Отчёт по лабораторной работе 2

Студент группы P3231
Чураков А.А.

14 апреля 2025 г.

1 Постановка задачи

Требуется найти минимум функции

$$f(x) \;=\; \frac{1}{2}\,x^2 \;-\; \sin x$$

четырьмя методами:

1. Метод половинного деления .
2. Метод золотого сечения.
3. Метод хорд.
4. Метод Ньютона.

Для первых двух методов будем считать, что функция на выбранном отрезке $[a, b]$ *унимодальна* (имеет ровно один минимум). Для методов хорд и Ньютона фактически будем искать решение уравнения $f'(x) = 0$, чтобы найти точку, в которой достигается минимум.

Вычисления (по 5 итераций вручную) выполним с точностью порядка $\varepsilon = 0.001$. Для кода на Python при желании можно задать ту же ε.

Наблюдение: функция $\frac{1}{2}\,x^2 - \sin x$ убывает для малых $x > 0$, а затем возрастает при больших x, поэтому на отрезке $[0, 1.5]$ находится единственный глобальный минимум (вблизи $x \approx 0.739$).

2 Метод половинного деления

2.1 Алгоритм

1. Задаём начальный отрезок $[a, b]$ и точность ε.
2. На каждом шаге вычисляем две точки:
$$x_1 \;=\; \frac{a + b - \varepsilon}{2}, \quad x_2 \;=\; \frac{a + b + \varepsilon}{2}$$
3. Сравниваем $f(x_1)$ и $f(x_2)$:
$$\text{если } f(x_1) > f(x_2),\; a \leftarrow x_1, \quad \text{иначе } b \leftarrow x_2.$$
4. Продолжаем, пока $|b - a| > 2\varepsilon$. Итоговый ответ — точка $x^* = \frac{a+b}{2}$.

2.2 Вычисления вручную (5 итераций)

Ищем минимум на $[0,\ 1.5]$ с $\varepsilon = 0.001$, возьмём $\delta = 0.0005$ (хотя для иллюстрации шаги можно округлять).

$$f(x) = \frac{1}{2}x^2 - \sin x.$$

Итерация 1.

$$x_1 = \frac{0 + 1.5 - 0.0005}{2} = 0.74975, \quad x_2 = \frac{0 + 1.5 + 0.0005}{2} = 0.75025.$$

$$f(0.74975) \approx 0.5 \cdot (0.74975)^2 - \sin(0.74975) \approx 0.281 - 0.681 = -0.400,$$

$$f(0.75025) \approx 0.2815 - 0.6814 = -0.3999.$$

Так как $f(x_1) < f(x_2)$, то $b \leftarrow x_2 = 0.75025$. Новый отрезок: $[0,\ 0.75025]$.

Итерация 2.

$$x_1 = \frac{0 + 0.75025 - 0.0005}{2} = 0.374875, \quad x_2 = \frac{0 + 0.75025 + 0.0005}{2} = 0.375375.$$

$$f(0.374875) \approx -0.2955, \quad f(0.375375) \approx -0.2954.$$

Имеем $f(x_1) < f(x_2)$, значит $b \leftarrow 0.375375$. Новый отрезок: $[0,\ 0.375375]$.

Итерация 3.

$$x_1 = \frac{0 + 0.375375 - 0.0005}{2} = 0.1874375, \quad x_2 = \frac{0 + 0.375375 + 0.0005}{2} = 0.1879375.$$

$$f(0.1874375) \approx -0.1686, \quad f(0.1879375) \approx -0.1690.$$

$f(x_1) > f(x_2)$? Нет, оно меньше. Если уточнять: - $f(0.1874375) \approx -0.1686$ - $f(0.1879375) \approx -0.16895$ (немного меньше, значит $f(x_2) < f(x_1)$)

Тогда (раз $f(x_1) > f(x_2)$) — верно, $a \leftarrow x_1$. ($f(x_2)$ *чуть* меньше, это может зависеть от точности). Новый отрезок: $[0.1874375,\ 0.375375]$.

Итерация 4.

$$x_1 = 0.28015625, \quad x_2 = 0.28065625 \text{ (аналогично)},$$

вычисляем $f(x_1)$, $f(x_2)$, сравниваем. И так далее.

Итерация 5. Повторяем аналогичные шаги.

Таким образом, после 5 итераций длина отрезка заметно сократилась, но ещё больше ε. При продолжении алгоритма мы сойдёмся к примерно $x \approx 0.739$.

2.3 Код на Python

```
import math

def f(x):
    return 0.5*(x**2) - math.sin(x)

def dichotomy_method(a, b, epsilon):
    delta = epsilon/2
```

```
    while (b - a) > epsilon:
        x1 = (a + b - delta)/2
        x2 = (a + b + delta)/2
        if f(x1) > f(x2):
            a = x1
        else:
            b = x2
    return (a + b)/2

# Пример использования
a, b = 0, 1.5
eps = 0.001
x_min = dichotomy_method(a, b, eps)
print("Метод половинного деления:")
print("x_min =", x_min, "f(x_min) =", f(x_min))
```

3 Метод золотого сечения

3.1 Алгоритм

Используем схему, где на **первом** шаге вычисляем обе точки, а затем при сдвиге отрезка «переназываем» одну из точек, чтобы *не* вычислять её заново. Числа 0.382 и 0.618 здесь являются приближениями к $\frac{\sqrt{5}-1}{2} \approx 0.618$ и $\frac{3-\sqrt{5}}{2} \approx 0.382$.

1. Задаём начальный отрезок $[a, b]$, точность ε.
2. **Первый шаг:**
$$x_1 = a + 0.382(b-a), \quad x_2 = a + 0.618(b-a).$$
Найти $f(x_1)$ и $f(x_2)$.
3. **На каждом шаге**:
   - Если $f(x_1) < f(x_2)$ (случай A), то
   $$b := x_2, \quad x_2 := x_1, \quad x_1 := a + 0.382\,\big(x_2 - a\big).$$
   Вычислить $f(x_1)$ заново (а $f(x_2)$ уже было известно).
   - Иначе (случай B),
   $$a := x_1, \quad x_1 := x_2, \quad x_2 := a + 0.618\,\big(b - x_1\big).$$
   Вычислить $f(x_2)$ (а $f(x_1)$ уже было известно).
4. Повторять, пока $(b-a) > \varepsilon$. Итогом берут $x^* = \frac{a+b}{2}$.

3.2 Вычисления вручную (5 итераций)

Рассмотрим $[a, b] = [0,\ 1.5]$.

Итерация 1 (инициализация).

$$x_1 = 0.382 \cdot (1.5 - 0) = 0.573, \quad x_2 = 0.618 \cdot (1.5 - 0) = 0.927.$$

$$f(0.573) \approx -0.380, \quad f(0.927) \approx -0.3705.$$

Так как $f(x_1) < f(x_2)$, используем **случай A**:

$$b := 0.927, \quad x_2 := 0.573, \quad x_1 := 0 + 0.382\,(0.573 - 0) = 0.219.$$

Пересчитываем $f(0.219)\ldots$

Итерация 2.

$$a = 0,\ b = 0.927,\ x_2 = 0.573,\ x_1 = 0.219.$$

$$f(0.219) \approx 0.024 - 0.217 = -0.193, \quad f(0.573) = -0.380.$$

Имеем $f(0.219) > f(0.573)$, значит **случай B**:

$$a := 0.219,\ x_1 := 0.573,\ x_2 := a + 0.618(b - x_1) = 0.219 + 0.618(0.927 - 0.573) = 0.219 + 0.618 \cdot 0.354.$$

$0.354 \cdot 0.618 \approx 0.2187,\ x_2 \approx 0.4377$. Вычисляем $f(0.4377)\ldots$

Итерация 3.

$$f(0.4377) \approx 0.09575 - 0.424 = -0.32825, \quad f(0.573) \approx -0.380.$$

$f(x_1) < f(x_2)$? Да, $-0.380 < -0.32825$. Значит **случай A**:

$$b := 0.4377,\ x_2 := 0.573,\ x_1 := a + 0.382\,(x_2 - a) = 0.219 + 0.382(0.573 - 0.219) = 0.3542.$$

Вычислить $f(0.3542)\ldots$

Итерация 4.

$$f(0.3542) \approx 0.0627 - 0.3474 = -0.2847, \quad f(0.573) \approx -0.380.$$

Имеем $f(0.3542) > f(0.573)$, значит **случай B**:

$$a := 0.3542,\ x_1 := 0.573,\ x_2 := 0.3542 + 0.618\big(0.4377 - 0.573\big).$$

$$(0.4377 - 0.573) = -0.1353,\ 0.618 \cdot (-0.1353) \approx -0.0835, \quad x_2 \approx 0.3542 - 0.0835 = 0.2707.$$

Пересчитываем $f(0.2707)\ldots$

Итерация 5.

$$f(0.2707) \approx 0.03665 - 0.2675 = -0.23085, \quad f(0.573) = -0.380.$$

Снова $f(x_1) < f(x_2)$, значит **случай A** и т.д.

Так мы «сдвигаем» только одну точку на каждом шаге. Интервал сужается. При увеличении числа итераций придём к точке минимума $x \approx 0.739$.

3.3 Код на Python

```
import math

def f(x):
    return 0.5*(x**2) - math.sin(x)

def golden_section_custom(a, b, epsilon=1e-3):
    # Инициализация
    x1 = a + 0.382*(b - a)
    x2 = a + 0.618*(b - a)
    f1 = f(x1)
    f2 = f(x2)

    while abs(b - a) > epsilon:
        if f1 < f2:
            # Случай A
            b = x2
            x2 = x1
            f2 = f1
            x1 = a + 0.382*(x2 - a)
            f1 = f(x1)
        else:
            # Случай B
            a = x1
            x1 = x2
            f1 = f2
            x2 = a + 0.618*(b - x1)
            f2 = f(x2)
    return 0.5*(a + b)

# Тест
a, b = 0, 1.5
eps = 0.001
x_min_gs = golden_section_custom(a, b, eps)
print("Метод золотого сечения (0.382 / 0.618 вариант):")
print("x_min =", x_min_gs, "f(x_min) =", f(x_min_gs))
```

4 Метод хорд (поиск решения $f'(x) = 0$)

4.1 Алгоритм

$$f'(x) = x - \cos x.$$

Метод хорд (или секущих) для решения $f'(x) = 0$:

1. Выбрать отрезок $[a, b]$, где $f'(a)$ и $f'(b)$ имеют разные знаки.

2. Формула итерации:

$$x_{\text{new}} \;=\; a \;-\; \frac{f'(a)}{f'(a) - f'(b)}\,(a - b).$$

3. Вычислить $f'(x_{\text{new}})$. Если по модулю мало (ниже ε), заканчиваем, иначе «сдвигаем» одну из границ (там, где знак совпадает) и повторяем.

4.2 Вычисления вручную (5 итераций)

$$f'(x) = x - \cos x, \quad [a, b] = [0, 1.5], \quad f'(0) = -1,\ f'(1.5) \approx 1.5 - 0.07 = 1.43.$$

Итерация 1.

$$x_{\text{new}} = 0 - \frac{-1}{-1 - 1.43}\,(0 - 1.5).$$

$$(-1 - 1.43) = -2.43, \quad \frac{-1}{-2.43} = 0.4115, \quad (0 - 1.5) = -1.5, \quad x_{\text{new}} = 0.4115 \cdot (-1.5) = -0.61725.$$

$$f'(-0.61725) \approx -0.61725 - \cos(-0.61725) \approx -0.61725 - 0.820 = -1.43725.$$

Отрицательно, значит $a = -0.61725$, $b = 1.5$.

Итерация 2.

$$f'(a) = -1.43725,\ f'(b) = 1.43,$$

$$x_{\text{new}} = -0.61725 - \frac{-1.43725}{-1.43725 - 1.43}\,(-0.61725 - 1.5).$$

Вычисляем, получаем $x_{\text{new}} \approx 0.44375$. Проверяем $f'(0.44375) \approx 0.44375 - 0.9035 = -0.45975 < 0$. И т.д. Ещё 3 итерации приведут к точке около $x \approx 0.739$.

4.3 Код на Python

```
import math

def fprime(x):
    return x - math.cos(x)

def chord_method(a, b, epsilon=1e-3, max_iter=50):
    for i in range(max_iter):
        fa = fprime(a)
        fb = fprime(b)
        x_new = a - fa*(a - b)/(fa - fb)
        f_new = fprime(x_new)
        if abs(f_new) < epsilon:
            return x_new
        if f_new > 0:
            b = x_new
        else:
            a = x_new
    return None

root_chord = chord_method(0, 1.5, 1e-3)
```

```
print("Метод хорд (f'(x)=0):")
print("x* =", root_chord, ", f'(x*) =", fprime(root_chord))
```

Чтобы получить сам минимум $f(x^*)$, достаточно подставить x^* в $f(x)$.

5 Метод Ньютона (поиск решения $f'(x) = 0$)

5.1 Алгоритм

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}, \quad \text{где } f'(x) = x - \cos x,\ f''(x) = 1 + \sin x.$$

Начальное приближение: $x_0 = 1.0$

Итерация 1.

$$f'(1.0) = 1.0 - \cos(1.0) \approx 1 - 0.5403 = 0.4597, \quad f''(1.0) = 1 + \sin(1.0) \approx 1 + 0.8415 = 1.8415$$

$$x_1 = 1.0 - \frac{0.4597}{1.8415} \approx 1.0 - 0.2496 = 0.7504$$

Итерация 2.

$$f'(0.7504) \approx 0.7504 - \cos(0.7504) \approx 0.7504 - 0.7317 = 0.0187, \quad f''(0.7504) = 1 + \sin(0.7504) \approx 1 + 0.6816 = 1.6$$

$$x_2 = 0.7504 - \frac{0.0187}{1.6816} \approx 0.7504 - 0.0111 = 0.7393$$

Итерация 3.

$$f'(0.7393) \approx 0.7393 - \cos(0.7393) \approx 0.7393 - 0.7390 = 0.0003, \quad f''(0.7393) \approx 1 + \sin(0.7393) \approx 1 + 0.6730 = 1.6$$

$$x_3 = 0.7393 - \frac{0.0003}{1.6730} \approx 0.7393 - 0.00018 = 0.7391$$

Итерация 4.

$$f'(0.7391) \approx 0.7391 - \cos(0.7391) \approx 0.7391 - 0.7391 = \mathbf{0.00002} < \varepsilon$$

$$\Rightarrow \text{Достигнута точность } \varepsilon = 0.001, \text{ процесс остановлен.}$$

Ответ: $x^* \approx 0.7391$, $f(x^*) \approx \frac{1}{2}(0.7391)^2 - \sin(0.7391) \approx -0.4601$

5.2 Код на Python

```
def fsecond(x):
    return 1 + math.sin(x)

def newton_method(x0, epsilon=1e-3, max_iter=50):
    for i in range(max_iter):
        f1 = fprime(x0)
        f2 = fsecond(x0)
        if abs(f2) < 1e-12:
            return None
```

```
        x_new = x0 - f1/f2
        if abs(fprime(x_new)) < epsilon:
            return x_new
        x0 = x_new
    return None

root_newton = newton_method(1.0, 1e-3)
print("Метод Ньютона (f'(x)=0):")
print("x* =", root_newton, ", f'(x*) =", fprime(root_newton))
```