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.**

Направление подготовки: «Прикладная информатика»

Профиль подготовки: «Суперкомпьютерное моделирование и инженерный анализ»

**ОТЧЕТ**

по заданию по дисциплине  
**«Методы оптимизации»**  
на тему:

**«Реализация и экспериментальное исследование метода Нелдера–Мида»**

|  |  |
| --- | --- |
|  | **Выполнили:** студенты  группы 3822Б1ПИсм1  Владимиров Иван Алексеевич  Симаева Александра Дмитриевна  Черняева Анна Владиславовна |
|  | **Преподаватель:**  Сморякова Валентина Михайловна |

Нижний Новгород

2025

Оглавление

[Введение 3](#_Toc199628709)

[Математическая модель метода Нелдера–Мида 3](#_Toc199628710)

[Основной алгоритм 3](#_Toc199628711)

[Критерии остановки 4](#_Toc199628712)

[Параметры метода Нелдера–Мида и соответствие переменным в коде 4](#_Toc199628713)

[Описание функций, реализующих метод Нелдера–Мида 5](#_Toc199628714)

[Описание тестовых функций 6](#_Toc199628715)

[Тестирование алгоритма 7](#_Toc199628716)

[Эксперимент №1 7](#_Toc199628717)

[Вывод по результатам 8](#_Toc199628718)

[Эксперимент №2 8](#_Toc199628719)

[Вывод по результатам 9](#_Toc199628720)

[Общий вывод по проделанному тестированию 10](#_Toc199628721)

# Введение

Методы оптимизации находят широкое применение в самых разных областях — от экономики и инженерии до машинного обучения и анализа данных. Во многих практических задачах необходимо найти минимум функции, для которой недоступна производная, либо она слишком сложна для вычисления. В таких случаях применяются методы без использования градиентов, к числу которых относится метод Нелдера–Мида.

Этот метод не требует знания аналитического выражения производной и основан на геометрических преобразованиях, что делает его удобным для оптимизации функций, заданных только в виде численного вычисления. В его основе лежит идея постепенного изменения формы симплекса (геометрической фигуры из n+1 вершины в 𝑛-мерном пространстве) таким образом, чтобы он "сползал" в сторону минимума функции.

В данной работе рассматривается реализация метода Нелдера–Мида на языке C++, а также проводится экспериментальный анализ его поведения при различных значениях параметров алгоритма. Основной целью является оценка устойчивости, точности и скорости сходимости метода при решении задач минимизации на разных тестовых функциях.

# Математическая модель метода Нелдера–Мида

Рассмотрим задачу на минимизацию функции многих переменных:

На вход принимается начальная точка – массив чисел x(0) , откуда начинается поиск.

где ei — единичный вектор вдоль i-й координаты.

## Основной алгоритм

На каждой итерации алгоритм выполняет следующие шаги:

1. Упорядочивание точек симплекса по значениям функции:
2. Вычисление центроида всех точек, кроме наихудшей:
3. Отражение худшей точки:

Если отражённая точка оказалась лучше всех, выполняется расширение:

Если отражённая точка хуже, выполняется сжатие:

Если и это не улучшает результат, применяется редукция всех точек симплекса к лучшей:

## Критерии остановки

Алгоритм завершается, если выполняется условие:

или если превышено число итераций:

# Параметры метода Нелдера–Мида и соответствие переменным в коде

| **Название параметра** | **Обозначение** | **Переменная в коде (NelderMeadParams)** | **Назначение в алгоритме** |
| --- | --- | --- | --- |
| Коэффициент отражения | α | params.alpha | Управляет тем, как далеко отражается худшая точка симплекса через центроид |
| Коэффициент сжатия | β | params.beta | Используется для сближения точки с центроидом, если отражение не дало улучшения |
| Коэффициент расширения | γ | params.gamma | Определяет, насколько далеко продвигаемся, если отражённая точка оказалась удачной |
| Коэффициент редукции | δ | params.delta | Используется при сжатии всего симплекса к лучшей вершине, если остальные шаги не помогли |
| Максимальное количество итераций | — | params.max\_iter | Ограничивает число шагов алгоритма, чтобы избежать бесконечного цикла |
| Точность (допуск) | ε | params.tolerance | Если разница между значениями функции в симплексе меньше этого значения — остановка |
| Начальный размер смещения | h | initial\_step (внутри nelder\_mead\_optimize) | Определяет, как далеко отстоят вершины симплекса от стартовой точки |

# Описание функций, реализующих метод Нелдера–Мида

**1.** create\_default\_params()

Создаёт и возвращает структуру NelderMeadParams с начальными значениями параметров алгоритма: коэффициенты отражения, сжатия, расширения, редукции, допустимая погрешность и максимальное число итераций.

**2.** create\_initial\_simplex()

Создаёт начальный симплекс (набор из n+1 точки) на основе стартовой точки x0. Остальные вершины получаются прибавлением небольшого значения (step\_size = 0.1) к каждой координате по очереди. Также сразу вычисляется значение функции f в каждой вершине.

**3.** compute\_centroid()

Вычисляет центроид (центр масс) всех точек симплекса, кроме худшей. Это используется как база для построения отражённой, расширенной и сжатой точек.

**4.** reflect\_point()

Выполняет отражение худшей точки симплекса относительно центроида. Отражённая точка вычисляется по формуле:

x\_r = c + alpha \* (c - x\_worst)

где c — центроид, x\_worst — худшая точка.

**5.** expand\_point()

Выполняет расширение: если отражённая точка оказалась очень удачной, пробует продвинуться ещё дальше в том же направлении. Формула:

x\_e = c + gamma \* (x\_r - c)

**6.** contract\_point()

Выполняет сжатие симплекса: пробует переместить худшую точку ближе к центроиду, если отражение оказалось неэффективным. Формула:

x\_c = c + beta \* (x\_worst - c)

**7.** shrink\_simplex()

Если все предыдущие шаги не дали улучшения, выполняется полная редукция (сжатие всего симплекса) к лучшей точке. Все вершины (кроме лучшей) перемещаются ближе к ней:

x\_i = x\_best + delta \* (x\_i - x\_best)

**8.** nelder\_mead\_optimize()

Главная функция, которая реализует сам метод оптимизации. Она выполняет:

* построение симплекса;
* сортировку точек по значению функции;
* вычисление центроида;
* вызов отражения, расширения, сжатия и редукции;
* проверку критерия сходимости;
* возврат результата: точка минимума и значение функции в ней.

**9. Структура** Vertex

Вспомогательная структура, содержащая координаты точки (x) и значение функции в этой точке (value). Используется для представления вершин симплекса.

# Описание тестовых функций

Для оценки работы метода Нелдера–Мида были использованы следующие функции:

1. **Функция Розенброка**

* **Минимум:** в точке (1, 1)
* **Назначение:** проверка поведения метода в узкой изогнутой долине. Требует высокой точности и хорошей стратегии поиска.

1. **Функция Химмельблау**

* **Минимумы:** в точках (3.0, 2.0), (-2.805, 3.131), (-3.779, -3.283), (3.584, -1.848)
* **Назначение:** исследование поведения метода в условиях наличия нескольких локальных минимумов. Используется для анализа глобального и локального поиска.

1. **Функция Растригина**

* **Минимум:** в точке (0, 0)
* **Назначение:** тест на устойчивость метода к множеству локальных экстремумов. Является типичной функцией для оценки глобальной сходимости.

1. **Функция Беала**

* **Минимум:** в точке (3, 0.5)
* **Назначение:** демонстрация чувствительности к начальному приближению и возможности сходимости в узком диапазоне.

1. **Функция Экли**

* **Минимум:** в точке (0, 0)
* **Назначение:** проверка глобального поведения метода в условиях плоских областей и синусоидальных провалов.

1. **Функция Голдштейна–Прайса**

* **Минимум:** в точке (0, -1)
* **Назначение:** исследование поведения метода на сложных многоэкстремальных функциях. Высокая нагрузка на устойчивость и сходимость.

# Тестирование алгоритма

## Эксперимент №1

Метод Нелдера–Мида был запущен с фиксированным набором параметров:

* **Коэффициент отражения (α)** = 1.0
* **Коэффициент сжатия (β)** = 0.5
* **Коэффициент расширения (γ)** = 2.0
* **Коэффициент редукции (δ)** = 0.5
* **Максимальное число итераций** = 1000
* **Допуск (ε)** = 1e-6
* **Начальный размер симплекса (шаг)** = 0.1

|  | **Название функции** | **Ожидаемый минимум** | **Начальное приближение** | **Найденный минимум** | **Погрешность** | **Итерации** |
| --- | --- | --- | --- | --- | --- | --- |
|  | Розенброка | (1, 1) | (0.0, 0.0) | (1.00007, 1.00022) | ≈ 0.000001 | 94 |
|  | Химмельблау | (3, 2) | (0.0, 0.0) | (3.00009, 2.00006) | ≈ 0.000000 | 84 |
|  | Растригина | (0, 0) | (0.0, 0.0) | (0.00000, 0.00000) | 0.000000 | 47 |
|  | Беала | (3, 0.5) | (1.0, 1.0) | (3.00055, 0.50006) | ≈ 0.000000 | 68 |
|  | Экли | (0, 0) | (1.0, 1.0) | (0.96840, 0.96843) | ≈ 3.574452 | 44 |
|  | Голдштейн–Прайса | (0, -1) | (0.0, 0.0) | (-0.60001, -0.40001) | ≈ 30.000000 | 71 |

### Вывод по результатам

Метод показал хорошую сходимость на функциях Розенброка, Химмельблау, Растригина и Беала. Во всех этих случаях минимум был найден с высокой точностью при разумном числе итераций.

Для функций Экли и Голдштейн–Прайса результат оказался менее успешным: при выбранных начальных условиях и параметрах алгоритм сошёлся к локальному минимуму, не достигнув глобального. Это объясняется тем, что обе функции имеют сложный рельеф с множеством локальных экстремумов.

## Эксперимент №2

| **Название функции** | **Начальная точка** | **Параметры** | **Ожидаемый минимум** | **Полученный минимум** | **Итерации** |
| --- | --- | --- | --- | --- | --- |
| Экли | (1.00, 1.00) | alpha=1.0, beta=0.3 | (0, 0) | (0.97, 0.97) | 44 |
| Экли | (1.00, 1.00) | alpha=1.0, beta=0.5 | (0, 0) | (0.97, 0.97) | 44 |
| Экли | (1.00, 1.00) | alpha=2.0, beta=0.3 | (0, 0) | (0.97, 0.97) | 46 |
| Экли | (1.00, 1.00) | alpha=2.0, beta=0.5 | (0, 0) | (0.97, 0.97) | 94 |
| Экли | (0.10, 0.10) | alpha=1.0, beta=0.3 | (0, 0) | (0.00, 0.00) | 70 |
| Экли | (0.10, 0.10) | alpha=1.0, beta=0.5 | (0, 0) | (-0.05, 0.05) | 10 |
| Экли | (0.10, 0.10) | alpha=2.0, beta=0.3 | (0, 0) | (-0.00, 0.00) | 108 |
| Экли | (0.10, 0.10) | alpha=2.0, beta=0.5 | (0, 0) | (0.00, -0.00) | 100 |
| Голдштейн-Прайс | (0.00, 0.00) | alpha=1.0, beta=0.3 | (0, -1) | (-0.60, -0.40) | 51 |
| Голдштейн-Прайс | (0.00, 0.00) | alpha=1.0, beta=0.5 | (0, -1) | (-0.60, -0.40) | 71 |
| Голдштейн-Прайс | (0.00, 0.00) | alpha=2.0, beta=0.3 | (0, -1) | (-0.60, -0.40) | 83 |
| Голдштейн-Прайс | (0.00, 0.00) | alpha=2.0, beta=0.5 | (0, -1) | (-0.60, -0.40) | 98 |
| Голдштейн-Прайс | (0.20, -0.80) | alpha=1.0, beta=0.3 | (0, -1) | (0.00, -1.00) | 51 |
| Голдштейн-Прайс | (0.20, -0.80) | alpha=1.0, beta=0.5 | (0, -1) | (-0.00, -1.00) | 59 |
| Голдштейн-Прайс | (0.20, -0.80) | alpha=2.0, beta=0.3 | (0, -1) | (-0.00, -1.00) | 60 |
| Голдштейн-Прайс | (0.20, -0.80) | alpha=2.0, beta=0.5 | (0, -1) | (-0.00, -1.00) | 78 |

### Вывод по результатам

Мы проверили, как метод Нелдера–Мида работает с разными параметрами (alpha, beta) и начальными точками на функциях Экли и Голдштейн–Прайс.

**Главное, что поняли:**

* Если стартовая точка неудачная, метод почти всегда попадает в локальный минимум. Это особенно видно на Экли.
* Когда точка ближе к глобальному минимуму, результат нормальный даже при базовых параметрах.
* alpha = 2.0 и beta = 0.3 в некоторых случаях дали лучший результат, но не всегда — иногда просто больше итераций.
* Начальная точка влияет сильнее, чем параметры.

В ходе исследования мы пробовали менять и параметр γ, но заметили, что это в основном только увеличивает количество итераций и строк в таблице, а на результат сильно не влияет. Поэтому в дальнейшем мы решили не акцентировать внимание на этом параметре.

# Общий вывод по проделанному тестированию

В ходе лабораторной работы мы реализовали метод Нелдера–Мида и проверили его поведение на нескольких классических тестовых функциях. Для начала мы протестировали алгоритм с базовыми параметрами на 6 функциях и увидели, что в большинстве случаев он сходится к правильному минимуму. Однако на некоторых функциях (например, Экли и Голдштейн–Прайс) результат зависел от начальной точки — это показало, что метод может застрять в локальном минимуме.

Чтобы разобраться, от чего это зависит, мы провели дополнительное тестирование: меняли параметры alpha и beta, а также начальные приближения. Выяснилось, что параметры действительно влияют на поведение метода: слишком маленькие значения замедляют сходимость, а слишком большие могут привести к скачкам и нестабильности. Оптимальный результат достигался при умеренных значениях (например, alpha = 1.0–2.0, beta = 0.3–0.5).

Также мы пытались менять параметр gamma, но по результатам поняли, что он влияет не так сильно, и поэтому не стали включать его в финальный анализ. В целом, эксперимент показал, что метод работает стабильно при хорошей настройке параметров и точке старта, но гарантии глобального минимума он не даёт — особенно на сложных многомодальных функциях.

Работа помогла лучше понять, как ведёт себя метод в разных ситуациях и как подбирать параметры, чтобы добиться сходимости.