

**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 производной.

Для выполнения работы будет использована программа Jupyter Notebook. С помощью методических материалов были написаны программы для нахождения производной функции (рис. 1а,2а,3а). 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. 1б,2б,3б). Затем полученные результаты были сравнены с графиком функций.

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

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

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

```
[1]: import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
x = sp.symbols('x')

def analyze_function(f, range(-5, 5)):
    f_prime = sp.diff(f, x)
    print("Функция: f(x) = " + str(f))
    print("Производная: f'(x) = " + str(f_prime))

    critical_points = sp.solve(f_prime, x)
    print("Критические точки: " + str(critical_points))

    increasing_intervals = []
    decreasing_intervals = []
    if not critical_points:
        for i in range(1, len(range(-5, 5)) // 2):
            intervals.append([range(-5, 5)[i], range(-5, 5)[i+1]])
    else:
        for i in range(len(critical_points) + 1):
            intervals.append([range(-5, 5)[i], range(-5, 5)[i+1]])
        print("Функция имеет асимптоту: " + str(intervals))

    critical_points = sorted(critical_points)

    for interval in intervals:
        if interval[0] == float("-inf") and interval[1] == float("inf"):
            continue
        else:
            test_point = interval[1] - 1 / 100 * error
            if f_prime.subs(x, test_point) > 0:
                increasing_intervals.append(interval)
            else:
                decreasing_intervals.append(interval)
    print("Промежутки возрастания: " + str(increasing_intervals))
    print("Промежутки убывания: " + str(decreasing_intervals))

    maxima = []
    minima = []
    for i in range(len(critical_points)):
        left_interval = (float("-inf"), critical_points[i])
        right_interval = (critical_points[i], float("inf"))
        if f_prime.subs(x, critical_points[i]) < 0:
            minima.append([left_interval[1], right_interval[0]])
        else:
            maxima.append([left_interval[1], right_interval[0]])

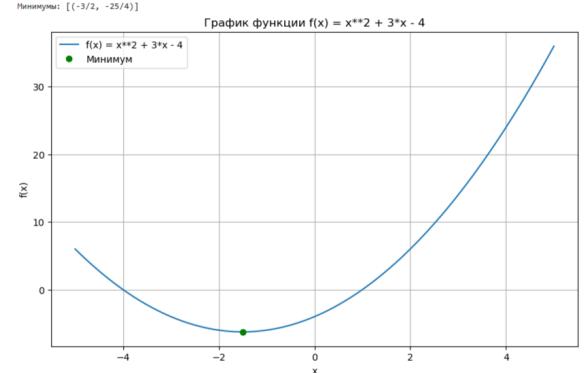
    print("Максимумы: " + str(maxima))
    print("Минимумы: " + str(minima))

    for i in range(len(maxima)):
        if maxima[i][0] < critical_points[i] < maxima[i][1]:
            maxima[i].append(f.subs(x, critical_points[i]))
        else:
            maxima[i].append(f.subs(x, critical_points[i-1]))
    for i in range(len(minima)):
        if minima[i][0] < critical_points[i] < minima[i][1]:
            minima[i].append(f.subs(x, critical_points[i]))
        else:
            minima[i].append(f.subs(x, critical_points[i+1]))

    print("Максимумы: " + str(maxima))
    print("Минимумы: " + str(minima))

    for i in range(len(maxima)):
        if maxima[i][2] > 0:
            maxima[i].append(f_prime.subs(x, critical_points[i]))
        else:
            maxima[i].append(f_prime.subs(x, critical_points[i-1]))
    for i in range(len(minima)):
        if minima[i][2] < 0:
            minima[i].append(f_prime.subs(x, critical_points[i]))
        else:
            minima[i].append(f_prime.subs(x, critical_points[i+1]))
```

Функция: $f(x) = x^2 + 3x - 4$
 Производная: $f'(x) = 2x + 3$
 Критические точки: [-3/2]
 Интервалы для анализа: [(-Inf, -3/2), (-3/2, Inf)]
 Интервалы возрастания: [(-3/2, Inf)]
 Интервалы убывания: [(-Inf, -3/2)]
 Максимумы: []
 Минимумы: [-3/2, -25/4]



(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: $[(-\infty, -3/2), (-3/2, \infty)]$

Интервалы возрастания: $[(-3/2, \infty)]$

Интервалы убывания: $[(-\infty, -3/2)]$

Максимумы: []

Минимумы: $[(-3/2, -25/4)]$

```
(1) import numpy as np
import numpy as np
import numpy as np
x = np.linspace(-5, 5)
print("x: ", x)

def analyze_function(f, x, range=(-5, 5)):
    f_x = f(x)
    print("f(x): ", f_x)
    print("f'(x): ", f_x[0])
    print("f''(x): ", f_x[1])

    critical_points = np.where(np.isnan(f_x[0]), np.inf, f_x[0])
    print("critical points: ", critical_points)

    increasing_intervals = []
    decreasing_intervals = []
    for i in range(len(critical_points) - 1):
        test_point = (x[i] + x[i+1]) / 2
        if f_x[0][i] < f_x[0][i+1]:
            increasing_intervals.append((float('inf'), float('inf')))
        else:
            decreasing_intervals.append((float('inf'), float('inf')))

    critical_points = sorted(critical_points)

    intervals = [(float('inf'), critical_points[0])]
    for i in range(len(critical_points) - 1):
        if f_x[0][i] > f_x[0][i+1]:
            intervals.append((critical_points[i], critical_points[i+1]))
        else:
            intervals.append((critical_points[i+1], float('inf')))

    for interval in intervals:
        if len(interval) == float('inf'):
            continue
        elif len(interval) == 1:
            print("interval: ", interval)
        else:
            test_point = (interval[0] + interval[1]) / 2
            if f_x[0][int((test_point / (x[1] - x[0])))] < f_x[0][int((test_point / (x[1] - x[0])) + 1)]:
                increasing_intervals.append(interval)
            else:
                decreasing_intervals.append(interval)

    print("Increasing intervals: ", increasing_intervals)
    print("Decreasing intervals: ", decreasing_intervals)

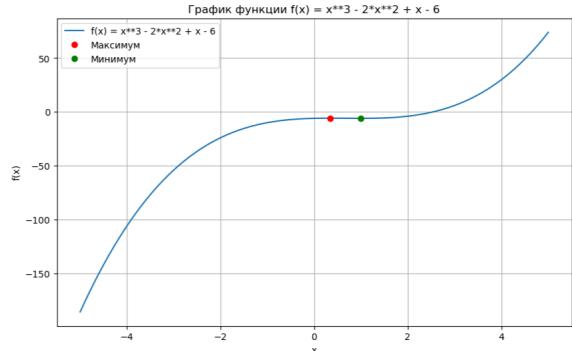
    maxima = []
    minima = []
    for i, cp in enumerate(critical_points):
        if cp == critical_points[0] or cp == critical_points[-1]:
            if f_x[0][i] > f_x[0][i+1] and f_x[0][i] > f_x[0][i-1] or (cp == critical_points[-1] and f_x[0][i] > f_x[0][i-1]):
                maxima.append(cp)
            elif f_x[0][i] < f_x[0][i+1] and f_x[0][i] < f_x[0][i-1]:
                minima.append(cp)
        else:
            if f_x[0][i] > f_x[0][i+1] and f_x[0][i] > f_x[0][i-1]:
                maxima.append(cp)
            elif f_x[0][i] < f_x[0][i+1] and f_x[0][i] < f_x[0][i-1]:
                minima.append(cp)

    print("Maxima: ", maxima)
    print("Minima: ", minima)

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

(a)

Функция: $f(x) = x^3 + 2x^2 + x - 6$
 Производная: $f'(x) = 3x^2 + 4x + 1$
 Критические точки: $[1/3, 1]$
 Интервалы для анализа: $[(-\infty, 1/3), (1/3, 1), (1, \infty)]$
 Интервалы возрастания: $[(-\infty, 1/3), (1, \infty)]$
 Интервалы убывания: $[(1/3, 1)]$
 Максимумы: $[(1/3, -158/27)]$
 Минимумы: $[(1, -6)]$



(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: $[(-\infty, 1/3), (1/3, 1), (1, \infty)]$

Интервалы возрастания: $[(-\infty, 1/3), (1, \infty)]$

Интервалы убывания: $[(1/3, 1)]$

Максимумы: $[(1/3, -158/27)]$

Минимумы: $[(1, -6)]$

```

#(1) import numpy as np
import math as m
import mathlib as ml
x = np.linspace(-2, 4, 1000)
def analyze_function(x, range(-2, 5)):
    f_prime = ml.diff(f, x)
    print("f'(x) = " + str(f_prime))
    print("Производная f'(x) = (" + str(f_prime) + ")")
    print("Признаки производной f'(x) = (" + str(f_prime) + ")")

    critical_points = sp.solveset(f_prime, x)
    print("Критические точки: " + str(critical_points))

    increasing_intervals = []
    decreasing_intervals = []

    if not critical_points:
        print("Нет критических точек в диапазоне!")
    else:
        for point in critical_points:
            if f_prime.simplify() < 0:
                increasing_intervals.append(point)
            else:
                decreasing_intervals.append(point)

    critical_points = sorted(critical_points)

    intervals = []
    for interval in intervals:
        if interval[0] == float("-inf"):
            if interval[1] == float("inf"):
                print("Функция возрастает на (-infinity, infinity)!")
            else:
                print("Функция возрастает на (-infinity, " + str(interval[1]) + ")")
        else:
            if interval[0] == float("inf"):
                print("Функция убывает на (" + str(interval[0]) + ", infinity)!")
            else:
                print("Функция убывает на (" + str(interval[0]) + ", " + str(interval[1]) + ")")

    print("Функция имеет максимумы: " + str(increasing_intervals))
    print("Функция имеет минимумы: " + str(decreasing_intervals))

    maxima = []
    minima = []
    for i, x_val in enumerate(critical_points):
        left_val = x_val - 0.1
        right_val = x_val + 0.1
        if f_prime.simplify() < 0 and i == len(critical_points)-1 or f_prime.simplify() > 0 and i == len(critical_points)-1:
            if left_val == float("-inf") or right_val == float("inf"):
                print("Максимум: (" + str(x_val) + ", f(" + str(x_val) + "))")
            else:
                print("Максимум: (" + str(left_val) + ", " + str(right_val) + ", f(" + str(x_val) + "))")
        elif f_prime.simplify() < 0 and i == len(critical_points)-1 or f_prime.simplify() > 0 and i == len(critical_points)-1:
            if left_val == float("-inf") or right_val == float("inf"):
                print("Минимум: (" + str(x_val) + ", f(" + str(x_val) + "))")
            else:
                print("Минимум: (" + str(left_val) + ", " + str(right_val) + ", f(" + str(x_val) + "))")

    print("Признаки максимума: " + str(increasing_intervals))
    print("Признаки минимума: " + str(decreasing_intervals))

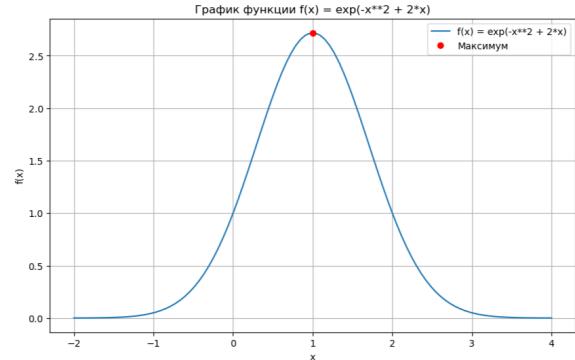
    x_values = np.linspace(x, range(0, 100))
    x_values = [x_val.evalf() for x_val in x_values]
    print("x_values: " + str(x_values))

    plt.figure(figsize=(10, 6))
    plt.plot(x, f)
    plt.plot(x, f_prime)
    plt.title("График функции f(x) = exp(-x**2 + 2*x)")
    plt.xlabel("x")
    plt.ylabel("f(x)")

    plt.grid(True)
    plt.show()

```

Функция: $f(x) = \exp(-x^2 + 2x)$
 Производная: $f'(x) = (2 - 2x) * \exp(-x^2 + 2x)$
 Критическая точка: $[1]$
 Интервалы для анализа: $\{(-\infty, 1), (1, \infty)\}$
 Интервалы возрастания: $\{(-\infty, 1)\}$
 Интервалы убывания: $\{(1, \infty)\}$
 Максимумы: $[(1, e)]$
 Минимумы: $[]$



(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: $\{(-\infty, 1), (1, \infty)\}$

Интервалы возрастания: $\{(-\infty, 1]\}$

Интервалы убывания: $[(1, \infty)]$

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.

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

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

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

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

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

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

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

```
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).

```
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).

```
[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) = 0.6666666666666667*exp(x)
Производная: f'(x) = 0.6666666666666667*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) $f(x) = 3\sin^2(x)$
- (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).

```
[11]: 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

[12]: 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

[13]: import sympy as sp
x = sp.symbols('x')
f = 6*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) = 3*x**4 + 12*x**3 + 4
Производная: f'(x) = 12*x**3 + 36*x**2

[14]: 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

[15]: import sympy as sp
x = sp.symbols('x')
f = (3*x**2 + exp(x))*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) = (0*x + exp(x))*sin(x) + (3*x**2 + exp(x))*cos(x)

[16]: import sympy as sp
x = sp.symbols('x')
f = 3*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)

[17]: import sympy as sp
x = sp.symbols('x')
f = 13*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)
```

(a)

(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.