НАО «Восточно-Казахстанский технический университет имени Д. Серикбаева»

Дисциплина «Методы оптимизации»

ЛАБОРАТОРНАЯ РАБОТА 3

Методы с использованием информации о производной функции

Лабораторная работа 3 по рабочей программе. Объём — 5 академических часов. Образовательная программа 6B06103 Математическое и компьютерное моделирование.

1 Цель и порядок работы

Цель работы — научиться находить минимум функции одной переменной с использованием её производных, обосновывать точность результата и сравнивать вычислительные затраты. Изучаются деление пополам по знаку производной, метод хорд с сохранением отрезка локализации и метод Ньютона с защитными шагами.

В лабораторной 2 сравнения значений функции позволяли сокращать интервал поиска. Здесь используется знак первой производной, а в методе Ньютона также вторая производная. Для выполнения работы необходимо понимать непрерывность, производную, выпуклость, теорему Лагранжа и формулу Тейлора. Сначала изучите методическую часть и выполните ручные примеры, затем переходите к программам и индивидуальному варианту.

Ожидаемый результат — отчёт с аналитическим обоснованием, ручными шагами, таблицами вычислений, графиками, сравнительным выводом и ответами на контрольные вопросы. Один академический час принимается равным 45 минутам; суммарный ориентир составляет 225 минут.

Этап	Содержание	Минуты
Подготовка	Постановка, выпуклость и критерии точности	35
Ручной расчёт	Деление пополам, хорда и шаги Ньютона	40
Работа с кодом	Разбор кода и проверка контрольного примера	60
Эксперимент	Свой вариант, таблицы и графики	50
Отчёт и защита	Сравнение методов и ответы	40
Итого	Пять академических часов	225

2 Математическая постановка

Требуется решить задачу

$$\min_{x\in[a_0,b_0]} f(x), \qquad a_0 < b_0.$$

В обязательной части предполагается $f \in C^2[a_0, b_0]$ и известна аналитически обоснованная константа $\mu > 0$, для которой

$$f''(x) \geq \mu \quad \text{для всех } x \in [a_0, b_0].$$

Это достаточное условие строгой выпуклости. Производная $g(x) = f'(x)$ строго возрастает. Непрерывная функция на замкнутом отрезке достигает минимума; при строгой выпуклости минимизатор единственен.

2.1 Внутренний и граничный минимум

Сначала вычисляют $g(a_0)$ и $g(b_0)$. Возможны три случая:

$$\begin{cases} g(a_0) \geq 0 & \Rightarrow x^* = a_0, \\ g(b_0) \leq 0 & \Rightarrow x^* = b_0, \\ g(a_0) < 0 < g(b_0) & \Rightarrow g(x^*) = 0, \quad x^* \in (a_0, b_0). \end{cases}$$

В первом случае производная на всём отрезке неотрицательна, функция возрастает. Во втором — она убывает. В третьем непрерывная строго возрастающая производная имеет ровно один нуль. Именно этот нуль соответствует минимуму.

Стационарность без выпуклости недостаточна. Например, для $f(x) = -x^2$ точка $x = 0$ удовлетворяет $f'(0) = 0$, но является максимумом. Для $f(x) = x^3$ стационарная точка 0 вообще не является локальным экстремумом. Поэтому проверка формы функции предшествует запуску численного метода.

2.2 Три вида численной информации

Алгоритмы получают значения $f(x)$, $g(x) = f'(x)$ и, при необходимости, $h(x) = f''(x)$. В этой работе производные задаются аналитическими формулами. Их вычисления считаются отдельно:

$$N_f = \text{число вызовов } f, \qquad N_g = \text{число вызовов } f', \qquad N_h = \text{число вызовов } f''.$$

Стоимость одного вызова разных функций может существенно различаться. Поэтому небольшое число итераций само по себе ещё не означает малой вычислительной стоимости. Численное дифференцирование в обязательную часть не включено: оно потребовало бы учёта шага разности, погрешности производной и дополнительных обращений к f.

3 Как контролировать точность

В этой работе ε_x задаёт допустимую абсолютную ошибку аргумента:

$$|\hat{x} - x^*| \leq \varepsilon_x.$$

В лабораторной 2 допуск задавался для длины итогового интервала. При выдаче его середины тому условию соответствует ошибка не более половины длины. Здесь обозначение ε_x специально отличает требуемую ошибку точки от длины интервала и от ошибки значения функции.

3.1 Оценка по локализующему отрезку

Если $x^* \in [a, b]$ и $\hat{x} \in [a, b]$, то

$$|\hat{x} - x^*| \leq \max\{\hat{x} - a,\ b - \hat{x}\}.$$

Для середины $\hat{x} = (a + b)/2$ правая часть равна $(b - a)/2$. Поэтому достаточное условие остановки для середины — $b - a \leq 2\varepsilon_x$. Для произвольной точки хорды или Ньютона применять половину длины интервала без дополнительного обоснования нельзя.

3.2 Оценка по производной

Пусть минимум внутренний. По теореме Лагранжа для некоторой точки ξ между x и x^*:

$$g(x) - g(x^*) = f''(\xi)(x - x^*).$$

Так как $g(x^*) = 0$ и $f''(\xi) \geq \mu$, получаем

$$|x - x^*| \leq \frac{|g(x)|}{\mu}.$$

Следовательно, условие $|g(x)| \leq \mu\varepsilon_x$ гарантирует требуемую ошибку аргумента при точном вычислении производной. Константа μ должна быть нижней границей второй производной на всём рассматриваемом отрезке. Подстановка произвольного положительного числа лишает проверку её смысла.

Для $f(x) = 10^{-8}(x-1)^2/2$ производная в точке 0 равна -10^{-8}, хотя ошибка аргумента равна 1. Малость производной объясняется плоской формой функции. Верная нижняя граница $\mu = 10^{-8}$ даёт оценку $|g(0)|/\mu = 1$.

В программе используются сразу две независимые оценки:

$$E(x; a, b) = \min\left\{\frac{|g(x)|}{\mu},\ \max\{x - a, b - x\}\right\}.$$

Остановка выполняется при $E \leq \varepsilon_x$. В отчёт заносится и эта верхняя оценка, и фактическая ошибка относительно известного контрольного решения. Они обычно различаются.

3.3 Ошибка значения функции и округление

При внутреннем минимуме и $|f''(x)| \leq M_2$ на отрезке формула Тейлора даёт

$$0 \leq f(\hat{x}) - f(x^*) \leq \frac{M_2}{2}|\hat{x} - x^*|^2.$$

Это другая величина, чем ошибка аргумента. В частности, маленькая разность значений функции сама по себе не даёт требуемой оценки аргумента без дополнительных условий.

Приведённые оценки предполагают точные значения производных. Если известно, что вычисленная производная $\tilde{g}$ имеет ошибку не более η, безопасная остаточная оценка принимает вид $(|\tilde{g}(x)| + \eta)/\mu$. При работе с float очень малый допуск может оказаться недостижимым; совпадение двух машинных чисел не является новым математическим доказательством точного равенства.

4 Деление пополам по знаку производной

Пусть $g(a) < 0 < g(b)$. Вычислим середину $m = (a + b)/2$ и $g(m)$. Так как производная строго возрастает, при $g(m) < 0$ корень находится справа и можно положить $a = m$. При $g(m) > 0$ корень находится слева и полагают $b = m$. При $g(m) = 0$ найдено стационарное решение в точной арифметике.

После каждого выполненного сокращения отрезка

$$L_j = \frac{b_0 - a_0}{2^j}.$$

Для обеспечения $L_j/2 \le \varepsilon_x$ достаточно

$$j \ge \left\lceil \log_2 \frac{b_0 - a_0}{2\varepsilon_x} \right\rceil,$$

если начальный отрезок ещё не удовлетворяет условию. Число j здесь обозначает сокращения, а не все проверенные пробные точки. В программе k — число внутренних пробных точек, в которых вычислена производная, включая последнюю проверку перед остановкой. Поэтому $k \le \max\{0, \lceil \log_2((b_0 - a_0)/(2\varepsilon_x)) \rceil\} + 1$; по остаточной оценке остановка может произойти раньше.

Метод не требует второй производной при выборе точек. Нижняя граница μ используется только для дополнительного критерия остановки. Если такой границы нет, для середины можно использовать критерий по длине интервала при сохранении знакопеременности и монотонности g.

5 Метод хорд с сохранением отрезка

Через точки $(a, g(a))$ и $(b, g(b))$ проведём прямую. Её уравнение

$$\ell(x) = g(a) + \frac{g(b) - g(a)}{b - a}(x - a).$$

Полагая $\ell(x) = 0$, получаем новую точку:

$$x_{\text{new}} = a - \frac{g(a)(b - a)}{g(b) - g(a)} = \frac{a g(b) - b g(a)}{g(b) - g(a)}.$$

При $g(a) < 0 < g(b)$ знаменатель положителен, а точка находится строго внутри отрезка в точной арифметике. Вычислив $g(x_{\text{new}})$, заменяют ту границу, производная на которой имеет тот же знак. Значение производной на неизменившемся конце сохраняют и повторно не вычисляют.

Такая схема также называется regula falsi. Она использует значения в концах текущего знакопеременного отрезка. При близком к линейному поведении g хорда может быстро дать хорошее приближение.

У метода есть существенная особенность: один конец отрезка может долго оставаться неподвижным. Например, для выпуклой производной $g(x) = x^3 + x - 1$ на $[0,2]$ приближения хорд подходят к корню слева, а правый конец может оставаться равным 2. Поэтому длина сохраняемого отрезка не обязана быстро стремиться к нулю. Критерий $|g(x)|/\mu \le \varepsilon_x$ позволяет правильно завершить такой расчёт.

Метод не имеет универсальной гарантии выигрыша по числу вычислений над делением пополам. Это проверяется экспериментально при одинаковом определении точности.

6 Метод Ньютона и защитные шаги

6.1 Вывод формулы

Применим метод Ньютона к уравнению $g(x) = f'(x) = 0$. В точке x_k заменим g её касательной:

$$g(x) \approx g(x_k) + g'(x_k)(x - x_k).$$

Нуль этой линейной модели даёт

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)} = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

Таким образом, касательная строится к графику первой производной. Та же формула получается из минимума квадратичной модели целевой функции при $f''(x_k) > 0$:

$$q_k(s) = f(x_k) + f'(x_k)s + \frac{1}{2}f''(x_k)s^2, \qquad s_k = -\frac{f'(x_k)}{f''(x_k)}.$$

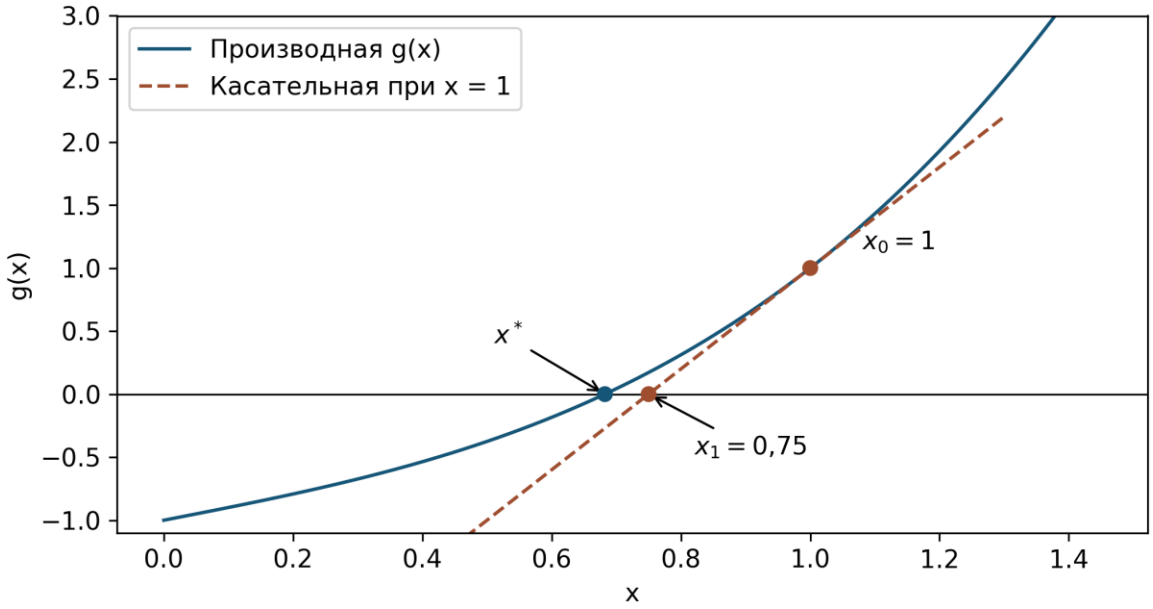


Рисунок 1 — Геометрический смысл шага Ньютона для уравнения производной

6.2 Локальная сходимость

Пусть $f \in C^3$ в окрестности минимума, $f'' \geq \mu > 0$, $|f'''| \leq M_3$, а начальная точка достаточно близка к x^*. Разложение $g(x^*)$ около x_k приводит к

$$|x_{k+1} - x^*| \leq \frac{M_3}{2\mu}|x_k - x^*|^2.$$

При достаточно малой начальной ошибке правая часть становится существенно меньше текущей ошибки. Это квадратичная локальная сходимость. Слова «локальная» и «при выполнении условий» здесь существенны: произвольная начальная точка не обеспечивает такой результат. Общую формулу Ньютона и условия локальной сходимости можно сопоставить с открытой справкой NIST [1, §3.8(ii)].

Для $f(x) = \sqrt{1 + x^2}$ имеем $f'(x) = x/\sqrt{1 + x^2}$ и $f''(x) = (1 + x^2)^{-3/2} > 0$. Обычный шаг Ньютона равен $x_{k+1} = -x_k^3$. Из $x_0 = 2$ получается $x_1 = -8$: даже строго выпуклая функция не гарантирует удачный шаг из далёкой точки.

6.3 Защита в учебной реализации

В программе сохраняется отрезок $[a, b]$ с $g(a) < 0 < g(b)$. Начальная пробная точка — его середина. После каждой проверки обновляется соответствующая граница. Новая точка выбирается так:

1. Если число уже проверенных внутренних точек кратно пяти, следующей берётся середина текущего отрезка.
2. В остальных случаях вычисляется $f''(x)$ и предлагается шаг Ньютона.
3. Если знаменатель неположителен или полученная точка выходит из открытого интервала (a, b), используется середина.
4. Если точка допустима, принимается шаг Ньютона.

Периодические шаги середины обеспечивают регулярное сокращение интервала, пока критерий точности не выполнен. Защитная схема может сделать больше шагов, чем чистый метод Ньютона при хорошем старте. Квадратичная оценка относится к обычным шагам Ньютона в подходящей окрестности, а не ко всем шагам смешанного алгоритма.

По условиям обязательной части f'' положительна. Обнаруженная в программе неположительная кривизна требует дополнительно проверить введённую производную и доказательство предпосылок; защитная замена сама по себе не доказывает выпуклость произвольной функции.

7 Сопоставление методов

Метод	Новая точка	Информация для шага	Особенность
По знаку производной	Середина отрезка	f'	Длина после сокращения уменьшается вдвое
Хорды	Нуль линейной интерполяции f'	f' в концах	Возможен почти неподвижный конец
Ньютон с защитой	Нуль касательной к f' или середина	f' и f''	Быстрые локальные шаги и периодическая защита

Во всех трёх методах нужно заранее обосновать применимость, определить смысл допуска и одинаково считать вызовы. Для граничного минимума внутренние итерации вообще не требуются.

8 Разобранный контрольный пример

Рассмотрим

$$f(x) = \frac{x^4}{4} + \frac{x^2}{2} - x, \qquad x \in [0,2].$$

Тогда

$$g(x) = x^3 + x - 1, \qquad f''(x) = 3x^2 + 1 \geq 1.$$

Можно взять $\mu = 1$. Имеем $g(0) = -1$, $g(2) = 9$, поэтому минимум внутренний и единственный. Контрольное решение уравнения $x^3 + x - 1 = 0$:

$$x^* \approx 0{,}6823278038280193, \qquad f(x^*) \approx -0{,}3953530449018225.$$

Эта контрольная точка вычислена отдельно с повышенной точностью; её вычисления не включаются в счётчики исследуемых алгоритмов.

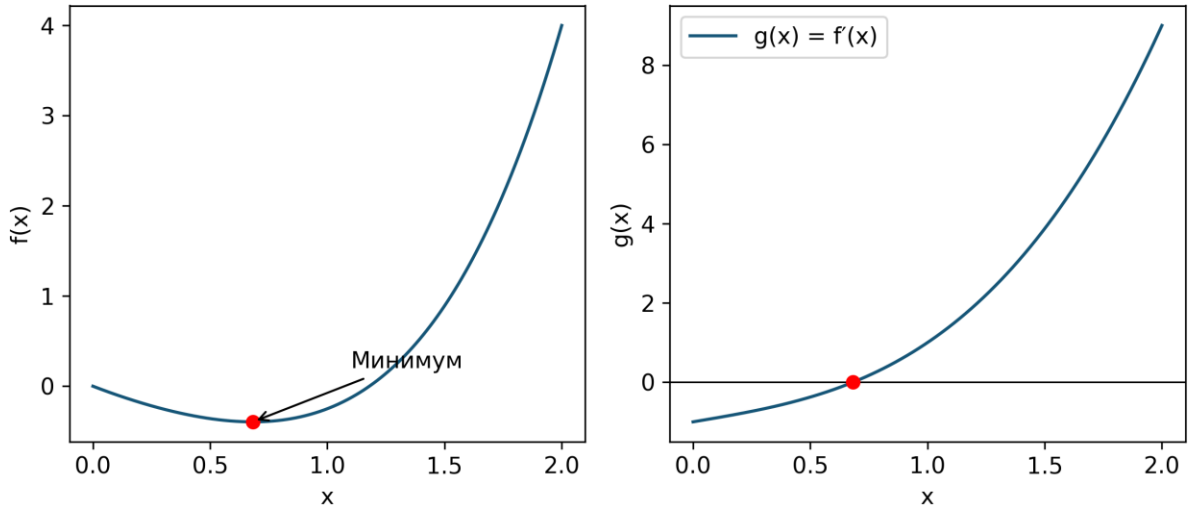


Рисунок 2 — Минимум функции соответствует единственному нулю её производной

8.1 Первые шаги деления пополам

Первая середина равна 1, $g(1) = 1 > 0$, поэтому новым отрезком становится $[0,1]$. Следующая середина 0,5 даёт $g(0{,}5) = -0{,}375$, и остаётся $[0{,}5,1]$. В третьей точке 0,75 производная равна $0{,}171875 > 0$, поэтому остаётся $[0{,}5,0{,}75]$.

В таблице указаны границы до проверки пробной точки; это позволяет однозначно восстановить переход.

k	a	b	x	$g(x)$
1	0.000000	2.000000	1.000000	1.000000
2	0.000000	1.000000	0.500000	-0.375000
3	0.500000	1.000000	0.750000	0.171875
4	0.500000	0.750000	0.625000	-0.130859
5	0.625000	0.750000	0.687500	0.012451

8.2 Первый шаг метода хорд

Из начальных значений $a = 0, b = 2, g(a) = -1, g(b) = 9$ получаем

$$x_1 = 0 - \frac{(-1)(2-0)}{9-(-1)} = 0{,}2, \qquad g(x_1) = -0{,}792.$$

Новая левая граница равна 0,2, правая остаётся равной 2. Следующая пробная точка

$$x_2 = 0{,}2 - \frac{(-0{,}792)(2 - 0{,}2)}{9 - (-0{,}792)} \approx 0{,}3455882353.$$

Сравните это перемещение с сокращением отрезка вдвое. Хорошее использование наклона хорды зависит от формы производной на всём отрезке.

8.3 Первые шаги Ньютона

Начальная середина $x_0 = 1$. Так как $g(1) = 1$ и $f''(1) = 4$, обычный шаг Ньютона даёт

$$x_1 = 1 - \frac{1}{4} = 0{,}75.$$

В следующей точке $g(0{,}75) = 11/64$, $f''(0{,}75) = 43/16$, поэтому

$$x_2 = \frac{3}{4} - \frac{11/64}{43/16} = \frac{59}{86} \approx 0{,}6860465116.$$

В таблице j обозначает номер приближения обычного Ньютона, начиная с 0. Программа считает проверенные точки: начальная точка уже даёт $k = 1$.

j	x_j	$g(x_j)$	$\lvert x_j - x^* \rvert$
0	1.0000000000	1.000e+00	3.177e-01
1	0.7500000000	1.719e-01	6.767e-02
2	0.6860465116	8.941e-03	3.719e-03
3	0.6823395826	2.823e-05	1.178e-05
4	0.6823278039	2.840e-10	1.185e-10

Все эти точки лежат внутри допустимого отрезка. При $\varepsilon_x = 10^{-6}$ критерий выполняется на пятой проверенной точке; до обязательного защитного шага алгоритм успевает завершиться.

8.4 Результаты одного вычислительного эксперимента

В таблице E — верхняя оценка ошибки аргумента, k — число внутренних пробных точек. Для каждого запуска $N_f = 1$: значение цели вычисляется только для выдаваемого результата. Начальные значения производной в двух концах включены в N_g, поэтому $N_g = k + 2$ для внутреннего минимума.

Метод	ε_x	$\hat{x}$	E	k	N_g	N_h	**Ошибка**
По производной	1e-02	0.67968750	6.31e-03	8	10	0	2.64e-03
Хорды	1e-02	0.67884619	8.32e-03	13	15	0	3.48e-03
Ньютон с защитой	1e-02	0.68604651	8.94e-03	3	5	2	3.72e-03
По производной	1e-04	0.68231201	3.78e-05	15	17	0	1.58e-05
Хорды	1e-04	0.68229774	7.21e-05	24	26	0	3.01e-05
Ньютон с защитой	1e-04	0.68233958	2.82e-05	4	6	3	1.18e-05

Метод	ε_x	$\hat{x}$	E	k	N_g	N_h	Ошибка
По производной	1e-06	0.68232822	9.54e-07	21	23	0	4.20e-07
Хорды	1e-06	0.68232740	9.57e-07	34	36	0	3.99e-07
Ньютон с защитой	1e-06	0.68232780	2.84e-10	5	7	4	1.18e-10

При $\varepsilon_x = 10^{-6}$ метод по производной использует 23 вызова f', хорды — 36, а Ньютон — 7 вызовов f' и 4 вызова f''. Последний результат требует оценки стоимости второй производной. Преимущество данного запуска нельзя автоматически переносить на любую функцию и любой начальный интервал.

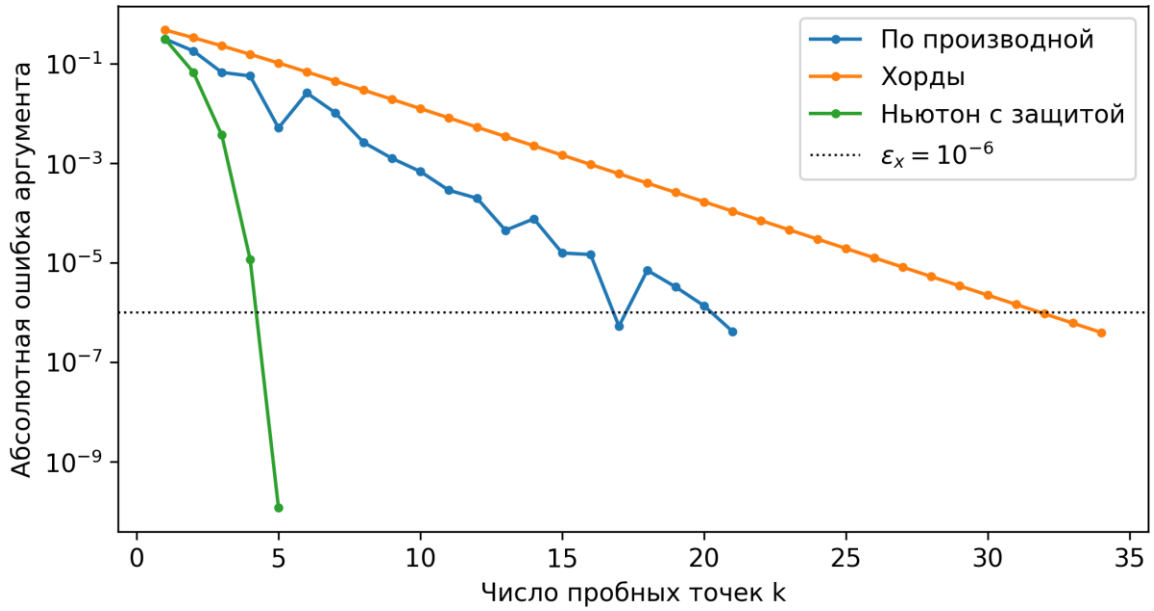


Рисунок 3 — Изменение фактической ошибки по числу проверенных точек

9 Программа на Python

Сохраните блоки разделов 9.1–9.3 последовательно в один файл derivative_methods.py. Готовый файл также включён в архив. Для основного расчёта достаточно стандартной библиотеки Python; NumPy и SciPy не требуются.

9.1 Структура результата и счётчики

```
from dataclasses import dataclass
from math import isfinite


@dataclass
class Result:
    method: str
    x: float
    fx: float
    gx: float
    a: float
    b: float
    bound: float
    probes: int
```

```
    nf: int
    ng: int
    nh: int
    reason: str
    history: list


class Counter:
    def __init__(self, f):
        self.f, self.calls = f, 0

    def __call__(self, x):
        self.calls += 1
        value = float(self.f(x))
        if not isfinite(value):
            raise ValueError("Получено бесконечное или нечисловое значение")
        return value
```

Поле bound хранит оценку ошибки, probes — число внутренних пробных точек, history — таблицу истории. Граничные проверки не считаются внутренними точками, но входят в число обращений к производной.

9.2 Общая функция минимизации

```
def minimize(f, df, ddf, a, b, mu, eps_x=1e-6,
             method="bisection", max_iter=1000):
    names = {"bisection": "По производной", "chord": "Хорды",
             "newton": "Ньютон с защитой"}
    if method not in names:
        raise ValueError("Неизвестный метод")
    a, b, mu, eps_x = map(float, (a, b, mu, eps_x))
    if not all(isfinite(v) for v in (a, b, mu, eps_x)):
        raise ValueError("Требуются конечные параметры")
    if not a < b or mu <= 0 or eps_x <= 0 or max_iter < 1:
        raise ValueError("Требуются a < b, mu > 0, eps_x > 0")
    if eps_x < 100 * 2.22e-16 * max(1.0, abs(a), abs(b)):
        raise ValueError("Допуск слишком мал для масштаба аргумента")
    F, G, H = Counter(f), Counter(df), Counter(ddf)
    ga, gb = G(a), G(b)
    history = []

    def finish(x, gx, error_bound, reason):
        fx = F(x)
        return Result(names[method], x, fx, gx, a, b,
                      error_bound, len(history), F.calls,
                      G.calls, H.calls, reason, history)

    if ga >= 0:
        return finish(a, ga, 0.0, "левая граница")
    if gb <= 0:
        return finish(b, gb, 0.0, "правая граница")
    x = a + (b-a)/2
    step = "середина"
    if method == "chord":
        x = a - ga*(b-a)/(gb-ga)
        step = "хорда"

    for k in range(1, max_iter+1):
        if not a < x < b:
            raise ArithmeticError("Пробная точка не различается с границей")
        gx = G(x)
        # Два независимых верхних ограничения ошибки аргумента.
        residual_bound = abs(gx)/mu
```

```
        interval_bound = max(x-a, b-x)
        error_bound = min(residual_bound, interval_bound)
        history.append({"k": k, "a": a, "b": b, "x": x,
                        "gx": gx, "bound": error_bound, "step": step})
        if error_bound <= eps_x:
            reason = ("производная" if residual_bound <= eps_x
                      else "интервал")
            return finish(x, gx, error_bound, reason)
        if gx < 0:
            a, ga = x, gx
        else:
            b, gb = x, gx

        if method == "chord":
            x = a - ga*(b-a)/(gb-ga)
            step = "хорда"
        elif method == "newton" and k % 5 != 0:
            hx = H(x)
            candidate = x - gx/hx if hx > 0 else float("inf")
            if isfinite(candidate) and a < candidate < b:
                x, step = candidate, "Ньютон"
            else:
                x, step = a + (b-a)/2, "защитная середина"
        else:
            x, step = a + (b-a)/2, "середина"
    raise RuntimeError("Достигнут предел итераций без заданной точности")
```

Аргумент mu задаётся по аналитическому доказательству. Программа проверяет его положительность, но не может доказать неравенство $f''(x) \geq \mu$ на непрерывном отрезке. Предел max_iter ограничивает вычисления; его достижение вызывает сообщение об ошибке, а не выдачу неподтверждённого оптимума.

История хранит границы до обновления и позволяет проверить сохранение минимума. Величина max(x-a, b-x) применима к любой внутренней точке; для середины она автоматически равна половине длины.

9.3 Запуск контрольного примера

```
METHODS = ("bisection", "chord", "newton")

if __name__ == "__main__":
    f = lambda x: x**4/4 + x*x/2 - x
    df = lambda x: x**3 + x - 1
    ddf = lambda x: 3*x*x + 1
    for eps in (1e-2, 1e-4, 1e-6):
        print(f"\neps_x = {eps:g}")
        for method in METHODS:
            r = minimize(f, df, ddf, 0, 2, 1, eps, method)
            print(f"{r.method:18} x={r.x:.10f} f={r.fx:.10f} "
                  f"E={r.bound:.3e} k={r.probes:3} "
                  f"Nf={r.nf} Ng={r.ng:3} Nh={r.nh:3}")
```

В терминале выполните python derivative_methods.py. Если в вашей системе интерпретатор вызывается как python3, используйте это имя. При самостоятельных изменениях сначала воспроизведите контрольные таблицы, а затем переходите к своему варианту.

10 Индивидуальные варианты

Для всех вариантов задаётся семейство

$$f(x) = \frac{q}{2}(x-h)^2 + \frac{r}{4}(x-h)^4 + c, \qquad x \in [a_0, b_0], \quad q > 0, \quad r > 0.$$

Производные имеют вид

$$f'(x) = q(x-h) + r(x-h)^3, \qquad f''(x) = q + 3r(x-h)^2 \geq q.$$

Следовательно, можно взять $\mu = q$. Поскольку $a_0 < h < b_0$, контрольный минимум равен $x^* = h, f(x^*) = c$. Известное решение нужно для проверки алгоритмов; в сам численный метод его передавать нельзя.

Вариант	q	r	h	c	a_0	b_0
1	0.5	0.1	-2	2	-3.4	0.3
2	1	0.25	-1.7	3	-3.3	0.9
3	2	0.5	-1.4	4	-2.6	1.5
4	3	1	-1.1	1	-2.5	0.9
5	1.5	0.2	-0.8	2	-2.4	1.5
6	2.5	0.4	-0.5	3	-1.7	2.1
7	0.5	0.1	-0.2	4	-1.6	2.7
8	1	0.25	0.1	1	-1.5	2.1
9	2	0.5	0.4	2	-0.8	2.7
10	3	1	0.7	3	-0.7	3.3
11	1.5	0.2	1	4	-0.6	3.9
12	2.5	0.4	1.3	1	0.1	3.3
13	0.5	0.1	1.6	2	0.2	3.9
14	1	0.25	1.9	3	0.3	4.5
15	2	0.5	2.2	4	1	5.1
16	3	1	2.5	1	1.1	4.5
17	1.5	0.2	2.8	2	1.2	5.1
18	2.5	0.4	3.1	3	1.9	5.7

10.1 Обязательная часть

1. Выписать функцию своего варианта и обе производные. Обосновать строгое возрастание g, единственность минимума и выбор μ.
2. Вычислить знаки производной на границах. Выполнить вручную первые три пробные точки деления пополам и первые две точки хорд.
3. Из начальной середины выполнить вручную два обычных шага Ньютона; на каждом шаге проверить попадание предлагаемой точки в текущий интервал.
4. Запустить три метода при $\varepsilon_x = 10^{-2}, 10^{-4}, 10^{-6}$.
5. Для каждого запуска проверить $|\hat{x} - h| \leq E \leq \varepsilon_x$ с учётом округления. Если минимум определён граничной проверкой в дополнительном опыте, использовать соответствующий конец отрезка.

6. Записать $\hat{x}$, $f(\hat{x})$, фактическую ошибку, E, k, N_f, N_g, N_h и причину остановки. Объяснить связь $N_g = k + 2$ для внутренних запусков данной программы.
7. Построить графики функции, производной и ошибки по пробным точкам. Сопоставить число обращений к первой и второй производным.
8. Сформулировать вывод по своим результатам: какой метод потребовал меньше вызовов, как повлиял допуск и понадобились ли защитные шаги.

10.2 Дополнительные исследования

Граничный минимум. Для $f(x) = (x + 1)^2$ на [0,2] проверить, что минимум находится в 0, а для $f(x) = (x - 3)^2$ — в 2. Объяснить отсутствие внутренних итераций.

Плоская функция. Для $f(x) = 10^{-8}(x - 1)^2/2$ на [0,3] сравнить правильный остаточный критерий с ошибочной проверкой только $|f'(x)| \le 10^{-6}$. Показать, почему второй критерий может допустить большую ошибку аргумента.

Неудачный старт Ньютона. Для $f(x) = \sqrt{1 + x^2}$ вручную выполнить два обычных шага из $x_0 = 2$. Затем объяснить, какие предложения отвергла бы проверка попадания в заданный интервал.

Стоимость вычислений. Ввести условную стоимость $C = N_g + \alpha N_h$ и сравнить методы при $\alpha = 1,5,10$. Это модель относительной трудоёмкости; фактическое время зависит также от языка, функции и среды выполнения.

11 Графики и оформление отчёта

В архиве находится run_experiments.py. Для графиков требуется Matplotlib. Установка: python -m pip install matplotlib. Запуск первого варианта: python run_experiments.py 1. Замените последний аргумент своим номером от 1 до 18.

Программа создаёт папку results_variant_01 с CSV-таблицей результатов, отдельными историями для допуска 10^{-6} и двумя рисунками. Имена столбцов CSV соответствуют обозначениям в методичке. Расчёты основного файла derivative_methods.py можно выполнять и без библиотеки графиков.

Отчёт должен содержать номер варианта, цель, постановку, формулы производных, обоснование предпосылок, ручные расчёты, код или приложенные программные файлы, результаты девяти запусков, графики и содержательный вывод. Для истории указывайте границы до вычисления новой точки, её координату, производную, способ выбора и оценку ошибки.

Не следует писать «метод сошёлся», опираясь только на красивый график или малое изменение соседних точек. Укажите сработавший критерий и численно подтвердите требуемую точность. При сравнении методов используйте одинаковую функцию, начальный интервал, допуск и смысл счётчиков.

12 Критерии оценивания

Элемент отчёта	Баллы
Постановка и доказательство условий применимости	15

Элемент отчёта	Баллы
Ручные шаги и обоснование критерия точности	20
Корректная реализация трёх методов	25
Численный эксперимент и проверка счётчиков	15
Графики и сравнительный вывод	15
Защита и ответы на вопросы	10
Итого	100

13 Контрольные вопросы

1. Когда условие $f'(x) = 0$ соответствует единственному минимуму?
2. Как обнаружить граничный минимум по знакам производной?
3. Что означает условие $f''(x) \geq \mu > 0$?
4. Как вывести оценку $|x - x^*| \leq |f'(x)|/\mu$?
5. Почему малое значение производной может сочетаться с большой ошибкой аргумента?
6. Чем допуск ε_x отличается от допуска на длину интервала?
7. Почему для произвольной точки нельзя автоматически использовать половину длины отрезка как оценку ошибки?
8. Как правило знаков сохраняет корень внутри отрезка?
9. Из какого уравнения получается точка хорды?
10. Почему в методе хорд один конец может почти не двигаться?
11. К какому графику проводится касательная в рассматриваемом методе Ньютона?
12. При каких предпосылках обычный метод Ньютона имеет квадратичную локальную сходимость?
13. Зачем в программе периодически выполняется шаг середины?
14. Почему число итераций и число вычислений производных следует указывать отдельно?
15. Какие предположения нарушаются при использовании неточной численной производной без оценки её ошибки?

Материалы для самостоятельного чтения

1. NIST Digital Library of Mathematical Functions. §3.8 Nonlinear Equations: формула Ньютона и локальная сходимость — §3.8(ii); деление пополам и regula falsi — §3.8(iii). https://dlmf.nist.gov/3.8. В справочнике уравнение записано для функции f; в настоящей работе методы применены к $g = f'$.

Постановка, вывод критериев точности, числовые примеры, индивидуальные варианты и Python-код подготовлены для этой лабораторной работы.