Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Домашнее задание №2
по дисциплине «Методы оптимизации»

Вариант: **14**

Преподаватель:
Селина Елена Георгиевна

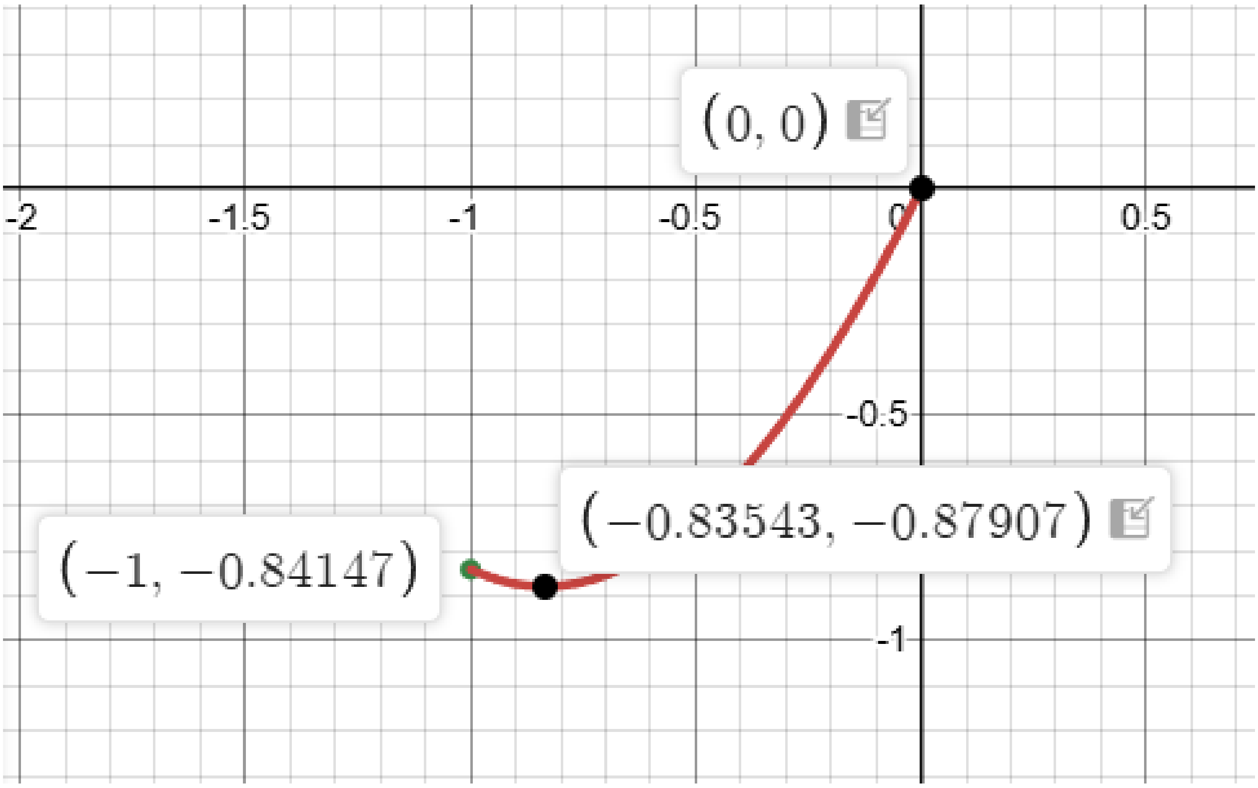
Выполнил:
Хромов Даниил Тимофеевич
Группа: P3218

Санкт-Петербург, 2024

Цель работы: Решить задачу **четырьмя методами**: методом половинного деления, методом золотого сечения, методом хорд и методом Ньютона.
По **5 шагов** каждого метода выполнить вручную + написать программу по каждому методу на одном из языков программирования.

1. Решение вручную

Исходная функция: $f(x) = x^2 + x + sin\,(x)$; $[a,\ b] = [-1,\ 0]$, $\varepsilon = 0,003$



1. **Вычисление по методу Ньютона:**

Шаг 0:
Начнем с середины заданного отрезка, т.е x0 = -0.5.
Шаг 1:
x = -0.8539459234514497, f `(x) = -0.05087832635920564

Шаг 2:
x = -0.8354707725126137, f `(x) = -0.00011291874937802149

|f'(x)| <= ε. Минимум с заданной погрешностью ε = 0.003 найден!
Минимум достигается в точке xm = -0.8354707725126137.
Значение в минимуме ym = -0.8790717623641279.

Минимум достигается в точке x_m = -0.8354707725126137.
Значение в минимуме y_m = f(x_m) = -0.8790717623641279.

2. Вычисление по методу половинного деления:

Шаг 1:
Рассматриваем отрезок [-1; 0].
a = -1, b = -0.4985.

Шаг 2:
Рассматриваем отрезок [-1; -0.4985].
a = -1, b = -0.74775.

Шаг 3:
Рассматриваем отрезок [-1; -0.74775].
a = -0.8753749999999999, b = -0.74775.

Шаг 4:
Рассматриваем отрезок [-0.8753749999999999; -0.74775].
a = -0.8753749999999999, b = -0.8100625.

Шаг 5:
Рассматриваем отрезок [-0.8753749999999999; -0.8100625].
a = -0.8442187499999999, b = -0.8100625.

Рассмотрено 5 шагов.

b - a < 2ε. Минимум с заданной погрешностью ε = 0.003 лежит на середине этого отрезка.

Текущий минимум: x_m = -0.8359033203124999.
Значение в минимуме $y_m = f(x_m)$ = -0.8790714570377751.

3. Вычисление по методу золотого сечения:

Шаг 1:
Рассматриваем отрезок [a = -1; b = 0].
a = -1, b = -0.382.

Шаг 2:
Рассматриваем отрезок [a = -1; b = -0.382].
a = -1, b = -0.618.

Шаг 3:
Рассматриваем отрезок [a = -1; b = -0.618].
a = -1, b = -0.763924.

Шаг 4:
Рассматриваем отрезок [a = -1; b = -0.763924].
a = -0.9098189680000001, b = -0.763924.

Шаг 5:
Рассматриваем отрезок [a = -0.9098189680000001; b = -0.763924].
a = -0.8540760000000001, b = -0.763924.

Рассмотрено 5 шагов.

b - a < e. Минмум с заданной погрешностью ε = 0.003 лежит на середине данного отрезка.
Текущий минимум: x_m = -0.8356735325758848.
Значение в минимуме $y_m = f(x_m)$ = -0.8790716831116877.

4. Вычисление по методу Хорд:

Шаг 1:
Текущий интервал: [a = -1.00000; b = 0.00000]
x_hat = -0.81311, f'(x_hat) = 0.06103
Новый интервал: [a = -1.00000; b = -0.81311]

Шаг 2:
Текущий интервал: [a = -1.00000; b = -0.81311]
x_hat = -0.83501, f'(x_hat) = 0.00115

|f'(x)| <= ε. Минимум найден с точностью ε = 0.003
Минимум в точке xm = -0.83501
Значение в минимуме ym = -0.87907

Текущий минимум: x_m = -0.8350113729756887.
Значение в минимуме $y_m = f(x_m)$ = -0.8790715249439732.

2. Программное решение

constants.py

```
import math
A = -1
B = 0
E = 0.003
```

derivatives.py

```
from math import sin, cos
def f(x: float) -> float:
    """Функция f."""
    return x**2 + x + sin(x)
def f_derivative_1(x: float) -> float:
    """Первая производная функции f."""
    return 2*x + 1 + cos(x)
def f_derivative_2(x: float) -> float:
    """Вторая производная функции f."""
    return 2 - sin(x)
F_DERIVATIES = [
    f,
    f_derivative_1,
    f_derivative_2,
]
```

golden_ratio.py

```
from typing import Callable
GOLDEN_RATIO_1 = 0.382
GOLDEN_RATIO_2 = 0.618
def solve(f_derivatives: list[Callable[[float], float]], _a: float, _b: float,
e: float) -> tuple[float, float]:
    f = f_derivatives[0]
    a = _a
    b = _b
    x1 = a + GOLDEN_RATIO_1 * (b - a)
    x2 = a + GOLDEN_RATIO_2 * (b - a)
    iteration = 1
    while (b - a > e):
        print(f"Шаг {iteration}:")
        print(f"Рассматриваем отрезок [a = {a}; b = {b}].")
        print(f"x1 = {x1}; x2 = {x2}; y1 = {f(x1)}; y2 = {f(x2)}.")
        if f(x1) < f(x2):
            print(f"y1 < y2 → b = {x2}; x2 = x1. Пересчет y2 не требуется.")
            b = x2
            x2 = x1
            x1 = a + GOLDEN_RATIO_1 * (b - a)
        else:
            print(f"y1 ≥ y2 → a = {x1}; x1 = x2. Пересчет y1 не требуется.")
            a = x1
            x1 = x2
            x2 = a + GOLDEN_RATIO_2 * (b - a)
        print(f"b - a = {b - a}.\n")
```

```
        iteration += 1
    print(f"\nb - a < e. Минмум с заданной погрешностью ε = {e} лежит на
середине данного отрезка.")
    x_m = (a + b) / 2
    y_m = f(x_m)
    print(f"Минимум в точке xm = {x_m}.")
    print(f"Значение в минимуме ym = {y_m}.\n")
    return x_m, y_m
```

half_division.py

```
from typing import Callable
def solve(f_derivatives: list[Callable[[float], float]], _a: float, _b: float,
e: float) -> tuple[float, float]:
    f = f_derivatives[0]
    a = _a
    b = _b
    iteration = 1
    while (b - a > 2 * e):
        print(f"Шаг {iteration}:")
        print(f"Рассматриваем отрезок [{a}; {b}].")
        x1 = (a + b - e) / 2
        x2 = (a + b + e) / 2
        y1 = f(x1)
        y2 = f(x2)
        print(f"x1 = {x1}, x2 = {x2}; y1 = {y1}, y2 = {y2}")
        if y1 > y2:
            print(f"y1 > y2 → Отсекаем начало отрезка: [a; x1], от {a} до
{x1}.")
            a = x1
        else:
            print(f"y1 ≤ y2 → Отсекаем конец отрезка: [x2; b], от {x2} до
{b}.")
            b = x2

        print(f"b - a = {b - a}.\n")
        iteration += 1
    print(f"\nb - a < 2ε. Минимум с заданной погрешностью ε = {e} лежит на
середине этого отрезка.")
    x_m = (a + b) / 2
    y_m = f(x_m)
    print(f"Минимум в точке xm = {x_m}.")
    print(f"Значение в минимуме ym = {y_m}.\n")
    return x_m, y_m
```

newton.py

```
from typing import Callable
def solve(f_derivatives: list[Callable[[float], float]], a: float, b: float, e:
float) -> tuple[float, float]:
    f = f_derivatives[0]
    f_d1 = f_derivatives[1]
    f_d2 = f_derivatives[2]
    x = (a + b) / 2
    print('Шаг 0:')
    print(f'Начнем с середины заданного отрезка, т.е x0 = {x}.')
    iteration = 1
    while abs(f_d1(x)) > e:
```

```
        print(f'Шаг {iteration}:')
        x = x - (f_d1(x) / f_d2(x))
        print(f"Касательная к графику функции f'(x) в точке x{iteration}
пересекает ось Oy в точке x{iteration+1} = {x}.")
        print(f"Выберем это новой точкой. В ней f'(x{iteration + 1}) =
{f_d1(x)}.\n")
        iteration += 1
    print(f"\n|f'(x)| <= ε. Минимум с заданной погрешностью ε = {e} найден!")
    print(f"Минимум достигается в точке xm = {x}.")
    print(f"Значение в минимуме ym = {f(x)}.\n")
    return x, f(x)
```

chord.py

```
from typing import Callable
def solve(
      f_derivatives: list[Callable[[float], float]],
      a: float,
      b: float,
      e: float,
) -> tuple[float, float]:
   f = f_derivatives[0]
   f_d1 = f_derivatives[1]

   fa = f_d1(a)
   fb = f_d1(b)
   if fa * fb >= 0:
      raise ValueError("Производные на концах отрезка должны иметь
разные знаки")

   iteration = 1
   x_hat = a

   while True:
      print(f"Шаг {iteration}:")
      print(f"Текущий интервал: [a = {a:.5f}; b = {b:.5f}]")

      fa = f_d1(a)
      fb = f_d1(b)
      x_hat = a - fa * (a - b) / (fa - fb)
      f_x = f_d1(x_hat)

      print(f"x_hat = {x_hat:.5f}, f'(x_hat) = {f_x:.5f}")

      if abs(f_x) <= e:
         print(f"\n|f'(x)| <= ε. Минимум найден с точностью ε = {e}")
         break

      if f_x > 0:
         b = x_hat
      else:
```

```
        a = x_hat

    print(f"Новый интервал: [a = {a:.5f}; b = {b:.5f}]\n")
    iteration += 1

x_m = x_hat
y_m = f(x_m)

print(f"Минимум в точке xm = {x_m:.5f}")
print(f"Значение в минимуме ym = {y_m:.5f}\n")

return x_m, y_m
```

main.py

```
from constants import A, B, E
from derivatives import F_DERIVATIES
from methods.newton import solve as solve_via_newton_method
from methods.golden_ratio import solve as solve_via_golden_ratio
from methods.half_division import solve as
solve_via_half_division
from methods.chord import solve as solve_via_chord_method
METHODS = [
    dict(
        name='Вычисление по методу Ньютона',
        func=solve_via_newton_method,
    ),
    dict(
        name='Вычисление по методу половинного деления',
        func=solve_via_half_division
    ),
    dict(
        name='Вычисление по методу золотого сечения',
        func=solve_via_golden_ratio,
    ),
    dict(
        name='Вычисление по методу хорд',
        func=solve_via_chord_method,
    ),
]
def main():
    for method in METHODS:
        print(f'==========\n\n{method["name"]}:\n')
        solve = method['func']
        x_m, y_m = solve(F_DERIVATIES, A, B, E)
        print(f'x_m = {x_m}')
        print(f'y_m = f(x_m) = {y_m}')
        print()
if __name__ == '__main__':
    main()
```

Вывод

В ходе выполнения домашнего задания я научился находить минимум функции методами Ньютона, половинного деления и золотого сечения с использованием Python. В результате работы были найдены минимумы уравнений на отрезке с определенной точностью.