# Совершенствование численных методов решения многомерных оптимизационных задач

Симановский Марк Анатольевич,
marksimanovskiy@gmail.com

УДК 519.6

**Математика**

## Содержание

**Аннотация**

В данной работе в качестве способов совершенствования методов оптимизации рассматривается параллелизм вычислительных задач, введение гиперкомплексного двумерного кольца над вещественными числами, расширенного нетривиальным делителем нуля, а также использование таблиц дифференцирования. Уравнение Даламбера-Эйлера, разложение аналитической функции в ряд Тейлора, а также некоторые другие утверждения из дифференциального исчисления и линейной алгебры позволяют определить производную вещественной функции как проекцию функции двумерной алгебры на ось базиса нетривиального делителя нуля. Независимость вычисления некоторых вспомогательных методов дает возможность их одновременного параллельного расчета. Рассмотренные модификации испытываются в задачах поиска производной функции и сравниваются с классическими численными и символьными методами. Они сильно опережают последних по скорости, но практически не отличаются от них по точности. На базе рассмотренных улучшений реализуется численный детерминированный метод первого порядка поиска оптимума функции, совершенствуемый параллелизмом вычислительной задачи. Он тестируется на примере реальных оптимизационных задач.

Рассмотренные модификации позволяют усовершенствовать методы дифференцирования и оптимизации. Они могут быть внедрены для практического применения.

**Ключевые слова:** Оптимизация, численные методы, символьные методы, детерминированные методы, многомерная оптимизация, математическое программирование, математическое моделирование, python, теория колец, высшая алгебра, математический анализ, теория функций комплексного переменного.

# 1 Введение

Современное развитие вычислительной техники и интенсивное проникновение математических моделей в различные сферы деятельности человека обеспечивает технический прогресс. В процессе исследования и проектирования моделей обычно ставится задача определения наилучших параметров тех или иных объектов. Задача поиска минимума или максимума целевой функции на заданной области конечномерного пространства, ограниченной набором равенств и/или неравенств, называется задачей оптимизации. Математическое программирование – область математики, которая изучает методы решения задач на оптимизацию с ограничениями. При решении задач оптимизации численными методами возникает ряд трудностей, таких как ошибки вычислительной процедуры. Нередко в процессе вычисления возникают большие погрешности, что снижает эффективность оптимизации. Наиболее часто доставляют неприятности ошибки при округлении, а также при аппроксимации производных разностными выражениями. При программировании компьютерных алгоритмов для минимизации ошибок обычно используются символьные вычисления вместо численных методов, но они слишком медленны, чтобы использовать их, например, для решения задач в реальном времени[7].

# 2 Цель работы

Цель данной работы заключается в исследовании численных методов решения многомерных оптимизационных задач и разработке способов улучшения рассмотренных методов как по точности, так и по производительности.

# 3 Теоретическая часть

## 3.1 О связи производных и оптимумов

### 3.1.1 Производная

**Определение 1.** Пусть функция $y = f(x)$ определена на множестве $M$ и существует точка $x_0 : x_0 \in M$. Изменим $x_0$ на такую величину $\Delta x$, не зависящую от $x_0$, что $x_0 + \Delta x \in M$. Тогда изменение функции $\Delta y = f(x + \Delta x) - f(x)$. Производной функции $f(x)$ в точке $x_0$ назовем передел отношения изменения функции к изменению его независимого аргумента $x$, при стремлении изменения к нулю (1).

$$f'_x(x) = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{1}$$

Для обозначения производных принято употреблять несколько различных нотаций[1] (Таблица 1). В данной работе будет использоваться, в основном, нотация Коши.

Таблица 1: Нотации производных

| | |
|---|---|
| $\frac{df}{dx}$ | Нотация Лейбница |
| $f'_x$ | Нотация Лагранжа |
| $D_x f$ | Нотация Коши |

Однако не для каждой точки из области определения функции может существовать производная. Необходимым и достаточным условием ее существования является равенство пределов изменений слева, справа и в точке при существовании каждого предела(2). Условие существования производной называют дифференцируемостью, а процесс нахождения производной - дифференцированием.

$$\exists D_x f(x_0) \Leftrightarrow \lim_{\Delta x \to 0_-} \frac{\Delta y(x_0)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta y(x_0)}{\Delta x} = \lim_{\Delta x \to 0_+} \frac{\Delta y(x_0)}{\Delta x} \tag{2}$$

**Определение 2.** Назовем функцию элементарной, если ее можно получить с помощью **конечного** числа арифметических операций и композиций основных (также элементарных) функций.

Основными считаются функции, приведенные в Таблице 2.

Таблица 2: Основные функции

| | |
|---|---|
| Степенная функция | $x^n$ |
| Показательная и логарифмическая функции | $a^x,\ lnx$ |
| Тригонометрические и обратные им функции | $sinx, cosx, arcsinx...$ |

Из тригонометрических функций можно рассматривать лишь синус и обратный ему арксинус, а все остальные – выражать через него $\left(tgx = \frac{sinx}{sin(\frac{\pi}{2} - x)}\right)$. Более того, из следствий основного логарифмического тождества и тождества Эйлера все элементарные функции представимы через комбинации экспонент и натуральный логарифм

$\left(a^b \equiv e^{b\ln(a)},\quad sin(x) \equiv \dfrac{e^{ix} - e^{-ix}}{2i}\right)$, но для удобства косинус также включается в эту таблицу.

Любая элементарная функция непрерывна и дифференцируема на своей области определения [2]. Зная некоторые арифметические правила, связанные с производной и производные основных функций, можно найти производную любой элементарной функции, не прибегая к определению производной (1) через предел [2]. Перечислим их в Таблице 3.

Таблица 3: Правила дифференцирования

| Название | $f(x)$ | $D_x f(x)$ |
|---|---|---|
| Производная константы | $C$ | $0$ |
| Производная умножения на константу | $C \cdot g(x)$ | $C \cdot (D_x g(x))$ |
| Производная суммы | $u(x) + v(x)$ | $D_x u(x) + D_x v(x)$ |
| Производная произведения функций | $u(x) \cdot v(x)$ | $(D_x u(x)) \cdot v(x) + (D_x v(x)) \cdot u(x)$ |
| Производная композиции функций | $u(v(x))$ | $D_v u(v) \cdot D_x v(x)$ |
| Производная обратной функции | $g(x)$ | $\dfrac{1}{D_g x(g)}$ |
| Производная степенной функции | $x^n$ | $nx^{n-1}$ |
| Производная показательной функции | $a^x$ | $a^x \cdot ln(a)$ |
| Производная логарифмической функции | $log_a(x)$ | $\dfrac{1}{x \cdot ln(a)}$ |
| Производная синуса | $sin(x)$ | $cos(x)$ |
| Производная косинуса | $cos(x)$ | $-sin(x)$ |
| Производная арксинуса | $arcsin(x)$ | $\dfrac{1}{\sqrt{1-x^2}}$ |

Рассмотрим основную теорему дифференциального исчисления, связанную с оптимизацией.

**Теорема 1.** *(Теорема Ферма).* Пусть функция $f(x)$ на промежутке $(a; b)$ принимает наибольшее (наименьшее) значение в точке $x_0 \in (a; b)$. Тогда $D_x f(x_0) = 0$.

***Доказательство.*** Предположим, что $D_x f(x_0) \neq 0$. Если функция принимает наибольшее значение в точке $x_0$ то, справедливо, что $\forall x_1 \in (x_0 - \delta; x_0 + \delta)\ f(x_0) > f(x_1)$. Из (1) следует, что $D_x f(x_0) = \lim_{x - x_0 \to 0} \dfrac{f(x) - f(x_0)}{x - x_0}$. Если $D_x f(x_0) > 0$, то $f(x_1) - f(x_0) > 0, x_1 > x_0$, откуда следует, что $f(x_0) < f(x_1)$, что противоречит условию наибольшего значения. Аналогично и с другим знаком.

**Теорема 2.** *(Теорема Ролля).* Пусть функция $f(x)$ на промежутке $[a; b]$ непрерывна и дифференцируема, а $f(a) = f(b)$. Тогда существует такая точка $c \in (a; b)$, что $D_x f(c) = 0$.

***Доказательство.*** По второй теореме Вейерштрасса, функция достигает своего наибольшего и наименьшего значения на отрезке. Пусть $f_{\text{наиб}} = M$, а $f_{\text{наим}} = m$. Если $m = M$, то функция стационарна на всем отрезке, значит производная в любой точке отрезка равна нулю. Тогда рассмотрим $M > m$ ($m > M$ не возможно по определению). Пусть $f(x_0) \in \{m, M\}, x_0 \in (a; b)$. Значит по теореме Ферма $D_x f(x_0) = 0$.

**Теорема 3.** *(Теорема Лагранжа).* Пусть функция $f(x)$ непрерывна и дифференцируема на отрезке $[a;b]$. Тогда существует такая точка $c \in (a;b)$, что $D_x f(c) = \dfrac{f(b)-f(a)}{b-a}$.

***Доказательство.*** Пусть $g(x) = f(x) - f(a) - \dfrac{f(b)-f(a)}{b-a}(x-a)$. Тогда $g(a) = g(b) = 0$, функция дифференцируема и принимает одинаковое значение на концах отрезка, значит по теореме Ролля существует точка $c \in (a;b)$, что $D_x g(c) = 0$. $D_x g(x) = D_x(f(x) - f(a) - \dfrac{f(b)-f(a)}{b-a}(x-a)) = D_x f(x) - \dfrac{f(b)-f(a)}{b-a}$, следовательно, верно (3).

$$D_x f(c) = \frac{f(b)-f(a)}{b-a} \tag{3}$$

Формулу (3) также называют формулой конечных приращений и используют для численных вычислений производных, устремляя разницу между $b$ и $a$.

Так как производная элементарной функции является элементарной функцией[1], то она тоже может иметь производную. Производная от производной называется производной второго порядка. $D_x(D_x f(x)) = D_{x^2} f(x)$. До сих пор рассматривалась унарная функция $y : \mathbb{D}(y) \to \mathbb{E}(y)$, но очень часто на практике зависимая переменная может быть связана не с одной, а с несколькими переменными[3]. Такая функция $y = f(x_1, x_2, ..., x_n)$ называется функцией многих переменных. Для них работают теоремы дифференциального исчисления функций одной переменной, но добавляются новые понятия, такие как частные приращения.

**Определение 3.** Частным приращением по независимой переменной $x_i$ функции многих переменных $\varphi(x_1, x_2, ..., x_i, ..., x_n)$ в точке $(x_{1_0})$ назовем изменение функции при изменении переменной $x_i$ $\Delta\varphi = \varphi(x_{1_0}, x_{2_0}, ..., x_{i_0} + \Delta x_i, ..., x_{n_0}) - \varphi(x_{1_0}, x_{2_0}, ..., x_{i_0}, ..., x_{n_0})$.

Исходя из этого определим частную производную как (4).

$$D_{x_i} f(x_{1_0}, ..., x_{i_0}, ..., x_{n_0}) = \lim_{\Delta x_i \to 0} \frac{f(x_{1_0}, x_{2_0}, ..., x_{i_0} + \Delta x_i, ..., x_{n_0}) - \varphi(x_{1_0}, x_{2_0}, ..., x_{i_0}, ..., x_{n_0})}{\Delta x_i} \tag{4}$$

**Определение 4.** Градиентом функции $f(x_1, x_2, ..., x_n)$ назовём вектор размерностью $n$, состоящий из частных производных по соответствующим переменным.

$$\nabla f(x_1, x_2, ..., x_n) = (D_{x_1} f(x_1, x_2, ..., x_n), D_{x_2} f(x_1, x_2, ..., x_n), ..., D_{x_n} f(x_1, x_2, ..., x_n))$$

Следует учитывать, что $x_j$ относительно $x_i$ считается постоянной, так как не зависит от нее, потому и $D_{x_i}(x_j) = 0$ (для $1 \le i, j \le n, i \ne j$). Также возможно брать производную функции по разным переменным. $D_{xy} f(x,y) = D_x(D_y f(x,y))$. Для непрерывных элементарных функций смешанные производные равны: $D_{xy} f = D_{yx} f$ [3].

### 3.1.2 Оптимум и алгоритмы его нахождения

**Определение 5.** Оптимизацией называется процесс поиска экстремального значения заданной функции на данном множестве. Оптимум - набор независимых переменных, значение функции от которых экстремально.

Существует достаточно много методов поиска оптимумов. Классифицируется методы по случайности результата (детерминированные и стохастические), по способу получения результата (аналитические и численные), по необходимости дифференцировать

[1] Считаем, $f(x) = 0$ тоже элементарная функция

заданную функцию (прямые - производные не используются; методы первого порядка - используется производная первого порядка, градиент; методы второго порядка - используются производные второго порядка, гессиан). В данной работе рассматриваются детерминированные численные прямые методы и методы первого порядка.

Будем считать, что необходимо оптимизировать функцию $F(\overrightarrow{x})$ на множестве $\mathbb{M}^n$, где $\overrightarrow{x}$ - вектор ее независимых переменных $(x_1, x_2, ..., x_n) \in \mathbb{M}^n$. Тогда пусть задача нахождения оптимума записывается одним из следующих вариантов:

$$\overrightarrow{x_0} = argmin_{x \in \mathbb{M}^n} F(\overrightarrow{x}) - ?$$

$$\overrightarrow{x_0} : F(\overrightarrow{x_0}) \overset{\mathbb{M}^n}{\longmapsto} min = ?$$

**Градиентный метод**

Рассмотрим алгоритм любого градиентного метода.

1. Выберем случайную точку из множества и подведем к ней радиус-вектор $\overrightarrow{x^0} \in \mathbb{M}^n$, точность вычисления $\varepsilon$ и множитель шага $\lambda$. Определим $i = 1$.
2. Вычислим $\overrightarrow{x^i} = \overrightarrow{x^{i-1}} - \lambda \nabla F(\overrightarrow{x^{i-1}})$.
3. Если $||\overrightarrow{x^i} - \overrightarrow{x^{i-1}}|| < \varepsilon$, то $\overrightarrow{x^i}$ - оптимум, иначе необходимо увеличить $i$ на единицу и вернутся к п.2.

Представив идеальную точность ($\varepsilon = 0$) получаем Теорему Ферма: производная функции в точке, где функция принимает экстремальное значение, равна нулю. Но так как иррациональное число возможно представить на ЭВМ как его рациональное приближение, при работе с числом накапливается ошибка и падает точность - для этого условием остановки алгоритма поиска является неравенство.

В то время как выбор начальной точки на сегодняшний день может улучшаться стохастическими методами, алгоритм градиентного спуска также может быть улучшен правильным выбором множителя шага. Когда выбирается наилучший шаг на каждой итерации, градиентный алгоритм называется методом наискорейшего спуска [4]. Для его поиска на каждом шаге необходима следующая одномерная оптимизация: $\lambda^i = argmin_{\mathbb{R}_{0+}} F(\overrightarrow{x^{i-1}} - \lambda^{i-1} \nabla F(\overrightarrow{x^{i-1}}))$. Проводить ее можно как прямым способом, например, методом дихотомии, так и методом прямого порядка.

# 4 Основная часть

## 4.1 Математическая модель

**Утверждение 1.** Пусть $\Phi < \mathbb{R}^2, +, \times >$ - двумерная алгебра над полем действительных чисел, для которой верны следующие утверждения:

$$\exists! \theta \in \Phi : \forall a \in \Phi \; a + \theta = \theta + a = a \tag{5}$$

$$\forall a \in \Phi \; \exists! b \in \Phi : a + b = b + a = \theta \tag{6}$$

$$\exists! \epsilon \in \Phi : \forall a \in \Phi \; a \times \epsilon = \epsilon \times a = a \tag{7}$$

$$\forall a, b \in \Phi \; a + b = b + a = c : c \in \Phi \tag{8}$$

$$\forall a, b, c \in \Phi \; (a + b) + c = a + (b + c) \tag{9}$$

$$\forall a, b \in \Phi \; a \times b = b \times a = c : c \in \Phi \tag{10}$$

$$\forall a, b, c \in \Phi a \times (b + c) = a \times b + a \times c \tag{11}$$

$$\exists \delta \in \Phi : \delta \neq \theta \wedge \delta \times \delta = \theta \tag{12}$$

$$\forall a \in \Phi : a = x + \delta y, \; x, y \in \mathbb{R} \tag{13}$$

$$\forall a = x + \delta y \in \Phi, \lambda \in \mathbb{R} \; \lambda a = \lambda x + \delta \lambda y \tag{14}$$

$$\forall n = a + \delta b, m = c + \delta d \in \Phi \; n + m = (a + c) + \delta(b + d) \tag{15}$$

$$\forall n = a + \delta b, m = c + \delta d \in \Phi \; n \times m = (a \cdot c) + \delta(a \cdot d + c \cdot b) \tag{16}$$

$$\theta = 0 + \delta \cdot 0, \epsilon = 1 + \delta \cdot 0 \tag{17}$$

Тогда $\Phi$ является кольцом над полем вещественных чисел с тремя особыми элементами: $\theta$ нейтрален по сложению, $\epsilon$ нейтрален по умножению, $\delta$ является гиперкомплексной единицей - нетривиальным делителем нейтрального по сложению элемента. Каждый элемент кольца - гиперкомплексное число, представимое алгебраически как разложенное по двум базисам, что равнозначно паре (вектору размерности 2): $a \in \Phi \equiv x + \delta y \equiv \begin{pmatrix} x \\ y \end{pmatrix}, \; x, y \in \mathbb{R}$. Рассмотрим уравнение Даламбера-Эйлера для комплексного переменного $z = x + \delta y$ (18). Тогда из его следствия[5] о производной функции комплексной переменной верно (21)

$$\omega(z) = u(x, y) + \delta v(x, y) \tag{18}$$

$$\nabla = \frac{\partial}{\partial x} + \delta \frac{\partial}{\partial y} \tag{19}$$

$$\nabla \omega = \left( \frac{\partial}{\partial x} + \delta \frac{\partial}{\partial y} \right) \cdot (u(x, y) + \delta v(x, y)) = D_x u(x, y) + \delta \left( D_x v(x, y) + D_y u(x, y) \right) \tag{20}$$

$$D_z \omega = \nabla \omega = D_x u(x, y) + \delta D_y u(x, y) \tag{21}$$

Пусть функция $\varphi(x), x \in \mathbb{R}$ непрерывна и дифференцируема в точке $a$ и ее окрестности. Рассмотрим ее разложение в ряд Тейлора (22). Заменим действительный аргумент $x$ на гиперкомплексный $z = x + \delta$. Воспользуемся биномом Ньютона, чтобы раскрыть сумму в степени $k$ (24). В силу утверждений построения алгебры $\Phi$ ($\lambda \delta^n = 0, \forall n >= 2, n.\lambda \in \mathbb{R}$) верно (25). Тогда из (21), (22) и (25) следует (26). Раскроем сумму и заметим сходства с (22). Таким образом сумма сумм является суммой функции и ее производной,

умноженной на гиперкомплексную единицу (27).

$$\varphi(x) = \sum_{k=0}^{\infty} D_{x^k}\varphi(a)\frac{(x-a)^k}{k!} \tag{22}$$

$$\varphi(z) = \sum_{k=0}^{\infty} D_{z^k}\varphi(z)\frac{((x-a)+\delta)^k}{k!} \tag{23}$$

$$((x-a)+\delta)^k = \sum_{i=0}^{k} C_k^i (x-a)^{k-i}(\delta)^i = (x-a)^k + k(x-a)^{k-1}\delta + C_k^2(x-a)^{k-2}(\delta)^2 + ... \tag{24}$$

$$((x-a)+\delta)^k = (x-a)^k + k(x-a)^{k-1}\delta \tag{25}$$

$$\varphi(z) = \sum_{k=0}^{\infty} \frac{1}{k!} D_{z^k}\varphi(z)\left((x-a)^k + k(x-a)^{k-1}\delta\right) = \tag{26}$$

$$= \sum_{k=0}^{\infty} D_{z^k}\varphi(z)\frac{(x-a)^k}{k!} + \delta\sum_{k=1}^{\infty} D_{z^k}\varphi(z)\frac{(x-a)^{k-1}}{(k-1)!}$$

$$\varphi(z) = \varphi(x+\delta) = \varphi(x) + \delta D_x\varphi(x) \tag{27}$$

Вычтем из (27) значение функции в точке $a$ и поделим на $\delta$. Получается уравнение, схожее с (1):

$$D_x\varphi(x) = \frac{\varphi(x+\delta)-\varphi(x)}{\delta} \equiv \lim_{\Delta x\to 0}\frac{\varphi(x+\Delta x)-\varphi(x)}{\Delta x}$$

«Совпадение» интуитивно объяснимо тем, что $\Delta x$ при стремлении к нулю и так мало, а в квадрате уже настолько пренебрежимо, что $(\Delta x)^2 \approx 0$. Однако совпадения нет, так как нет и уравнения, где $\delta$ стоит в знаменателе. Так как $\Phi$ - кольцо, то обратного элемента для делителя нуля не определено. Возможность умножения на элемент, обратный делителю нуля (деление на него) существует в колесах[6]. Расширение кольца до колеса лежит в перспективе развития данной работы. Но решением данной проблемы является равносильность записей гиперкомплексного числа в алгебраическом и векторном виде (28):

$$f(x+\delta) = f(x) + \delta D_x f(x) \equiv f\left(\begin{pmatrix} x \\ 1 \end{pmatrix}\right) = \begin{pmatrix} f(x) \\ D_x f(x) \end{pmatrix} \tag{28}$$

при следующем нижеследующем утверждении.

**Утверждение 2.** Функция от вектора есть вектор со следующей функцией

$$f\left(\begin{pmatrix} x \\ k \end{pmatrix}\right) = \begin{pmatrix} f(x) \\ \alpha(f(x),k) \end{pmatrix}$$

где $\alpha$ - некоторое дифференциальное отображение.

Тогда производную в точке можно определить из выбора второй координаты вектора-значения функции от вектора-переменной из точки и гиперкомплексной единицы. Выбор можно осуществить математически, как скалярное произведение вектора-значения функции с вектором $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ или умножением матрицы $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ на вектор-значение функции.

Очевидно, что использовать скалярное произведение выгодней: сложность вычисления результата произведения квадратных матриц размерностью $n$ составляет $O(n^3)$ по

определению или $O(n^{\log_2 7})$ по алгоритму Штрассена и требует $O(n^2)$ памяти, в то время как скалярное произведение двух векторов требует $O(n)$ времени, а это операция, выполняемая для каждой ячейки матрицы (всего $n^2$ раз), и требует лишь $O(n)$ памяти.

Тогда производная функции с помощью введения кольца $\Phi$ определяется по уравнению

$$D_x f(x) = \left\langle f\left(\begin{pmatrix} x \\ 1 \end{pmatrix}\right), \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\rangle \tag{29}$$

Так как скалярное произведение двух векторов представимо, как $\langle \overrightarrow{a}, \overrightarrow{b} \rangle = |a| \cdot |b| \cdot \angle\cos(\overrightarrow{a}, \overrightarrow{b})$, а длина проекции вектора $\overrightarrow{b}$ на вектор $\overrightarrow{a}$ есть $pr_b a = |a| \angle\cos(\overrightarrow{a}, \overrightarrow{b}) = \frac{\langle \overrightarrow{a}, \overrightarrow{b} \rangle}{|a| \cdot |b|} \cdot |a| = \frac{\langle \overrightarrow{a}, \overrightarrow{b} \rangle}{|b|}$, то геометрический смысл производной во веденной алгебре - проекция функции на мнимую ось.

Рассмотрим в трехмерном пространстве $XYZ$ множество всех таких точек $P = \begin{pmatrix} t \\ f(t) \\ D_t f(t) \end{pmatrix} \in \mathbb{R} \times \Phi$ для любого $t \in \mathbb{R}$ и их проекции. Для примера положим $f(x) = x^2 sin(x)$. График множества точек $P$ - (рис. 1). Заметим, что множество точек $\begin{pmatrix} t \\ f(t) \end{pmatrix}$ $\forall t \in \mathbb{R}$ - параметрическая запись функции $y = f(x)$, тогда проекция всех точек P на плоскость $XY$ - все точки функции $f(x)$ (рис. 2).

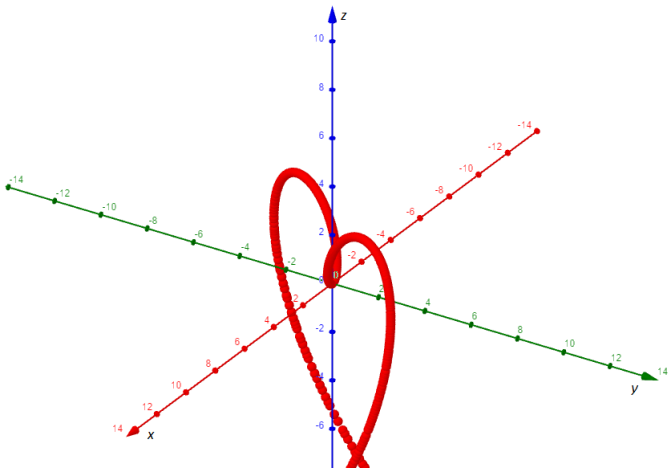


Рис. 1: Множество точек P. Функция в пространстве $\mathbb{R} \times \Phi$.

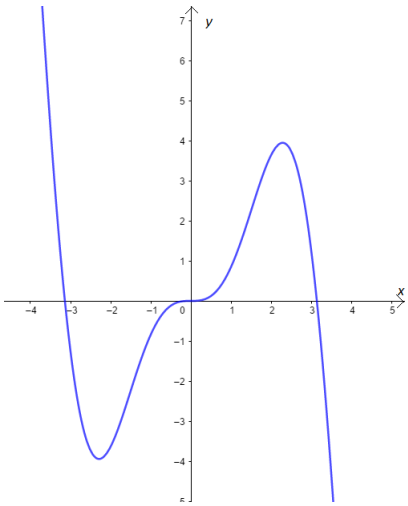


Рис. 2: Множество проекций точек P на плоскость XY. Функция в поле $\mathbb{R}$.

Аналогично $\begin{pmatrix} t \\ D_t f(t) \end{pmatrix}$ $\forall t \in \mathbb{R}$ - параметрическая запись функции $y = D_x f(x)$, тогда проекция всех точек P на плоскость $XZ$ - все точки производной функции $f(x)$ (Рис. 3).

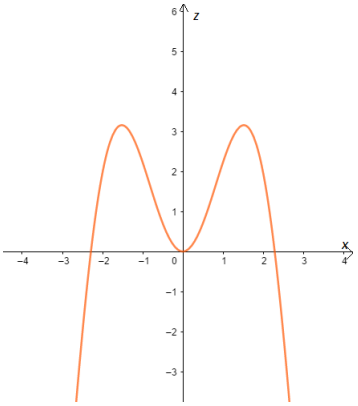


Рис. 3: Множество проекций точек P на плоскость XZ. Производная функции в поле $\mathbb{R}$.

На следующем рисунке (рис. 4) изображены проекции всех точек P на плоскость YZ. Каждая координата задается лишь значением функции для некого аргумента и значением производной от того же аргумента. Учитывая принятые утверждением 2 обозначения и аналогию с предыдущими примерами, множество таких точек - это диаграмма функции в кольце $\Phi$. Следует отметить, что ее нельзя назвать графиком функции, так как полученное множество рассчитывается для всех значений аргумента, ось которого перпендикулярна плоскости проекции, а значит, теряется условие одно-многозначного соответствия.

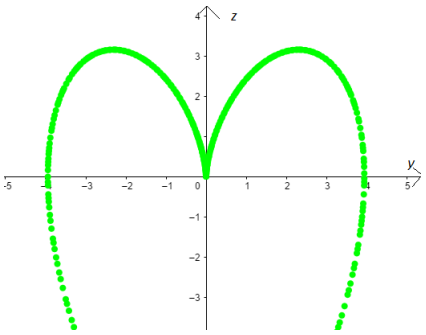


Рис. 4: Множество проекций точек P на плоскость YZ. Функция в кольце $\Phi$.

Тогда в одном пространстве все точки и их проекции выглядят как на рис. 5.

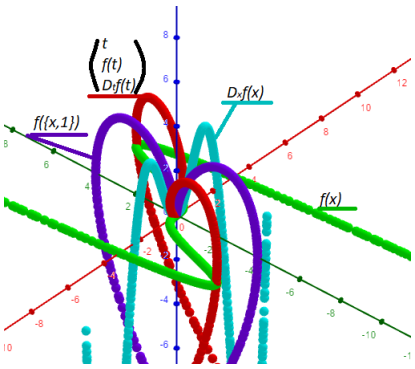


Рис. 5: Множество точек P и их проекций.

Теперь рассмотрим многомерный случай. Пусть задана функция $f(x, y, z)$, требуется найти ее градиент. Из «совпадения», полученного из (27), следует, что приращивается та переменная, вторая координата вектора которой равна 1. Назовем для удобства $\overrightarrow{snd} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Тогда выражение для градиента выглядит, как

$$\begin{aligned} \nabla f(x,y,z) = (&\left\langle f\left(\begin{pmatrix} x \\ 1 \end{pmatrix}, \begin{pmatrix} y \\ 0 \end{pmatrix}, \begin{pmatrix} z \\ 0 \end{pmatrix}\right), \overrightarrow{snd} \right\rangle, \\ &\left\langle f\left(\begin{pmatrix} x \\ 0 \end{pmatrix}, \begin{pmatrix} y \\ 1 \end{pmatrix}, \begin{pmatrix} z \\ 0 \end{pmatrix}\right), \overrightarrow{snd} \right\rangle, \\ &\left\langle f\left(\begin{pmatrix} x \\ 0 \end{pmatrix}, \begin{pmatrix} y \\ 0 \end{pmatrix}, \begin{pmatrix} z \\ 1 \end{pmatrix}\right), \overrightarrow{snd} \right\rangle) \end{aligned} \tag{30}$$

**Пример 1.** $f(x) = x^2 - 2$, $D_x f$ - ?

$D_x f = \langle f(x+\delta), \overrightarrow{snd} \rangle = \langle (x+\delta)^2 - 2, \overrightarrow{snd} \rangle = \langle x^2 + 2x\delta + \delta^2 - 2, \overrightarrow{snd} \rangle = \langle x^2 + 2x\delta - 2, \overrightarrow{snd} \rangle = \left\langle \begin{pmatrix} x^2 - 2 \\ 2x \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\rangle = 2x$

$D_x f = 2x$

**Пример 2.** $f(x, y) = x^2 - y^3$, $\nabla f$ - ?

$D_x f(x, y) = \langle f(x + \delta, y), \overrightarrow{snd} \rangle = \langle (x + \delta)^2 - y^3, \overrightarrow{snd} \rangle = \langle x^2 + 2x\delta + \delta^2 - y^3, \overrightarrow{snd} \rangle = \langle (x^2 - y^3) + (2x)\delta, \overrightarrow{snd} \rangle = \left\langle \begin{pmatrix} x^2 - y^3 \\ 2x \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\rangle = 2x$

$D_y f(x,y) = \langle f(x, y+\delta), \overrightarrow{snd} \rangle = \langle x^2 - (y+\delta)^3, \overrightarrow{snd} \rangle = \langle x^2 - y^3 - 3y\delta^2 - 3y^2\delta + \delta^3, \overrightarrow{snd} \rangle =$
$\langle (x^2 - y^3) + (3y^2)\delta, \overrightarrow{snd} \rangle = \left\langle \begin{pmatrix} x^2 - y^3 \\ 3y^2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\rangle = 3y^2$

$\nabla f = (2x, 3y^2)$

Рассмотрим дифференцирование композиции двух функций с помощью введенного кольца. Пусть $y = f(g(x))$, тогда из Таблицы 3: $D_x y = D_g f \cdot D_x g$. Из (28) следует, что $\alpha(f(x), 1) = D_x f(x)$, пусть тогда $\alpha(f, g) = Df \cdot g$. Заметим, что $(f(g(x+\delta))) = f\left(g\left(\begin{pmatrix} x \\ 1 \end{pmatrix}\right)\right) = f\left(\begin{pmatrix} g(x) \\ D_x g(x) \end{pmatrix}\right) = \begin{pmatrix} f(g(x)) \\ D_g f(g) \cdot D_x g(x) \end{pmatrix} = \begin{pmatrix} f(g(x)) \\ D_x f(g(x)) \end{pmatrix} = \begin{pmatrix} y \\ D_x y \end{pmatrix}$. Тогда для вычисления производной композиции двух функций нет необходимости вычислять отдельно производные этих функций и умножать их, достаточно лишь «правильно» применять функцию к вектору. Производная композиции выражается уравнением (31).

$$D_x f(g(x)) = \left\langle f(g(x+\delta)), \overrightarrow{snd} \right\rangle \tag{31}$$

Отсюда следует еще одно важное свойство. До этого мы вычисляли производную практически по определению - как разность функции от переменной с приращением и без. Воспользуемся определением элементарной функции. Тогда будем находить производную основной функции по правилу таблицы дифференцирования, а элементарной неосновной - по правилу композиции. Попробуем найти по определению производную синуса: $sin(x+\delta) - sin(x) = sin(x)cos(\delta) + sin(\delta)cos(x) - sin(x) = sin(x)(cos(\delta) - 1) + sin(\delta)cos(x)$. Ничего определенного. Применим тождество Эйлера: $sin(x+\delta) - sin(x) = \frac{e^{i(x+\delta)} - e^{-i(x+\delta)} - e^{ix} + e^{-ix}}{2i} = \frac{e^{ix}e^{i\delta} - e^{-ix}e^{-i\delta} - e^{ix} + e^{-ix}}{2i} = \frac{e^{ix}(e^{i\delta} - 1) - e^{ix}(e^{-i\delta} - 1)}{2i}$.
Учитывая, что экспонента по формуле Тейлора раскрывается как $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + ...$, а $\delta^2 = 0$, подставим: $e^{i\delta} = 1 + i\delta - \frac{\delta^2}{2} + \frac{-i\delta^3}{6} + ... = 1 + i\delta$. Тогда $sin(x+\delta) - sin(x) = \frac{e^{ix}(e^{i\delta} - 1) - e^{ix}(e^{-i\delta} - 1)}{2i} = \frac{e^{ix}(1 + i\delta - 1) - e^{ix}(1 - i\delta - 1)}{2i} = \delta\frac{e^{ix}i + e^{-ix}i}{2i} = \delta\frac{e^{ix} + e^{-ix}}{2} = cos(x)\delta$. Получилось довольно громоздко. А теперь вычислим с использованием предопределенных таблиц:

$$D_x sin(x) = \left\langle sin\left(\begin{pmatrix} x \\ 1 \end{pmatrix}\right), \overrightarrow{snd} \right\rangle = \left\langle \begin{pmatrix} sin(x) \\ \alpha(sin(x), 1) \end{pmatrix}, \overrightarrow{snd} \right\rangle = \left\langle \begin{pmatrix} sin(x) \\ cos(x) \end{pmatrix}, \overrightarrow{snd} \right\rangle = cos(x)$$

Тот же результат, но при использовании таблиц, мы получили за меньшее количество действий, от чего при использовании рациональных приближений ответ, очевидно, получается с меньшей погрешностью.

**Пример 3.** $u(v) = \sin v, v(x) = x^3$, $D_x u$ - ?

$D_x u = \left\langle sin\left(\begin{pmatrix} x \\ 1 \end{pmatrix}^3\right), \overrightarrow{snd} \right\rangle = \left\langle sin\left(\begin{pmatrix} x^3 \\ \alpha(x^3, 1) \end{pmatrix}\right), \overrightarrow{snd} \right\rangle = \left\langle sin\left(\begin{pmatrix} x^3 \\ 3x^2 \end{pmatrix}\right), \overrightarrow{snd} \right\rangle =$
$\left\langle \begin{pmatrix} sin(x^3) \\ \alpha(sin(x^3), 3x^2) \end{pmatrix}, \overrightarrow{snd} \right\rangle = \left\langle \begin{pmatrix} sin(x^3) \\ cos(x^3) \cdot 3x^2 \end{pmatrix}, \overrightarrow{snd} \right\rangle = 3x^2 cos(x^3)$

## 4.2 Программная реализация модели

### 4.2.1 Язык разработки

Языком реализации выбран python постольку поскольку он поддерживает замыкания, длинную арифметику и комплексные числа «из коробки», а также для него есть

удобные инструменты тестирования (python magic) и построения графиков (matplotlib). Кроме того существует библиотека символьных вычислений SymPy, с которой удобно будет сравнивать результат.

### 4.2.2 Установка ПО

Для установки интерпретатора языка python и интерактивной оболочки Jupyter достаточно выполнить следующую команду в консоли на *nix системе с пакетным менеджером APT

```
1       sudo apt install python3
2       sudo apt install ipython3 ipython3-qtconsole
```

После установим необходимые библиотеки: NumPy для работы с линейными пространствами, matplotlib для построения графиков, SymPy для символьных вычислений

```
1       pip3 install numpy
2       pip3 install matplotlib
3       pip3 install sympy
```

### 4.2.3 Представление числа

Описанная алгебра $\Phi$ может быть определена не только над полем вещественных чисел. Возможна практически любая алгебраическая структура, которую возможно описать императивно. Потому создадим класс Numer, описывающий базовое число, над которым строится алгебра. В данном случае алгебру строим над классом float - одномерными рациональными числами с плавающей запятой.

Так как $\Phi < \mathbb{F}^2, +, \times >$ - двумерная алгебра, то каждое число задается несколькими значениями. Для универсальности опишем вектор, он также понадобится для работы с многомерными функциями. Определим сложение векторов и их скалярное произведение. Теперь опишем само кольцо в классе HCRA(hyper complex ring algebra) на основе утверждений (5 – 17). В классе DCalculus реализуем таблицу производных, а также публичные методы с замыканиями дифференцирования функции: d(f)(x) и grad(f)(x) с обработкой аргументов в функциональном стиле.

```
1       def d(f):
2         def f_(x):
3           return f(DCalculus(x,1)).I()
4         return f_
5
6       def full_derivative(f, args):
7         return f(*map(lambda p: DCalculus(p,1), args)).I()
8       def partial_derivative(f, args, i):
9         return f(*map(lambda j: DCalculus(j[1], int(j[0] == i)),
              enumerate(args))).I()
10      def grad(f):
11        def grad_(args):
12          return tuple(f(*map(lambda j: DCalculus(j[1], int(j[0]
                == i)), enumerate(args))).I() for i,p in enumerate(
                args))
13        return grad_
```

### 4.2.4 Реализация методов оптимизации

Реализуем метод наискорейшего градиентного спуска. Будем оптимизировать $\lambda$ одновременно двумя способами: прямой дихотомией и найдя нуль производной функции (с использованием теоремы Ферма). В алгоритме Ньютона-Рафсона используется отношение функции к ее производной. Вспомним формулу 28. Таким образом, не производя скалярного произведения мы получаем и значение функции в точке, и значение ее производной, таким образом экономим вычислительное время не выполняя лишних расчетов. Хоть и в общем случае алгоритм поиска корня Ньютона-Рафсона сходится быстрее дихотомии, он может и не сойтись вовсе, для этого вместе с ним и работает другой алгоритм из другого класса (прямая оптимизация вместо первого порядка). Запустить расчёт двух функций одновременно можно с помощью механизма параллелизма multiprocessing. Для этого замкнем функции линейной оптимизации, чтобы они не вычислялись раньше времени, например, при создании списка их них, передадим функции imap unordered как второй аргумент, а как первый - конструкцию вызова размыкания lambda x: x(). imap является итерирующей, потому взять значение функции, которая вычислилась раньше другой просто:

```
1       res = 0
2       lo = lambda l: f(*tuple(map(lambda x,y: x-l*y, point_0,nabla_0
            )))
3       for k in imap_unordered(lambda x:x(),
4           [dichotomy_opt(lo, domain=(0,1000), eps = eps),
5           newtonraphson(d(l0), domain[0], eps=eps**0.5)]):
6           res = k
7           break
```

# 5 Сравнение с классическими методами

## 5.1 Тест 1. Производная

Найдем значения производных функций в некоторых точках с помощью метода конечных приращений, символьного вычисления и введения кольца, измерим точность и время работы каждого метода и занесем данные в таблицу. Функцию запишем следующим образом:

```
1       def f(x): return x**2 - 2*x + 1
```

Из классических методов можно отметить системы символьных вычислений, отличающихся наилучшей точностью, так как оперируют математическими символами, а не десятичными приближениями, и численный метод, основанный на теореме Лагранжа о конечных приращениях, выделяющийся скоростью вычислений, так как требует лишь вычисления значения функции в двух точках. Следует отметить заранее, что оригинальная теорема подверглась изменению для оптимизации точности (32), однако все равно для быстро растущих функций остается неустойчивой [9], а параметр eps подбирался в ходе эксперимента таким образом, чтобы обеспечить наименьшую разницу со значением, полученным от численной подстановки в функцию, полученную символьными методами в пределах одного класса чисел float.

$$D_x f \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x},\ |\Delta x| << 1 \tag{32}$$

Метод конечных приращений приведен ниже [Листинг 1], символьное вычисление производной - [Листинг 2], вычисление производной в кольце - [Листинг 3]. Все имеют одинаковую сигнатуру, оформленную в виде замыканий: производная в точке x вычисляется как d(f)(x).

Листинг 1: FiniteAdditions.py

```
1        def d(f):
2         eps = 1e-7 #
3         def f_(x):
4          return (f(x+eps) - f(x-eps))/(2*eps)
5         return f_
```

Листинг 2: SymbDerivative.py

```
1        from sympy import *
2        x = symbols('x')
3        f = ... #определение функции
4        def d(f):
5         def f_(v):
6          return f.diff(x).subs({x:v}).n()
7         return f_
```

Листинг 3: Differntial.py

```
1        ...
2        def d(f):
3         def f_(x):
4          return f(DCalculus(x,1)).I()
5         return f_
6        ...
```

Испытывались два вида функций: алгебраические и трансцендентные в случайных точках, в таблицу занесен средний результат на выборке из 10000 испытаний на каждый аргумент для уменьшения погрешности, испытания проводились на ЭВМ с характеристиками из [Приложения 9.2].

Таблица 4: Фрагмент таблицы "Результаты вычисления производных. Время, $\mu s$."

| $f(x)$ | $x$ | Конечные приращения | Символьные вычисления | Кольцо $\Phi$ |
|---|---|---|---|---|
| $x^2 - 6x + 9$ | 4 | 0.870 | 168 | 36.8 |
| | 127.5 | 0.733 | 195 | 36.7 |
| | 95412 | 1.010 | 170 | 36.9 |
| $sin(x) - e^x$ | -2 | 0.764 | 445 | 17.7 |
| | $-150.94$ | 0.765 | 198 | 17.7 |
| | 418.21 | 0.781 | 241 | 17.7 |

Таблица 5: Фрагмент таблицы "Результаты вычисления производных. Значения."

| $f(x)$ | $x$ | Конечные приращения | Симв. вычисления | Кольцо $\Phi$ |
|---|---|---|---|---|
| $x^2 - 6x + 9$ | 4 | 2.000000129953605 | 2.0 | 2.0 |
| | 127.5 | 248.99996788008139 | 249.00 | 249.0 |
| | 95412 | 190830.23071289062 | 190818.00 | 190818.0 |
| $sin(x) - e^x$ | $-2$ | -0.5514820822227762 | -0.551482119783755 | -0.5514821197837552 |
| | -150.94 | 0.9897139521819476 | 0.989714003690002 | 0.9897140036900024 |
| | 418.21 | -4.2295620016818725e+181 | -4.22956084313071e+181 | -4.2295608431307084e+181 |

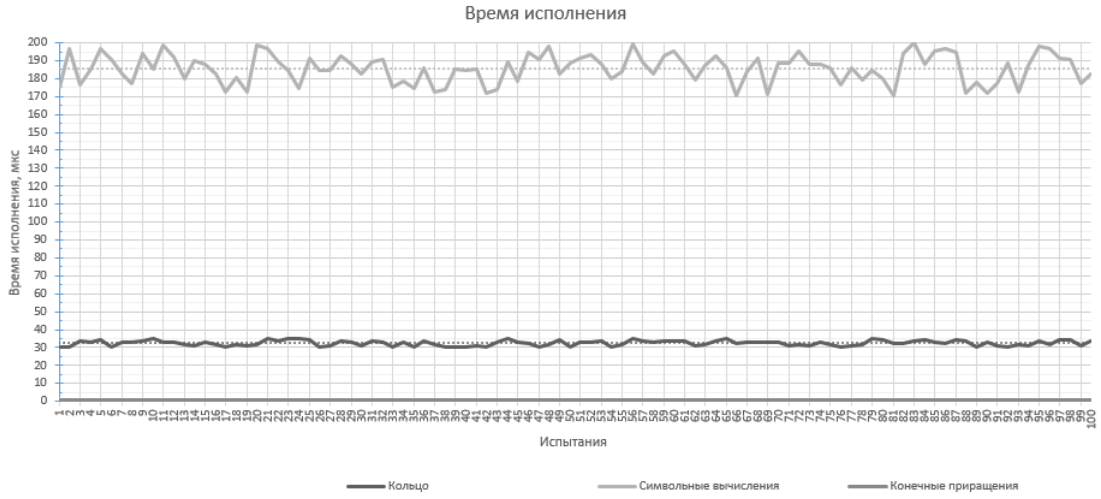


Рис. 6: Время вычисления значения производной алгебраической функции в точке.

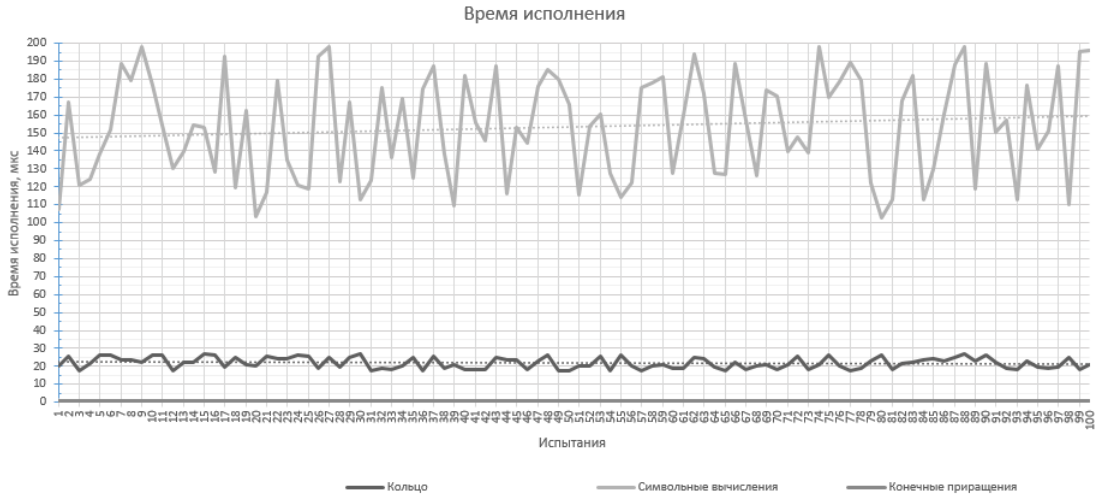


Рис. 7: Время вычисления значения производной трансцендентной функции в точке.

Благодаря использованию таблиц дифференцирования производные элементарных трансцендентных функций вычисляются быстрее. Более низкая скорость работы вычисления производной алгебраической функции объясняется тем, что изначально программа не знает, является ли показатель степени нестационарной функцией, а все числа

преобразуются в процессе вызова переопределенного оператора к числам из алгебры $\Phi$ для поддержания замкнутости, потому в общем виде по свойствам сложной функции $D_x(u(x)^{v(x)}) = u(x)^{v(x)-1} \cdot (v(x) \cdot D_x u(x) + u(x) \ln u(x) \cdot D_x v(x))$. Очевидно, что такое количество операций значительно увеличивает время работы.

Рассмотрим меру роста времени вычисления значения производной функции (33) от $n$ в точке $x = 5$.

$$\sum_{i=0}^{n} (x+1)^i \tag{33}$$

Запишем функцию и вычисление ее производной следующим образом.

Листинг 4: DerivativeOfSum.py

```
...
def S(i):
  def ff(x):
    s = 0
    for j in range(i):
      s += (x+1)**i
    return s
  return ff

i = ..
%timeit d(S(i))(5)
...
```

И проведем схожие с предыдущими измерения времени работы, результаты которых занесем в нижеприведенную таблицу.

Таблица 6: Результат вычисления производной, заданной конечной суммой. Время, $\mu$s.

| $n$ | Расчёт суммы | Конечные приращения | Симв. вычисления | Кольцо $\Phi$ |
|---|---|---|---|---|
| 1 | 0.86 | 1.78 | 113.2 | 18.4 |
| 2 | 1.24 | 2.41 | 133.6 | 32.3 |
| 3 | 1.62 | 3 | 145.6 | 46.5 |
| 5 | 2.45 | 4.35 | 169.1 | 74.3 |
| 10 | 4.44 | 7.26 | 231 | 143 |
| 15 | 7.2 | 11.2 | 283 | 216 |
| 25 | 12.3 | 18 | 394 | 357 |
| 50 | 26 | 36.8 | 861 | 712 |
| 100 | 59.7 | 80.2 | 1512 | 1450 |
| $R^2$ | 0.9973 | 0.9948 | 0.996 | 0.9999 |

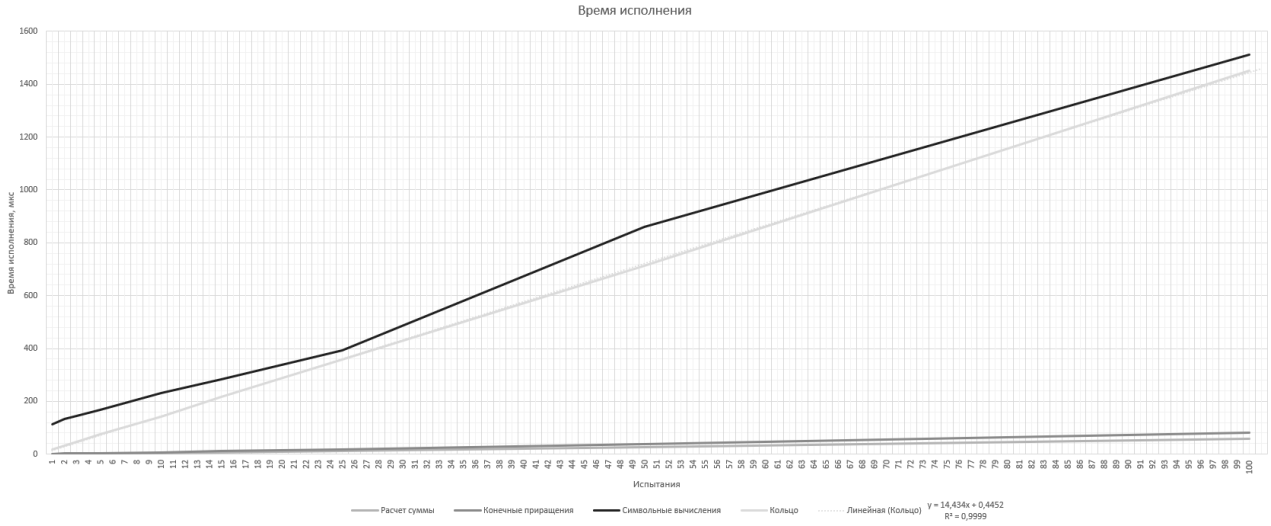


Рис. 8: Время вычисления значения производной функции, заданной суммой, в точке.

Из полученных данных видно, что при росте n, любой из методов растет линейно, так как и сам расчёт функции требует линейного времени. Однако, следует отметить, что вычисление производной с помощью кольца наиболее устойчиво, коэффициент детерминации $R^2 = 1 - \dfrac{\sigma_n^2}{\sigma_t^2}$ ближе к 1, чем остальные.

## 5.2 Тест 2. Оптимизация функции

Поставим задачу поиска оптимума функции $f(x, y)$, являющегося вершиной эллиптического параболоида.

$$f(x, y) = 10x^2 + y^2$$

Для начала найдем ответ аналитически:

$$\begin{aligned}
&f(x, y) = 10x^2 + y^2 \\
&\overrightarrow{n_0} = (x_0, y_0) = argmin_{n \in \mathbb{R}^2} f(*\overrightarrow{n}) = f(x_0, y_0) - ? \\
&\frac{\partial f}{\partial x} = 20x \\
&\frac{\partial f}{\partial y} = 2y \\
&\nabla f = (20x, 2y) \\
&\begin{cases} 20x_0 = 0 \\ 2y_0 = 0 \end{cases} \Rightarrow \begin{cases} x_0 = 0 \\ y_0 = 0 \end{cases}
\end{aligned}$$

$$A = \frac{\partial^2 f}{\partial x^2} \mid_{\overrightarrow{n_0}} = 20$$
$$B = \frac{\partial^2 f}{\partial y^2} \mid_{\overrightarrow{n_0}} = 2$$
$$C = \frac{\partial^2 f}{\partial x \partial y} \mid_{\overrightarrow{n_0}} = 0$$
$$AC - B = 20 \cdot 2 - 0 = 40 > 0$$
$$A = 20 > 0$$
$$\Rightarrow \overrightarrow{n_0} = (0,0) \text{ - минимум функции } f(x,y)$$

Таким образом, правильным ответом будет пара (0,0).

Теперь получим ответ численным методом. Запишем эту функцию и найдем ее оптимум методом градиентного спуска:

```
1   def f(x,y): return 10*x**2+y**2
2   opt = gradient_descent(f, start=(10,10,), domain=((-100, 100)
        ,), eps=1e-6)
3   print (opt)
```

Ответом программы является пара (5.922274725121076e-14 -1.1252020499405204e-10) довольно близкая по значению к правильному, а ответ получен всего за 10 итераций [Приложения: Спуск 7] в среднем за 0.9 ms на тестируемой машине с характеристиками из Приложения.

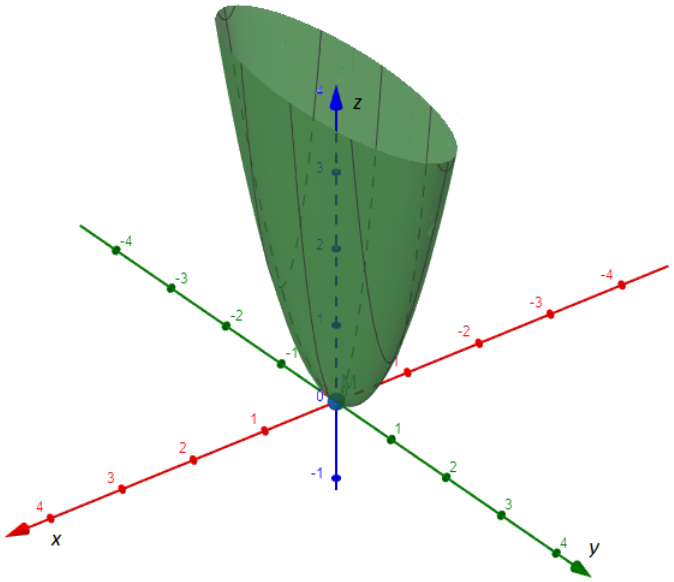


Рис. 9: $f(x,y) = 10x^2 + y^2$

Рассмотрим другой пример. Функция Розенброка - невыпуклая функция двух переменных, используемая для оценки оптимизационных алгоритмов (34).

$$R(x,y) = (1-x)^2 + 100(y-x^2)^2 \tag{34}$$

Пара $(1,1)$ является минимумом функции, $R(1,1) = 0$. Градиентный спуск с дифференцированием, реализованным с помощью метода конечных приращений, даже с

$\varepsilon = 10^{-12}$ не смог найти и за 50000 итераций оптимума данной функции. Наш градиентный спуск нашел за 503 итерации ( 7.5 ms) довольно близкую к правильному ответу пару - (0.9999925731400436, 0.9999853369329239)

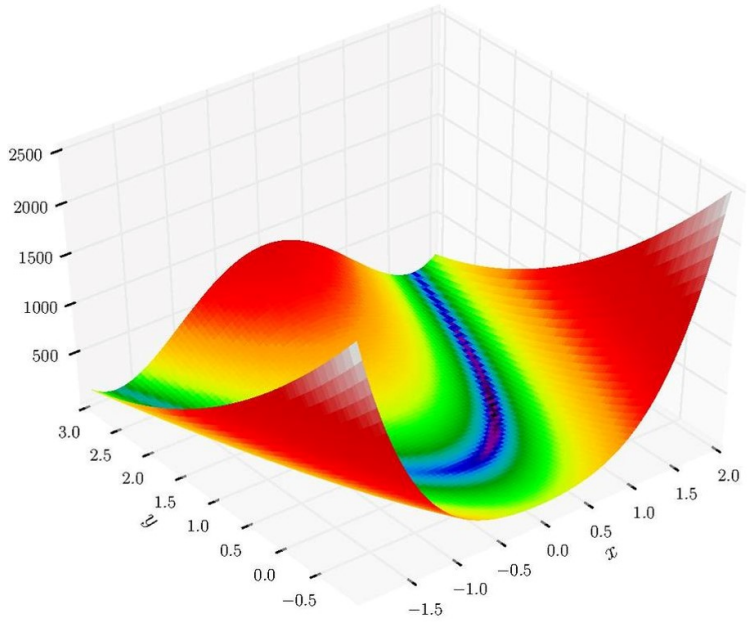


Рис. 10: $f(x, y) = (1 - x)^2 + 100(y - x^2)^2$

# 6 Решение оптимизационных задач

Рассмотрим решение оптимизационных задач.

**Задача 1.** Найти угол, при котором средняя скорость полёта за первую секунду тела максимальна, считая, что сила сопротивления воздуху прямо пропорциональна модулю скорости и противоположна по направлению, при запуске с горизонта с начальной скоростью $v_0 = 1$м/с, $m = 1$ кг.

***Решение.*** Составим, для начала, математическую модель. Тело летит в близи земли, считаем, что на него действует ускорение свободного падения $\overrightarrow{g}$. Запишем II закон Ньютона для него:

$$m\overrightarrow{a} = mD_t\overrightarrow{v} = m\overrightarrow{g} + \overrightarrow{F}_c$$

Пусть $\overrightarrow{F} = -k\overrightarrow{v}$. Введём ДСКП так, что прямая $y = 0$ совпадает с горизонтом. Спроецируем уравнение на оси:

$$\begin{cases} D_t v_x = -\frac{k}{m} v_x \\ D_t v_y = -g - \frac{k}{m} v_y \end{cases}$$

Заменим $\dfrac{k}{m}$ на $\omega_0$ Данные дифференциальные уравнения имеют следующие решения:

$$\begin{cases} v_x(t) = v_{0_x} e^{-\omega_0 t} \\ v_y(t) = v_{0_y} e^{-\omega_0 t} - \omega_0 g(1 - e^{-\omega_0 t}) \end{cases}$$

где $v_{0_x} = v_0 \cos(\alpha)$, $v_{0_y} = v_0 \sin(\alpha)$. Проинтегрируем оба уравнения:

$$\begin{cases} x(t) = \int_0^\tau v_x(t)dt = v_0 \cos(\alpha)\frac{m}{k}\left(1 - e^{-\frac{k}{m}t}\right) - gt \\ y(t) = \int_0^\tau v_y(t)dt = \frac{m}{k}\left(-gt + (v_0 \sin(\alpha) + \frac{mg}{k})(1 - e^{-\frac{k}{m}t})\right) \end{cases}$$

Средняя скорость рассчитывается по формуле $\frac{S}{t}$, где $S$ - путь, пройденный телом за время $t$. $S_x = x(\tau) - x(0), S_y = y(\tau) - y(0)$, тогда $S = \sqrt{S_x^2 + S_y^2}$. Учитывая, что $x(0) = y(0) = 0$,

$$v_{\text{ср}} = \frac{\sqrt{\left(v_0 \cos(\alpha)\frac{m}{k}\left(1 - e^{-\frac{k}{m}t}\right) - gt\right)^2 + \left(\frac{m}{k}\left(-gt + (v_0 \sin(\alpha) + \frac{mg}{k})(1 - e^{-\frac{k}{m}t})\right)\right)^2}}{\tau}$$

Запишем это как функцию для ЯП Python с учётом того, что $\tau = 1$с. и найдём ее оптимум градиентным спуском. Так как ранее мы находили минимумы «спускаясь», а теперь задача состоит в поиске максимума, домножим функцию на -1. Зададим $g = 10$, константы пропорциональности $k$ положим 1. Область поиска корня domain = (0, pi/2).

```
1        def v(a):
2            return -sqrt((-g*1 + m/k*v0*cos(a)*(1-exp(-k/m*1)))**2 +
                 (-g*1 + (v0*sin(a) + m*g/k)*(1 - exp(-k/m*1)))**2)/1
```

Программа возвращает ответ: 1.5707963267948974, что, практически, совпадает с рациональным приближением $\frac{\pi}{2}$

Ответ: $\alpha \approx 1.5707963267948974$

**Задача 2.** На какой высоте $h$ над центром круглого стола надо подвесить лампочку, чтобы освещенность края была наибольшей?[8]. Считать радиус стола $R = 3$м, силу света лампочки $I = 100$кд.

***Решение.*** Построим к задаче чертеж (Рис.11 ) Пусть расстояние до края - $L$, тогда освещенность края находится по формуле (35).

$$E = \frac{I}{L^2}\cos\alpha \tag{35}$$

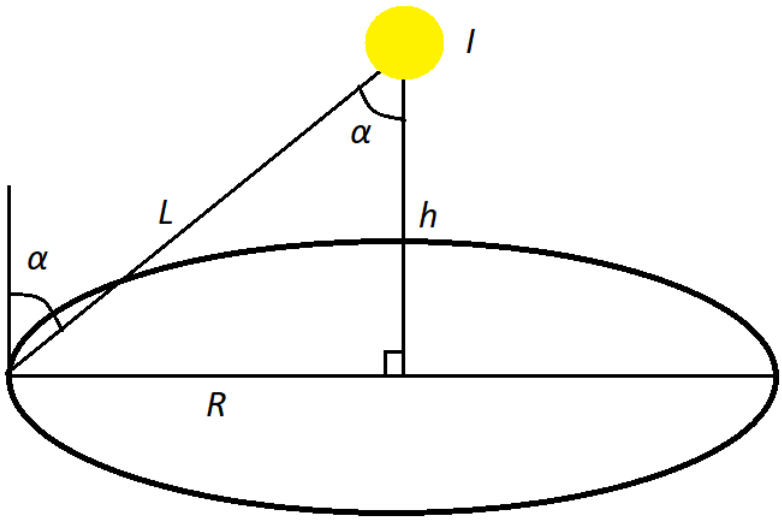


Рис. 11: Чертеж к Задаче 2

Заметим из прямоугольного треугольника, что $\cos\alpha = \frac{h}{L}$, а $L = \sqrt{h^2 + R^2}$. Теперь функцию освещенности возможно задать через высоту:

$$E(h) = \frac{I}{h^2 + R^2} \cdot \frac{h}{\sqrt{h^2 + R^2}}$$

$$E(h) = \frac{Ih}{(h^2 + R^2)^{\frac{3}{2}}}$$

Необходимо найти такие $h_0, E_0$, что $h_0 : E(h_0) \overset{\mathbb{R}_+}{\longmapsto} min = E_0 - ?$ Запишем функцию $E(h)$ в ЯП Python:

```
1        I = 100
2        R = 3
3        def E(h): return I*h/((h**2 + R**2)**1.5)
```

И оптимизируем с помощью градиентного спуска:

```
1        h_0 = gradient_descent(E, start=(0,0,), domain=((0, 10000)
             ,), eps=1e-6)
2        E_0 = E(h_0)
3        print(h_0, E_0)
```

Результат выполнения программы: 2.121320343559642 4.276668660663894.

Ответ: наибольшая освещенность края в 4.276668660663894 люкса достигается высоте лампочки над центром стола в 2.121320343559642 метров.

# 7 Перспективы

В дальнейшем планируется расширить алгебраическое кольцо до колеса, и вычислять с его помощью смешанные производные второго порядка, что необходимо для составления матрицы Гессе для метода оптимизации BFGS, а также реализо вать сам метод BFGS.

## Области применения

Результат работы может быть применен в сферах деятельности, где необходимо точно и быстро рассчитывать производные и вычислять оптимальные значения некоторых функций. Например, для автономного управления на предприятиях при расчете оптимальных изменений системы в реальном времени, принимая за аргументы эвристических функций показания датчиков данной системы. Также возможно внедрение в сервисы онлайн-дистрибуции, такие как интернет-магазины и кассы для динамического ценообразования при учете многих статистических и реальных параметров.

# 8 Заключение

Рассмотренные в данной работе модификации, а именно, параллелизм вычислений, введение алгебраического кольца, создание таблицы правил, действительно усовершенствовали как методы дифференцирования, так и алгоритмы поиска оптимумов заданных функций по точности относительно классических численных методов и по скорости при сравнении с системами символьных вычислений. Данные улучшения могут быть внедрены и для практического применения.

## Список литературы


[1] Фихтенгольц Г.М. Курс дифференциального и интегрального исчисления. том 1. — М.: Изд. ФИЗМАТЛИТ., 2001. 616 с.

[2] Сканави М.И. Элементарная математика. — М.: Изд. Наука, 1967, 609 с.

[3] Просветов Г.И. Математический анализ. Задачи и решения. — М.: Изд. Альфа-пресс, 2018, 271 с.

[4] Поршнев С.В. Численные методы на базе MathCAD. — СПб.: Изд. БХВ-Птербург, 2005, 464 с.

[5] Яглом И.М. Комплексные числа — М.: Изд. ФИЗМАТЛИТ., 1963.

[6] Setzer, A. Wheels — 1997, 9 p.

[7] Новиков М.Ю. ИСПОЛЬЗОВАНИЕ СИМВОЛЬНЫХ ВЫЧИСЛЕНИЙ В ЗАДАЧАХ КОНЕЧНОМЕРНОЙ ОПТИМИЗАЦИИ // Инновации в науке: сб. ст. по матер. XLIII междунар. науч.-практ. конф. № 3(40). – Новосибирск: СибАК, 2015.

[8] Васильева В.А., Кудрина Т.Д., Методическое пособие по математике для поступающих в вузы. — М.: Изд. МАИ, 1992, 304 с.

[9] Иванова Т.В., Численные методы в оптике. — СПб: Университет ИТМО, 2017, 84 с.

# 9 Приложения

## 9.1 Спуск

Таблица 7: Последовательный спуск до оптимума функции

| i | $(x, y)$ |
|---|---|
| 1 | (10,10) |
| 2 | $(-190.00001664153217, -10.000001664153219)$ |
| 3 | $(0.004859581337740337, -8.999975920991998)$ |
| 4 | $(-0.04372348522719183, -0.002360717875413698)$ |
| 5 | $(1.118304285376015 \cdot 10^{-6}, -0.002124640049925209)$ |
| 6 | $(-1.0061951036467986 \cdot 10^{-5}, -5.295966426081301 \cdot 10^{-7})$ |
| 7 | $(2.573519220807046 \cdot 10^{-10}, -4.7663562381166685 \cdot 10^{-7})$ |
| 8 | $(-2.315492257717275 \cdot 10^{-9}, -1.2502280529072341 \cdot 10^{-10})$ |
| 9 | $(5.922274725121076 \cdot 10^{-14}, -1.1252020499405204 \cdot 10^{-10})$ |

## 9.2 Характеристики тестируемой машины

| **Hard** | |
|---|---|
| CPU | Intel Core i5 8250U |
| RAM | 4GB |
| **Soft** | |
| OS | OpenSuse 42.3 |
| Enviroment | Python 3.7.2 |

## 9.3 Логотип работы