑥𝑠 и идём на шаг 9.
3. Если 𝑓𝑠>𝑓ℎ𝑓𝑒<𝑓𝑟 , то первоначальные точки оказались самыми удачными. Делаем «глобальное сжатие» симплекса — гомотетию к точке с наименьшим значением 𝑥𝑙𝑓𝑒<𝑓𝑟 :

𝑥𝑖←𝑥𝑙+(𝑥𝑖−𝑥𝑙)/2, 𝑖≠𝑙 i≠l.

1. Последний шаг — проверка сходимости. Может выполняться по-разному, например, оценкой дисперсии набора точек. Суть проверки заключается в том, чтобы проверить взаимную близость полученных вершин симплекса, что предполагает и близость их к искомому минимуму. Если требуемая точность ещё не достигнута, можно продолжить итерации с шага 2.

# Идентификация функций, проходящих через метод

Центральный принцип метода состоит в использовании значений функции во различных точках пространства параметров для определения пути к минимуму.

Рассмотрим четыре функции для проверки работоспособности кода:

* 1. Функция Химмельблау
  2. Квадратичная функция
  3. Функция Розенброка
  4. Функция Растригина
  5. Функция элипсоида

## Функция Химмельблау

Функция Химмельблау представляет собой функцию с двумя переменными, которая применяется для оценки качества алгоритмов оптимизации.

Она определяется формулой:

Функция Химмельбау характеризуется наличием четырех эквивалентных локальных минимумов, однако мы будем ориентироваться при выполнении алгоритма на этот локальный минимум: f(3, 2) = 0.

В начальном симплексе берём точки: [0,0; 3,0; 0,3] в результате чего получаем значение функции равной нулю в очень близкой точке (3, 2).

Это получилось в связи с тем, что метод Нелдера – Мида является приближённым.

## Квадратичная функция

Представим её в виде

Ожидаемая точка минимума равна 1, а ожидаемое значение функции в этой точке равно 0.

В начальном симплексе берём точки: [0,0; 3,0; 0,3] в результате чего получаем значение функции равной нулю в очень близкой точке 1, а значение функции в этой точке равно 0.

Ожидаемые и полученные значения совпали.

## Функция Розенброка

Ховард Розенброк предложил свою функцию как инструмент для оценки эффективности различных методов оптимизации. Это нелинейная функция, используемая для измерения производительности алгоритмов оптимизации. Она привлекательна для исследователей в этой области из-за своего характера, вызывающего трудности при нахождении глобального минимума. Данная функция имеет свой локальный минимум равный 0 в точке (1,1).

Функция выглядит следующим образом:

В начальном симплексе берём точки: [-5,-5; 5,0; 0,5] в результате чего получаем значение функции равной нулю в очень близкой точке (1,1), а значение функции в этой точке равно 0.

Ожидаемые и полученные значения совпали.

## Функция Растригина

Оптимизация данной функции представляет собой сложную задачу из-за обширного пространства поиска и множества локальных минимумов. Метод Нелдера-Мида предлагает простое и надежное решение для оптимизации непрерывных функций без ограничений, однако его применимость ограничена в случае наличия множественных локальных минимумов, когда он может столкнуться с проблемами достижения глобального минимума.

Функция Растригина имеет формулу:

Ожидаемая точка минимума равна (5,5), а ожидаемое значение функции в этой точке равно 0.

В начальном симплексе берём точки: [0,0; 1,0; 0,1] в результате чего получаем значение функции равной нулю в очень близкой точке (5,5), а значение функции в этой точке равно 0.

Ожидаемые и полученные значения совпали.

## Функция Эллипсоида

Представим её в виде , где k =10, 100 или 1000

Эту функцию тестируем для проверки тенденции увеличения количества шагов при вычислении, в зависимости от параметра k - множителя при .

**При k = 10:**

Ожидаемая точка минимума равна (0,0) и при этом ожидаемое значение функции в этой точке равно 0.

В начальном симплексе берём точки: [0,0; 1,0; 0,1] в результате чего получаем значение функции равной нулю в очень близкой точке (0,0), а значение функции в этой точке равно 0. Это получили за 91 итераций в коде.

Ожидаемые и полученные значения совпали.

**При k = 100:**

Ожидаемая точка минимума равна (0,0) и при этом ожидаемое значение функции в этой точке равно 0.

В начальном симплексе берём точки: [0,0; 1,0; 0,1] в результате чего получаем значение функции равной нулю в очень близкой точке (0,0), а значение функции в этой точке равно 0. Это получили за 94 итераций в коде.

Ожидаемые и полученные значения совпали.

**При k = 1000:**

Ожидаемая точка минимума равна (0,0) и при этом ожидаемое значение функции в этой точке равно 0.

В начальном симплексе берём точки: [0,0; 1,0; 0,1] в результате чего получаем значение функции равной нулю в очень близкой точке (0,0), а значение функции в этой точке равно 0. Это получили за 95 итераций в коде.

Ожидаемые и полученные значения совпали.

Наблюдается увеличение числа шагов при увеличении k в 10 раз: от k = 10 к k = 100 примерно на 3.19% раз; от k = 100 к k = 1000 примерно на 1.05% раз.

# Вывод

Метод Нелдера-Мида представляет собой важный инструмент в области оптимизации, особенно когда необходимо работать с функциями, не имеющими аналитических производных или когда производные сложно вычислить. Этот метод, также известный как метод деформируемого многогранника, позволяет находить локальные экстремумы функции без использования её градиента. Он основан на принципе последовательного перемещения и деформирования симплекса в пространстве параметров функции, что делает его гибким и применимым к различным типам функций, включая негладкие и зашумленные.

Однако, несмотря на свои преимущества, метод Нелдера-Мида имеет и определенные ограничения. Одним из ключевых недостатков является отсутствие теории сходимости, что означает, что алгоритм может расходиться даже на гладких функциях. Кроме того, хотя метод эффективен при низкой скорости вычисления минимизируемой функции, он может быть неэффективен в современных задачах оптимизации, требующих обработки больших объемов данных или многомерных пространств поиска.

Также стоит отметить, что для достижения лучших результатов в некоторых случаях могут потребоваться модификации метода, например, использование модифицированной версии симплексного поиска, которая учитывает особенности конкретной задачи и может улучшить стабильность и эффективность поиска.

В целом, метод Нелдера-Мида остается полезным инструментом в арсенале специалистов по оптимизации, особенно в тех случаях, когда стандартные градиентные методы не могут быть применены из-за отсутствия необходимых производных или сложности функции.

# Примечание

*Ссылка на проект:* [https://github.com/jazz931/MOzgoshmugi](https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2Fjazz931%2FMOzgoshmugi&cc_key=)

# Источники:

1. <https://ru.wikipedia.org/wiki/Метод_Нелдера_—_Мида>
2. <https://mathprofi.com/uploads/files/1156_f_41_metody-optimizacii-negladkih-funkcii.pdf?key=a94b32268f6014b85c9c48ef842cc5db/>
3. <https://www.sfu.ca/~ssurjano/rosen.html>
4. <https://www.sfu.ca/~ssurjano/rastr.html>
5. <https://en.wikipedia.org/wiki/Himmelblau%27s_function>