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

Дисциплина «Методы оптимизации»

ЛАБОРАТОРНАЯ РАБОТА 2

Методы последовательного поиска

Методы интервалов — методические указания, разобранные примеры и индивидуальные задания

Практическая тема 2 по рабочей программе. Объём — 5 академических часов. Работа продолжает лабораторную № 1. Основные новые методы — деление интервала пополам по трём пробным точкам и поиск Фибоначчи. Золотое сечение используется для сопоставления. Сначала изучите методику и ручные примеры, затем выполните программный эксперимент своего варианта.

1 Цель и результаты работы

Цель — научиться сохранять интервал, содержащий минимум, обосновывать каждое его сокращение и выбирать метод с учётом точности и числа вычислений функции. К концу работы студент должен уметь вывести оценки длины интервала, объяснить повторное использование значений функции, реализовать три метода и проверить результат независимым способом.

Предварительно нужны понятия унимодальности, абсолютной погрешности и алгоритмы из лабораторной № 1. Для основных вычислений достаточно стандартной библиотеки Python 3. Для графиков используется Matplotlib. Производные помогают проверить учебные функции, но внутри трёх алгоритмов поиска не вычисляются.

Рекомендуемое распределение 225 минут: 35 минут — методика; 40 — ручные примеры и оценки; 60 — реализация; 50 — сравнительный эксперимент; 40 — отчёт и защита. Часть программных заготовок можно изучить заранее.

2 Постановка задачи и условия применимости

На заданном отрезке требуется решить задачу:

$$f(x^*) = \min_{x \in [a_0, b_0]} f(x).$$

Для обоснования исключения интервалов будем считать функцию непрерывной и строго унимодальной: она строго убывает слева от единственной точки минимума и строго возрастает справа от неё. Минимум допускается на границе; тогда функция на всём отрезке монотонна в соответствующую сторону.

Для функции с несколькими локальными минимумами описанные методы не дают общей гарантии нахождения глобального минимума. График помогает увидеть форму функции, но сам по себе не доказывает унимодальность. В индивидуальном задании её необходимо обосновать аналитически.

2.1 Что именно означает точность

В этой работе ε задаёт **максимальную длину конечного интервала**, а приближённая точка выбирается посередине:

$$b_k - a_k \le \varepsilon, \qquad \hat{x} = \frac{a_k + b_k}{2}, \qquad |\hat{x} - x^*| \le \frac{\varepsilon}{2}.$$

Если требуется погрешность по аргументу не больше η, достаточно взять $\varepsilon = 2\eta$. Условие на аргумент не равнозначно условию $|f(\hat{x}) - f(x^*)| \le \varepsilon$: это разные величины и обычно разные единицы измерения.

Если минимум внутренний, $f'(x^*) = 0$, а $|f''(x)| \le M$ на рассматриваемом отрезке, то формула Тейлора даёт дополнительную оценку:

$$0 \le f(\hat{x}) - f(x^*) \le \frac{M}{2}|\hat{x} - x^*|^2 \le \frac{M\varepsilon^2}{8}.$$

Эта оценка требует перечисленных условий и не заменяет основной критерий длины интервала.

2.2 Правило исключения интервала

Пусть $a < x_1 < x_2 < b$. Если $f(x_1) < f(x_2)$, минимум не может находиться правее x_2: иначе обе пробные точки находились бы на убывающей ветви и выполнялось бы обратное неравенство. Значит, сохраняем $[a, x_2]$. Аналогично, при $f(x_1) > f(x_2)$ сохраняем $[x_1, b]$.

При равных значениях для строго унимодальной функции минимум лежит между x_1 и x_2. Можно оставить этот меньший отрезок. В программе золотого сечения и Фибоначчи для единообразия сохраняется более широкий $[a, x_2]$; он также содержит минимум.

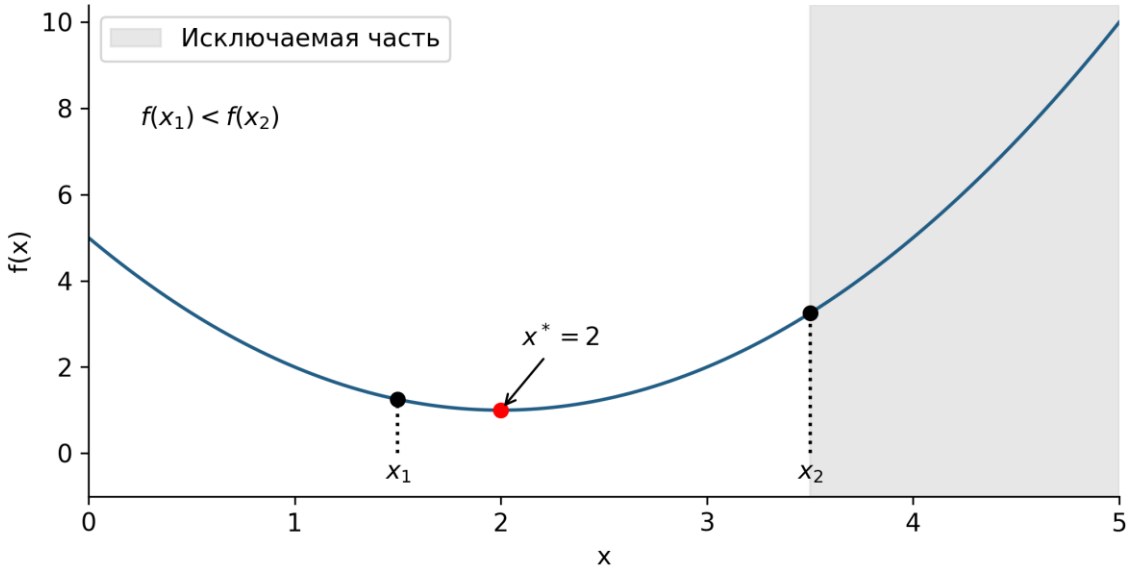


Рисунок 1 — Сравнение двух значений функции определяет допустимое сокращение интервала

3 Деление интервала пополам по трём точкам

3.1 Расположение пробных точек

На текущем отрезке длины $L = b - a$ зададим середину и четвертные точки:

$$m = \frac{a+b}{2}, \qquad x_1 = a + \frac{L}{4}, \qquad x_2 = b - \frac{L}{4}.$$

На первой итерации вычисляются $f(m), f(x_1), f(x_2)$. Затем значение в середине сохраняем и используем повторно: новая середина всегда совпадает с одной из уже проверенных точек.

3.2 Три возможных сокращения

Условие	Новый интервал	Его середина
$f(x_1) < f(m)$	$[a, m]$	x_1
$f(x_2) < f(m)$	$[m, b]$	x_2
Оба четвертных значения не меньше $f(m)$	$[x_1, x_2]$	m

Вторая строка проверяется, если первая не выполнена. Строгая унимодальность исключает ситуацию, когда обе четвертные точки ниже середины и минимум между ними отделён максимумом.

В третьем случае правило двух точек сначала исключает левую часть до x_1, затем правую после x_2. Во всех случаях длина уменьшается ровно вдвое:

$$L_{k+1} = \frac{L_k}{2}, \qquad L_k = \frac{L_0}{2^k}.$$

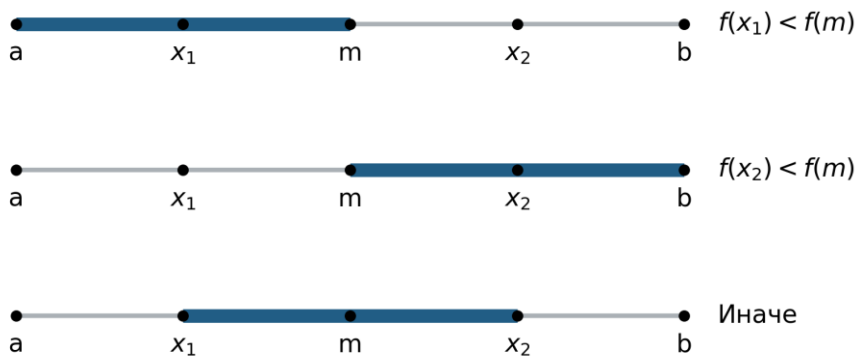


Рисунок 2 — Каждый из трёх сохраняемых отрезков имеет половину исходной длины

3.3 Число итераций и вызовов функции

При $L_0 > \varepsilon$ достаточно:

$$k = \left\lceil \log_2 \frac{L_0}{\varepsilon} \right\rceil.$$

В приведённой реализации однажды вычисляется начальная середина, а затем на каждом сокращении вычисляются две четвертные точки. Итоговое значение середины уже известно:

$$N_f = 1 + 2k.$$

Это деление интервала пополам отличается от дихотомии из работы № 1. В дихотомии сравниваются две близкие точки около середины, а длина после шага равна $L/2 + \delta$. Здесь используются четвертные точки и середина; сжатие составляет точно $1/2$ в точной арифметике.

4 Золотое сечение как метод сравнения

Пусть $\tau = \left(\sqrt{5} - 1\right)/2$. Пробные точки выбираются так:

$$x_1 = b - \tau(b - a), \qquad x_2 = a + \tau(b - a).$$

Условие $\tau^2 = 1 - \tau$ обеспечивает самоподобие расположения точек. После сокращения одно значение функции переносится на следующий шаг, а вычислить нужно только второе. Для длины интервала и числа сокращений получаем:

$$L_k = \tau^k L_0, \qquad k = \left\lceil \frac{\ln(L_0/\varepsilon)}{\ln(1/\tau)} \right\rceil.$$

В реализации этой работы после достижения допуска новая внутренняя точка уже не вычисляется. При $k \geq 1$ используются два начальных вызова, $k - 1$ новых внутренних вызовов и один вызов для конечной середины:

$$N_f = k + 2.$$

Поэтому счётчик может отличаться от простой реализации из лабораторной № 1, которая вычисляла ещё одну пробную точку перед выходом из цикла. Для сравнения методов важно считать реальные вызовы одинаковым способом.

5 Метод Фибоначчи

5.1 Последовательность и смысл заранее выбранного числа шагов

Числа Фибоначчи определим индексами:

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2} \quad (n \geq 2).$$

Начало последовательности: 0,1,1,2,3,5,8,13,21,34,55,89, … . Метод использует отношения соседних чисел, чтобы согласовать положение пробных точек с оставшимся числом шагов. В отличие от постоянного коэффициента золотого сечения эти отношения меняются по мере приближения к завершению поиска.

5.2 Выбор индекса и первых точек

В программе последний вырожденный шаг обрабатывается с небольшим положительным смещением. Выбираем $\delta = \varepsilon/100$, затем наименьший индекс $n \geq 4$, для которого:

$$F_n \geq \frac{L_0}{\varepsilon - \delta}.$$

Первые пробные точки:

$$x_1 = a_0 + \frac{F_{n-2}}{F_n} L_0, \qquad x_2 = a_0 + \frac{F_{n-1}}{F_n} L_0.$$

Такой выбор обеспечивает место для последнего смещения и итоговую длину не больше ε в точной арифметике.

5.3 Почему можно повторно использовать одно значение

Рассмотрим ветвь $f(x_1) \le f(x_2)$. Новый отрезок — $[a, x_2]$ длины $F_{r-1}L/F_r$. Старая левая пробная точка занимает в нём долю:

$$\frac{x_1 - a}{x_2 - a} = \frac{F_{r-2}}{F_{r-1}}.$$

Это как раз положение правой внутренней точки для следующего индекса $r-1$. Её значение функции сохраняется. Новая левая точка использует долю F_{r-3}/F_{r-1}. Для другой ветви рассуждение симметрично.

После k обычных сокращений длина равна:

$$L_k = \frac{F_{n-k}}{F_n} L_0, \qquad 0 \le k \le n-3.$$

5.4 Последний шаг без совпадающих точек

Когда оставшийся индекс равен 3, формальные внутренние точки совпали бы: $F_1/F_3 = F_2/F_3 = 1/2$. Повторное вычисление в одной точке не даёт новой информации.

К этому моменту длина отрезка равна $2L_0/F_n$, а значение в его середине m уже известно. Вычислим функцию в новой точке $m + \delta$. Если $f(m) \le f(m+\delta)$, сохраняем $[a, m+\delta]$; иначе — $[m, b]$. Тогда:

$$L_{\text{кон}} \le \frac{L_0}{F_n} + \delta \le \varepsilon.$$

Программа проверяет, что $m < m + \delta < b$. Если числа не различаются из-за округления, требуется ослабить допуск или изменить масштаб переменной.

5.5 Как считать затраты

Для выбранной реализации при $L_0 > \varepsilon$ обычных сокращений $n-3$, затем выполняется один заключительный шаг. Поэтому $k = n-2$. Поиск использует $n-1$ вычислений функции, а вычисление в конечной середине добавляет ещё одно:

$$N_f = n.$$

Это оценка именно приведённой программы с указанной индексацией и заключительным смещением. Сравнивать формулы из разных учебников можно только после согласования индексов и правила подсчёта вызовов.

Для больших индексов F_{r-1}/F_r приближается к τ. Поэтому поведение Фибоначчи и золотого сечения близко. Нельзя делать вывод, что на любой конкретной функции один метод всегда даёт меньшую фактическую ошибку: оценка гарантирует размер конечного интервала, а положение минимума внутри него различается.

6 Ручной разбор примера

Исследуем $f(x) = (x-2)^2 + 1$ на $[0,5]$. Производная $f'(x) = 2(x-2)$ меняет знак с минуса на плюс при $x = 2$, поэтому минимум единственный, $x^* = 2$ и $f^* = 1$.

6.1 Первые шаги деления пополам

На первом шаге $m = 2{,}5$, $x_1 = 1{,}25$, $x_2 = 3{,}75$. Значения функции: 1,25, 1,5625 и 4,0625 соответственно. Оба четвертных значения выше значения в середине; сохраняем [1,25,3,75].

На втором шаге новая середина остаётся 2,5. Четвертные точки равны 1,875 и 3,125, значения функции — 1,015625 и 2,265625. Левая точка лучше середины, поэтому остаётся [1,25,2,5].

k	a_k	b_k	L_k	**Середина**
0	0.000000	5.000000	5.000000	2.500000
1	1.250000	3.750000	2.500000	2.500000
2	1.250000	2.500000	1.250000	1.875000
3	1.562500	2.187500	0.625000	1.875000
4	1.875000	2.187500	0.312500	2.031250

В таблице записаны границы после сокращения. История с индексом 0 содержит исходный интервал. Для $\varepsilon = 0{,}01$ требуется $\lceil \log_2 500 \rceil = 9$ сокращений и 19 вызовов функции.

6.2 Начало и завершение поиска Фибоначчи

Для наглядного ручного расчёта возьмём $\varepsilon = 0{,}25$, тогда $\delta = 0{,}0025$. Требуется $F_n \geq 5/0{,}2475 \approx 20{,}202$. Подходит $F_8 = 21$.

$$x_1 = \frac{8}{21} \cdot 5 = \frac{40}{21} \approx 1{,}904762, \qquad x_2 = \frac{13}{21} \cdot 5 = \frac{65}{21} \approx 3{,}095238.$$

Здесь $f(x_1) = 1 + 4/441$, $f(x_2) = 1 + 529/441$, поэтому сохраняется $[0{,}65/21]$. Старая точка $40/21$ становится правой внутренней. Новая левая точка равна $25/21$. Следующее сравнение сохраняет $[25/21{,}65/21]$.

Этап	a	b	$b - a$
Исходный	0.000000	5.000000	5.000000
1	0.000000	3.095238	3.095238
2	1.190476	3.095238	1.904762
3	1.190476	2.380952	1.190476
4	1.666667	2.380952	0.714286
5	1.666667	2.142857	0.476190
Заключительный	1.904762	2.142857	0.238095

После пяти обычных сокращений остаётся отрезок длины $10/21$, а после заключительного шага длина не превосходит $5/21 + 0{,}0025 \approx 0{,}240595 < 0{,}25$. Пример показывает, зачем заранее оставлять запас под последнее смещение.

7 Проверенный вычислительный эксперимент

В таблице ниже все методы используют одну функцию, один исходный отрезок и одно понимание точности. N_f включает начальные вызовы и вычисление значения в выдаваемой точке, если оно ещё не было известно.

Метод	ε	$\hat{x}$	$f(\hat{x})$	k	N_f	L_k	Ошибка
Деление пополам	1e-02	2.00195312	1.0000038147	9	19	9.77e-03	1.95e-03
Золотое сечение	1e-02	2.00186590	1.0000034816	13	15	9.60e-03	1.87e-03
Фибоначчи	1e-02	1.99595164	1.0000163892	13	15	8.30e-03	4.05e-03
Деление пополам	1e-04	2.00000763	1.0000000001	16	33	7.63e-05	7.63e-06
Золотое сечение	1e-04	1.99998483	1.0000000002	23	25	7.80e-05	1.52e-05
Фибоначчи	1e-04	1.99996718	1.0000000011	23	25	6.76e-05	3.28e-05
Деление пополам	1e-06	1.99999988	1.0000000000	23	47	5.96e-07	1.19e-07
Золотое сечение	1e-06	2.00000012	1.0000000000	33	35	6.34e-07	1.23e-07
Фибоначчи	1e-06	1.99999974	1.0000000000	32	34	8.87e-07	2.58e-07

В данном примере деление пополам требует меньше сокращений, но больше вызовов функции. При $\varepsilon = 10^{-6}$ золотое сечение использует 35 вызовов, Фибоначчи — 34, деление пополам — 47. При этом меньшая фактическая ошибка по аргументу не обязана соответствовать меньшему числу вызовов.

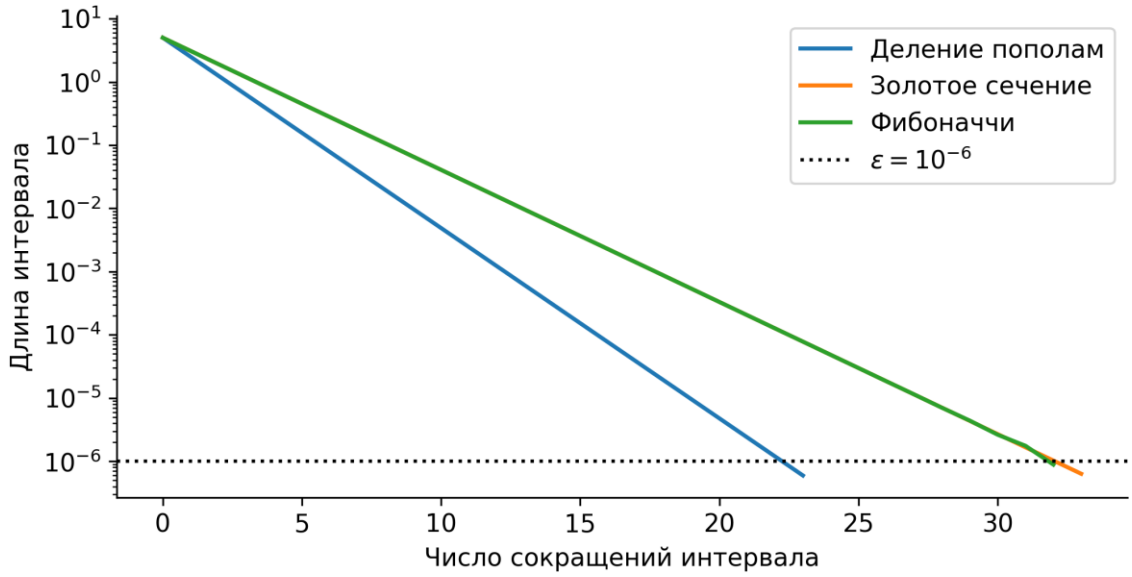


Рисунок 3 — Сходимость трёх методов при одинаковом допуске

На графике использован логарифмический масштаб по вертикали. Сравнение по номеру итерации показывает скорость сжатия интервала; для оценки вычислительной стоимости нужно дополнительно рассмотреть N_f.

8 Программа на Python

Сохраните блоки разделов 8.1–8.5 последовательно в один файл interval_methods.py. Готовый файл также включён в архив с кодом. Каждая функция возвращает найденную точку, значение функции, конечный интервал, число сокращений, число вызовов и историю границ.

8.1 Общие структуры и проверка входных данных

```
from dataclasses import dataclass
from math import isfinite, sqrt


@dataclass
class Result:
    method: str
    x: float
    fx: float
    a: float
    b: float
    iterations: int
    nfev: int
    history: list


class Oracle:
    def __init__(self, f):
        self.f, self.nfev = f, 0

    def __call__(self, x):
        self.nfev += 1
        value = float(self.f(x))
        if not isfinite(value):
            raise ValueError("Функция вернула нечисловое значение")
        return value


def prepare(f, a, b, eps):
    a, b, eps = float(a), float(b), float(eps)
    if not all(isfinite(v) for v in (a, b, eps)):
        raise ValueError("Границы и eps должны быть конечными")
    if not a < b or eps <= 0:
        raise ValueError("Требуются a < b и eps > 0")
    if eps < 100 * max(abs(a), abs(b), 1.0) * 2.22e-16:
        raise ValueError("Допуск слишком мал для данных границ")
    return Oracle(f), a, b, eps


def answer(name, fun, a, b, history, fx=None):
    x = (a + b) / 2
    if fx is None:
        fx = fun(x)
    return Result(name, x, fx, a, b, len(history)-1,
                  fun.nfev, history)
```

Счётчик увеличивается при каждом фактическом обращении к целевой функции. Повторное использование уже сохранённого значения не считается новым вызовом. История содержит исходный интервал, поэтому число сокращений равно её длине минус один.

8.2 Деление интервала пополам

```
def interval_halving(f, a, b, eps=1e-4):
    fun, a, b, eps = prepare(f, a, b, eps)
```

```
    history = [(a, b)]
    middle = (a + b) / 2
    fm = fun(middle)
    while b - a > eps:
        x1, x2 = (a + middle)/2, (middle + b)/2
        f1, f2 = fun(x1), fun(x2)
        if f1 < fm:
            b, middle, fm = middle, x1, f1
        elif f2 < fm:
            a, middle, fm = middle, x2, f2
        else:
            a, b = x1, x2
        history.append((a, b))
    return answer("Деление пополам", fun, a, b, history, fm)
```

Переменная fm хранит значение в текущей середине. В каждой ветви обновляются одновременно границы, середина и её значение. Несогласованное обновление этих переменных — частая причина неверных результатов.

8.3 Золотое сечение

```
def golden_section(f, a, b, eps=1e-4):
    fun, a, b, eps = prepare(f, a, b, eps)
    history = [(a, b)]
    if b - a <= eps:
        return answer("Золотое сечение", fun, a, b, history)
    tau = (sqrt(5) - 1) / 2
    x1, x2 = b - tau*(b-a), a + tau*(b-a)
    f1, f2 = fun(x1), fun(x2)
    while b - a > eps:
        if f1 <= f2:
            b, x2, f2 = x2, x1, f1
            history.append((a, b))
            if b - a <= eps:
                break
            x1 = b - tau*(b-a)
            f1 = fun(x1)
        else:
            a, x1, f1 = x1, x2, f2
            history.append((a, b))
            if b - a <= eps:
                break
            x2 = a + tau*(b-a)
            f2 = fun(x2)
    return answer("Золотое сечение", fun, a, b, history)
```

Проверка длины после сокращения предотвращает лишнее вычисление новой внутренней точки. Условия сравнения допускают равенство значений и сохраняют интервал, содержащий минимум.

8.4 Фибоначчи

```
def fibonacci_search(f, a, b, eps=1e-4):
    fun, a, b, eps = prepare(f, a, b, eps)
    history = [(a, b)]
    if b - a <= eps:
        return answer("Фибоначчи", fun, a, b, history)
    delta = eps / 100
    fib = [0, 1]
    target = (b - a) / (eps - delta)
    while len(fib) <= 4 or fib[-1] < target:
        fib.append(fib[-1] + fib[-2])
    r = len(fib) - 1
```

```
    x1 = a + fib[r-2]/fib[r]*(b-a)
    x2 = a + fib[r-1]/fib[r]*(b-a)
    f1, f2 = fun(x1), fun(x2)
    while r > 3:
        if f1 <= f2:
            b, x2, f2 = x2, x1, f1
            r -= 1
            history.append((a, b))
            if r == 3:
                middle, fm = x2, f2
                break
            x1 = a + fib[r-2]/fib[r]*(b-a)
            f1 = fun(x1)
        else:
            a, x1, f1 = x1, x2, f2
            r -= 1
            history.append((a, b))
            if r == 3:
                middle, fm = x1, f1
                break
            x2 = a + fib[r-1]/fib[r]*(b-a)
            f2 = fun(x2)
    # При r=3 формальные внутренние точки совпали бы.
    probe = middle + delta
    if not middle < probe < b:
        raise ArithmeticError("Не различаются точки последнего шага")
    if fm <= fun(probe):
        b = probe
    else:
        a = middle
    history.append((a, b))
    return answer("Фибоначчи", fun, a, b, history)
```

При переходе между ветвями уменьшается оставшийся индекс r, а не начальный индекс задачи. На последнем шаге сравниваются разные точки. Замена заключительного шага сравнением точки с самой собой недопустима.

8.5 Запуск контрольного примера

```
METHODS = (interval_halving, golden_section, fibonacci_search)

if __name__ == "__main__":
    f = lambda x: (x - 2)**2 + 1
    for eps in (1e-2, 1e-4, 1e-6):
        print(f"\neps = {eps:g}")
        for method in METHODS:
            r = method(f, 0, 5, eps)
            print(f"{r.method:18} x={r.x:.8f} f={r.fx:.10f} "
                  f"k={r.iterations:2} Nf={r.nfev:2} "
                  f"L={r.b-r.a:.3e}")
```

Для запуска из терминала используйте python interval_methods.py. В среде с командой python3 имя интерпретатора заменяется соответственно. Основной расчёт не требует NumPy, SciPy или других дополнительных пакетов.

9 Графики и сохранение результатов

Для отчёта подготовьте график функции с отмеченным минимумом и график длины интервала по итерациям. Дополнительно полезно сравнить число вызовов функции при нескольких допусках. Готовый файл run_experiments.py создаёт CSV-таблицу и рисунки для выбранного варианта.

Краткий пример построения графика сходимости:

```
import matplotlib.pyplot as plt
from interval_methods import METHODS

f = lambda x: (x - 2)**2 + 1
for method in METHODS:
    result = method(f, 0, 5, 1e-6)
    lengths = [b-a for a, b in result.history]
    plt.semilogy(range(len(lengths)), lengths,
                 label=result.method)
plt.xlabel("Число сокращений интервала")
plt.ylabel("Длина интервала")
plt.legend()
plt.tight_layout()
plt.savefig("convergence.png", dpi=300)
plt.show()
```

Для установки библиотеки графиков используется команда python -m pip install matplotlib. Численные алгоритмы можно проверить и без графического пакета.

10 Индивидуальные задания

Каждому варианту соответствует функция:

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

Параметры подобраны так, что $q > 0$, $r \geq 0$ и $a_0 < h < b_0$. Обоснуйте единственность минимума самостоятельно, используя производные:

$$f'(x) = 2q(x-h) + 4r(x-h)^3, \qquad f''(x) = 2q + 12r(x-h)^2 > 0.$$

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	0.05	-1.4	4	-2.6	1.5
4	3	0.2	-1.1	1	-2.5	0.9
5	1.5	0.15	-0.8	2	-2.4	1.5
6	2.5	0.3	-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	0.05	0.4	2	-0.8	2.7
10	3	0.2	0.7	3	-0.7	3.3
11	1.5	0.15	1	4	-0.6	3.9
12	2.5	0.3	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	0.05	2.2	4	1	5.1

Вариант	q	r	h	c	a_0	b_0
16	3	0.2	2.5	1	1.1	4.5
17	1.5	0.15	2.8	2	1.2	5.1
18	2.5	0.3	3.1	3	1.9	5.7

10.1 Обязательная часть

1. Выписать функцию и границы своего варианта. Доказать унимодальность и найти точное контрольное решение.
2. Выполнить вручную первые три сокращения деления пополам. Сохранить все пробные точки и сравнения.
3. Для Фибоначчи выбрать $\varepsilon = 0{,}1$, вычислить индекс n и выполнить первые два обычных сокращения.
4. Запустить все три метода при $\varepsilon = 10^{-2}, 10^{-4}, 10^{-6}$.
5. Для каждого запуска проверить попадание точного минимума в конечный интервал и оценку ошибки середины.
6. Сопоставить число сокращений и вызовов с выведенными формулами. Объяснить все расхождения, связанные с округлением или изменением реализации.
7. Построить графики функции и сходимости; для $\varepsilon = 10^{-2}, 10^{-3}, \dots, 10^{-6}$ построить зависимость N_f от требуемой точности.
8. Сформулировать вывод по собственным числам. Указать, какой метод экономичнее по вызовам, а какой быстрее сокращает интервал по итерациям.

10.2 Дополнительное исследование

Для исследования машинного округления повторите расчёт при $\varepsilon = 10^{-7}$. Вблизи минимума различные значения аргумента могут давать одинаковое вычисленное значение функции. Поэтому теоретическое сохранение минимума в интервале, доказанное для точных сравнений, в машинной арифметике может нарушиться. Зафиксируйте такой случай, если он возникнет. Сравните поиск для исходной функции и для $\tilde{f}(x) = q(x-h)^2 + r(x-h)^4$, вычисляя последнюю непосредственно, без прибавления и последующего вычитания c. Минимизатор этих функций одинаков, но влияние округления различается.

Исследуйте функцию $g(x) = x^2 + e^{-x}$ на $[-2{,}3]$. Поскольку $g''(x) = 2 + e^{-x} > 0$, минимум единственный. Для контрольной точки можно решить уравнение $2x = e^{-x}$ отдельным методом с существенно более строгим допуском. Этот контрольный расчёт не включайте в счётчик N_f исследуемых методов.

Второй опыт — поиск минимума $g(x) = x$ на $[0{,}5]$. Объясните, почему середина конечного интервала лишь приближает граничный минимум $x^* = 0$, хотя сам минимум находится на конце отрезка.

Для исследования границ применимости выберите функцию с двумя минимумами и покажите, что разные исходные интервалы могут привести к разным результатам. В этом опыте гарантии, доказанные для строго унимодальной функции, уже не применяются.

11 Форма отчёта

Отчёт должен содержать номер варианта, цель работы, математическую постановку, доказательство унимодальности, ручные шаги, описание алгоритмов, листинг или приложенные файлы программы, таблицу результатов, графики, сравнительный вывод и ответы на контрольные вопросы.

Метод	ε	$\hat{x}$	$f(\hat{x})$	k	N_f	$b_k - a_k$	$\lvert\hat{x} - x^*\rvert$
Деление пополам							
Золотое сечение							
Фибоначчи							

Заполните эту схему для каждого допуска. В отдельных таблицах ручного расчёта указывайте границы до шага, пробные точки, значения функции, выбранную ветвь и границы после шага. Не смешивайте данные до сокращения и после него в одной строке без пояснения.

12 Критерии оценивания

Элемент работы	Баллы
Постановка и обоснование унимодальности	15
Ручные шаги и вывод оценок	20
Корректность трёх реализаций	25
Эксперимент и счётчики вызовов	15
Графики и содержательный сравнительный вывод	15
Защита и ответы на вопросы	10
Итого	100

13 Контрольные вопросы

1. Как правило двух точек следует из строгой унимодальности?
2. Почему график функции не является достаточным доказательством единственности минимума?
3. Как длина конечного интервала связана с ошибкой его середины?
4. Чем деление пополам по трём точкам отличается от дихотомии?
5. Почему новая середина в методе деления пополам уже имеет известное значение функции?
6. Откуда появляется соотношение $\tau^2 = 1 - \tau$?
7. Как вывести формулу $L_k = F_{n-k}L_0/F_n$ для метода Фибоначчи?
8. Что происходит с пробными точками при оставшемся индексе 3?
9. Зачем при выборе n учитывать заключительное смещение δ?
10. Почему результаты по числу вызовов зависят от того, включено ли вычисление функции в выдаваемой точке?

11. Можно ли утверждать, что меньшее N_f обязательно означает меньшую фактическую ошибку?
12. Как изменится постановка для максимизации функции?
13. Какие признаки в истории вычислений указывают на ошибку обновления интервала?
14. Почему чрезмерное уменьшение допуска может привести к совпадению различных теоретических точек в машинной арифметике?

14 Проверка перед защитой

Убедитесь, что все методы работали на одной функции и одном отрезке, а допуск означал одно и то же. В конце каждого запуска должны выполняться условия $b_k - a_k \leq \varepsilon$ и $x^* \in [a_k, b_k]$ с учётом машинного округления. Число вызовов берётся из программы, графики имеют подписи осей, а вывод объясняет различие между итерациями, вызовами и фактической ошибкой.

Литература для самостоятельного изучения

1. Струченков В. И. Методы оптимизации в прикладных задачах. — М.: СОЛОН-ПРЕСС, 2012. — 320 с. Основная литература силлабуса.
2. Казанская О. В., Юн С. Г., Альсова О. К. Модели и методы оптимизации. Практикум. — Новосибирск: НГТУ, 2012. — 204 с. Основная литература силлабуса.
3. Черняк А. А., Черняк Ж. А., Метельский Ю. М., Богданович С. А. Методы оптимизации: теория и алгоритмы. — 2-е изд. — М.: Юрайт, 2023. — 357 с. Основная литература силлабуса.
4. Kochenderfer M. J., Wheeler T. A. Algorithms for Optimization. — MIT Press, 2019. Первое издание доступно на официальной странице авторов: https://algorithmsbook.com/optimization/. При сопоставлении алгоритмов учитывайте принятые в данной работе индексы и правила подсчёта вызовов.