МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
”ЛЭТИ”им. В.И.Ульянова(Ленина)

Кафедра САПР

ОТЧЁТ
к лабораторной работе №4
по дисциплине ”Информационные технологии”
Тема: ”Аппроксимация и производные”

Студент гр. 4353 _______Сизых П.В.

Преподаватель _______Копец Е.Е.

Санкт-Петербург
2025

Цель работы: научиться находить производные функций, а также промежутки убывания и возрастания, максимум и минимум функции, сверять полученные результаты с графиками функций

Действия проделанной работы

Задание 1. Понятие проивзодной.

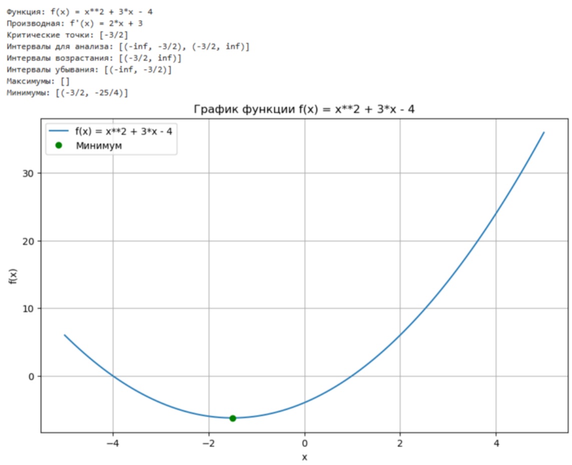
Для выполнения работы будет использована программа Jupyter Notebook. С помощью методических материалов были написаны программы для нахождения производной функции (рис. 1a,2a,3a). В этих же программах были найдены промежутки возрастания и убывания функций (1-3) с помощью их производных, максимум и минимум функций и построены графики (рис. 1b,2b,3b). Затем полученные результаты были сравнены с графиком функций.

(1)$f(x) = x^2 + 3x - 4$

(2)$f(x) = x^3 - 2x^2 + x - 6$

(3)$f(x) = e^{2x - x^2}$

(a)



(b)

Рис. 1 – Программа 1 и её график

Результат для функции (1):

Функция: $f(x) = x^2 + 3x - 4$

Производная: $f'(x) = 2x + 3$

Критические точки: [-3/2]

Интервалы для анализа: [(-∞, -3/2), (-3/2, ∞)]

Интервалы возрастания: [(-3/2, ∞)]

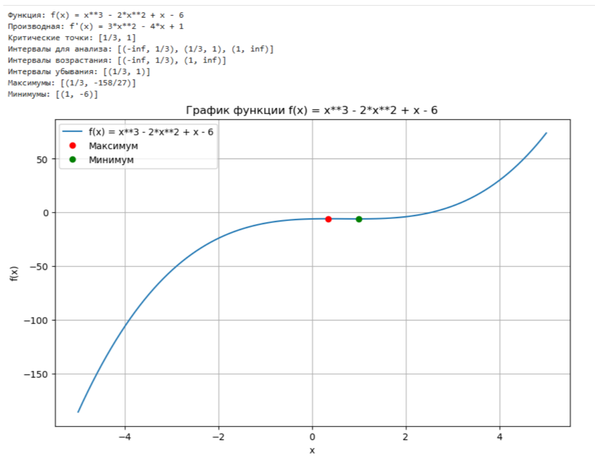
Интервалы убывания: [(-∞, -3/2)]

Максимумы: []

Минимумы: [(-3/2, -25/4)]

```python
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
x = sp.symbols('x')

def analyze_function(f, x_range=(-5, 5)):
    f_prime = sp.diff(f, x)
    print(f"Функция: f(x) = {f}")
    print(f"Производная: f'(x) = {f_prime}")

    critical_points = sp.solve(f_prime, x)
    print(f"Критические точки: {critical_points}")

    increasing_intervals = []
    decreasing_intervals = []
    if not critical_points:
        test_point = (x_range[0] + x_range[1]) / 2 #середина диапазона
        if f_prime.subs(x, test_point) > 0:
            increasing_intervals.append((float('-inf'), float('inf')))
        else:
            decreasing_intervals.append((float('-inf'), float('inf')))
    else:
        critical_points = sorted(critical_points)

        intervals = [ (float('-inf'), critical_points[0]) ]
        for i in range(len(critical_points) - 1):
            intervals.append( (critical_points[i], critical_points[i+1]) )
        intervals.append( (critical_points[-1], float('inf')) )
        print(f"Интервалы для анализа: {intervals}")

        for interval in intervals:
            if interval[0] == float('-inf'):
                test_point = interval[1] - 1  #чуть меньше
            elif interval[1] == float('inf'):
                test_point = interval[0] + 1  #чуть больше
            else:
                test_point = (interval[0] + interval[1]) / 2

            if f_prime.subs(x, test_point) > 0:
                increasing_intervals.append(interval)
            else:
                decreasing_intervals.append(interval)

    print(f"Интервалы возрастания: {increasing_intervals}")
    print(f"Интервалы убывания: {decreasing_intervals}")

    maxima = []
    minima = []
    for i, cp in enumerate(critical_points):
        left_interval = (float('-inf'), cp) if i == 0 else (critical_points[i-1], cp)
        right_interval = (cp, float('inf')) if i == len(critical_points)-1 else (cp, critical_points[i+1])

        if left_interval[0] == float('-inf'):
            test_point_left = left_interval[1] -1
        else:
            test_point_left = (left_interval[0] + left_interval[1]) / 2

        if right_interval[1] == float('inf'):
            test_point_right = right_interval[0] + 1
        else:
            test_point_right = (right_interval[0] + right_interval[1]) / 2


        if f_prime.subs(x, test_point_left) > 0 and f_prime.subs(x, test_point_right) < 0:
            maxima.append((cp, f.subs(x, cp)))
        elif f_prime.subs(x, test_point_left) < 0 and f_prime.subs(x, test_point_right) > 0:
            minima.append((cp, f.subs(x, cp)))

    print(f"Максимумы: {maxima}")
    print(f"Минимумы: {minima}")

    x_values = np.linspace(x_range[0], x_range[1], 100)
    f_values = [f.subs(x, val) for val in x_values]

    plt.figure(figsize=(10, 6))
    plt.plot(x_values, f_values, label=f"f(x) = {f}")

    for mx, my in maxima:
        plt.plot(mx, my, 'ro', label='Максимум')
    for mn, mny in minima:
        plt.plot(mn, mny, 'go', label='Минимум')

    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.title(f"График функции f(x) = {f}")
    plt.grid(True)
    plt.legend()
    plt.show()

f1 = x**3 - 2*x**2 + x - 6
analyze_function(f1)
```

(a)



(b)

Рис. 2 – Программа 2 и её график

Результат для функции (2):

Функция: $f(x) = x^3 - 2x^2 + x - 6$

Производная: $f'(x) = 3x^2 - 4x + 1$

Критические точки: [1/3, 1]

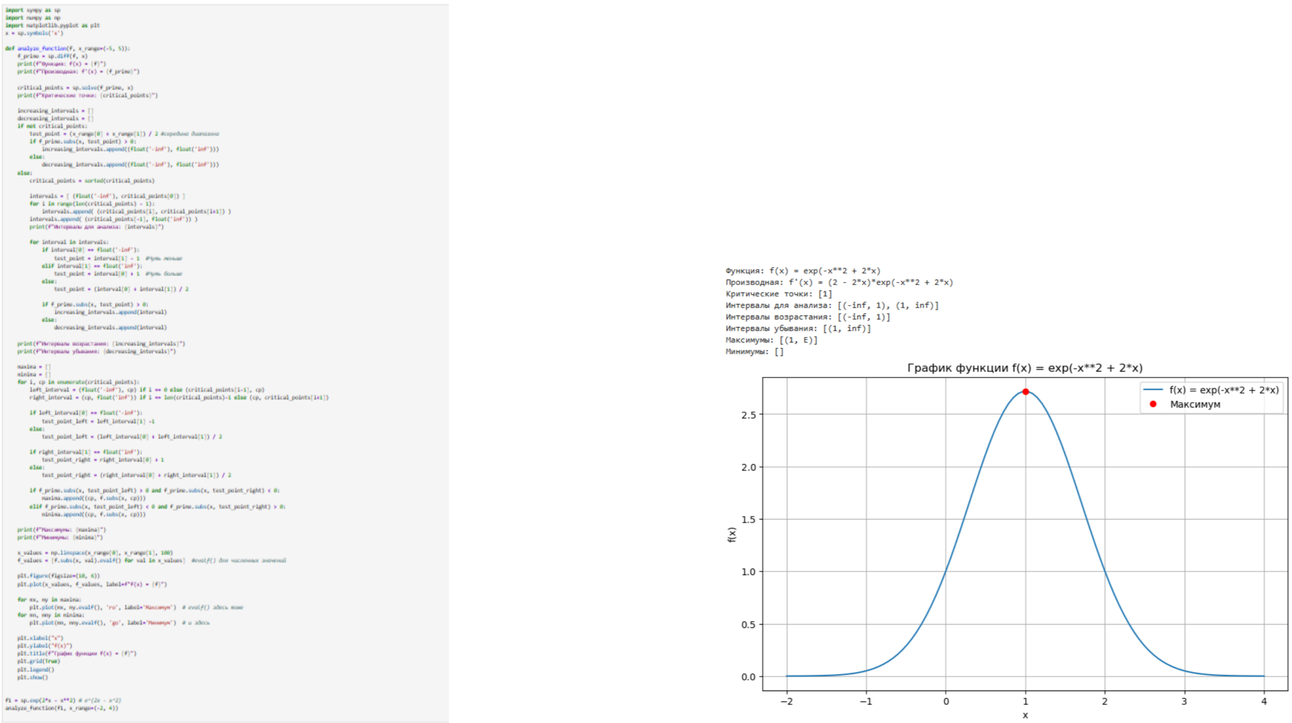
Интервалы для анализа: [(-∞, 1/3), (1/3, 1), (1, ∞)]

Интервалы возрастания: [(-∞, 1/3), (1, ∞)]

Интервалы убывания: [(1/3, 1)]

Максимумы: [(1/3, -158/27)]

Минимумы: [(1, -6)]



(a) (b)

Рис. 3 – Программа 3 и её график

Результат для функции (3):

Функция: $f(x) = e^{2x-x^2}$

Производная: $f'(x) = (2 - 2x) * e(-x^2 + 2x)$

Критические точки: [1]

Интервалы для анализа: [(-∞, 1), (1, ∞)]

Интервалы возрастания: [(-∞, 1)]

Интервалы убывания: [(1, ∞)]

Максимумы: [(1, e)]

Минимумы: []

Задание 2. Минимум MSE и техники вычисления производных (часть 1).

В задании дано 6 различных функций (1-6) и нужно найти их производные.

(1)$f(x) = 5x^2$

(2)$f(x) = 3sin(x)$

(3)$f(x) = 2/3e^x$

(4)$f(x) = 5x^2 + 3sin(x)$

(5)$f(x) = 7x^3 + 3x - 4$

(6)$f(x) = 13e^x + 3x^2 - 4sin(x)$

Для начала импортируем библиотеку sympy для символьных вычислений, создаём символьную переменную x (рис. 4).

```python
import sympy as sp
x = sp.symbols('x')
```

Рис. 4 – Программа 4.1

Затем задаём функцию, через sp.diff высчитаем производную и выводим результат (рис. 5).

```python
f = 5*x**2
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")
```

Рис. 5 – Программа 4.2

Аналогичные действия проделываем для всех 6-ти функций (рис.6).

```
x

[7]: import sympy as sp
x = sp.symbols('x')
f = 5*x**2
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 5*x**2
Производная: f'(x) = 10*x

[8]: import sympy as sp
x = sp.symbols('x')
f = 3 * sp.sin(x)
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 3*sin(x)
Производная: f'(x) = 3*cos(x)

[9]: import sympy as sp
x = sp.symbols('x')
f = (2/3) * sp.exp(x)
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 0.666666666666667*exp(x)
Производная: f'(x) = 0.666666666666667*exp(x)

[10]: import sympy as sp
x = sp.symbols('x')
f = 5*x**2 + 3*sp.sin(x)
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 5*x**2 + 3*sin(x)
Производная: f'(x) = 10*x + 3*cos(x)

[11]: import sympy as sp
x = sp.symbols('x')
f = 7*x**3 + 3*x - 4
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 7*x**3 + 3*x - 4
Производная: f'(x) = 21*x**2 + 3

[12]: import sympy as sp
x = sp.symbols('x')
f = 13*sp.exp(x) + 3*x**2 - 4*sp.sin(x)
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 3*x**2 + 13*exp(x) - 4*sin(x)
Производная: f'(x) = 6*x + 13*exp(x) - 4*cos(x)
```

Рис. 6 – Программа 5

Задание 3. Нахождение производных.

В задании дано 7 различных функций (1-7) и нужно найти их производные.

(1)$f(x) = 5x^5 + 7x$

(2)$f(x) = 6x^4 + 12x^2 + 5$

(3)$f(x) = 6x^3 + 3x^4 - 4$

(4)$f(x) = (2x + 3)(6x + 2)$

(5)$f(x) = (3x^2 + e^x)sin(x)$

(6)$f(x) = 3sin^2(x)$

(7)$f(x) = 13e^2x + 5x^7$

Для начала импортируем библиотеку sympy для символьных вычислений, создаём символьную переменную x (рис. 7).

```
import sympy as sp
x = sp.symbols('x')
```

Рис. 7 – Программа 6.1

Затем задаём функцию, через sp.diff высчитаем производную и выводим результат (рис. 8).

```
f = 5*x**2
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")
```

Рис. 8 – Программа 6.2

Аналогичные действия проделываем для всех 7-ми функций (рис. 9).

```
[13]: import sympy as sp
x = sp.symbols('x')
f = 5*x**5 + 7*x
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 5*x**5 + 7*x
Производная: f'(x) = 25*x**4 + 7

[14]: import sympy as sp
x = sp.symbols('x')
f = 6*x**4 + 12*x**2 + 5
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 6*x**4 + 12*x**2 + 5
Производная: f'(x) = 24*x**3 + 24*x

[15]: import sympy as sp
x = sp.symbols('x')
f = 6*x**3 + 3*x**4 - 4
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 3*x**4 + 6*x**3 - 4
Производная: f'(x) = 12*x**3 + 18*x**2

[16]: import sympy as sp
x = sp.symbols('x')
f = (2*x + 3) * (6*x + 2)
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = (2*x + 3)*(6*x + 2)
Производная: f'(x) = 24*x + 22
```

(a)

```
[17]: import sympy as sp
x = sp.symbols('x')
f = (3*x**2 + sp.exp(x)) * sp.sin(x)
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = (3*x**2 + exp(x))*sin(x)
Производная: f'(x) = (6*x + exp(x))*sin(x) + (3*x**2 + exp(x))*cos(x)

[18]: import sympy as sp
x = sp.symbols('x')
f = 3 * (sp.sin(x))**2
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 3*sin(x)**2
Производная: f'(x) = 6*sin(x)*cos(x)

[19]: import sympy as sp
x = sp.symbols('x')
f = 13 * sp.exp(2*x) + 5*x**7
f_prime = sp.diff(f, x)
print(f"Функция: f(x) = {f}")
print(f"Производная: f'(x) = {f_prime}")

Функция: f(x) = 5*x**7 + 13*exp(2*x)
Производная: f'(x) = 35*x**6 + 26*exp(2*x)
```

(b)

Рис. 9 – Программа 7

Вывод: были изучены техники вычисления производных функций, а также промежутки убывания и возрастания, максимум и минимум функции, полученные результаты были сверены с графиками функций.