	Министерство науки и высшего образования Российской Федерации **Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Московский государственный технический университет имени Н.Э. Баумана** **(национальный исследовательский университет)»** **(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ "Информатика и системы управления"

КАФЕДРА "Программное обеспечение ЭВМ и информационные технологии"

ОТЧЁТ

К ЛАБОРАТОРНОЙ РАБОТЕ №2

ПО КУРСУ МОДЕЛИРОВАНИЕ НА ТЕМУ:

"Построение и программная реализация алгоритмов численного интегрирования"

Студент ИУ7-53 БВ _________________ Д.П. Косаревский
(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель _________________ В.М. Градов
(Подпись, дата) (И.О.Фамилия)

Москва 2021 г.

Цель работы: Получение навыков построения алгоритма вычисления двукратного интеграла с использованием квадратурных формул Гаусса и Симпсона.

Задание:

Построить алгоритм и программу для вычисления двукратного интеграла при фиксированном значении параметра

$$\varepsilon(\tau) = \frac{4}{\pi}\int_0^{\pi/2} d\varphi \int_0^{\pi/2} [1-\exp(-\tau \frac{l}{R})]\ \cos\theta\ \sin\theta\ d\theta\ ,$$

где $$\frac{l}{R} = \frac{2\cos\theta}{1-\sin^2\theta\cos^2\varphi},$$

θ, φ - углы сферических координат.

Применить метод последовательного интегрирования. По одному направлению использовать формулу Гаусса, а по другому - формулу Симпсона.

Результаты

В результате работы была достигнута поставленная цель и получены следующие результаты:

1. **Описать алгоритм вычисления n корней полинома Лежандра n-ой степени** $P_n(x)$ **при реализации формулы Гаусса.**

   Три свойства полиномов Лежандра:

   1) На отрезке [-1, 1] полином Лежандра n-ой степени имеет n различных и действительных корней.
   2) Рекуррентное соотношение, которое связывает полином:
      $L_m(x) = 1/m[(2m - 1)xL_{m-1}(x) - (m - 1)L_{m-2}(x)]$
   3) $\int_{-1}^{1} P_m(x)L_n(x)dx = 0,\ m < n$

   Из св-ва (3) при $P_k(t) = t^2,\ k = 0,\ 1,\ 2 \dots\ N - 1$

   $\int_{-1}^{1} t^k L_N(t)dt = 0 = \sum_{i=1}^{N} A_i t_i^k L_N(t_i)$ – равенство соблюдается, если

$L_N(t_i) = 0$

В зависимости от степени n полином Лежандра является либо четной, либо нечетной функцией. Таким образом, при поиске корней можно исследовать только отрезок [0, 1]. Каждый корень можно искать методом половинного деления, проходя по промежутку [0, 1] с маленьким шагом. На каждом шаге проверять знаки функции на концах интервала, если они различны, то на интервале есть корень, который и находится методом половинного деления, если функция с заданной точностью в одном из концов равна 0, то корень найден без дополнительных действий, если знаки одинаковы — корней на заданном интервале нет, так как случай кратных корней не учитывается в силу свойств полинома Лежандра.

2. **Исследовать влияние количества выбранных узлов сетки по каждому направлению на точность расчетов.**

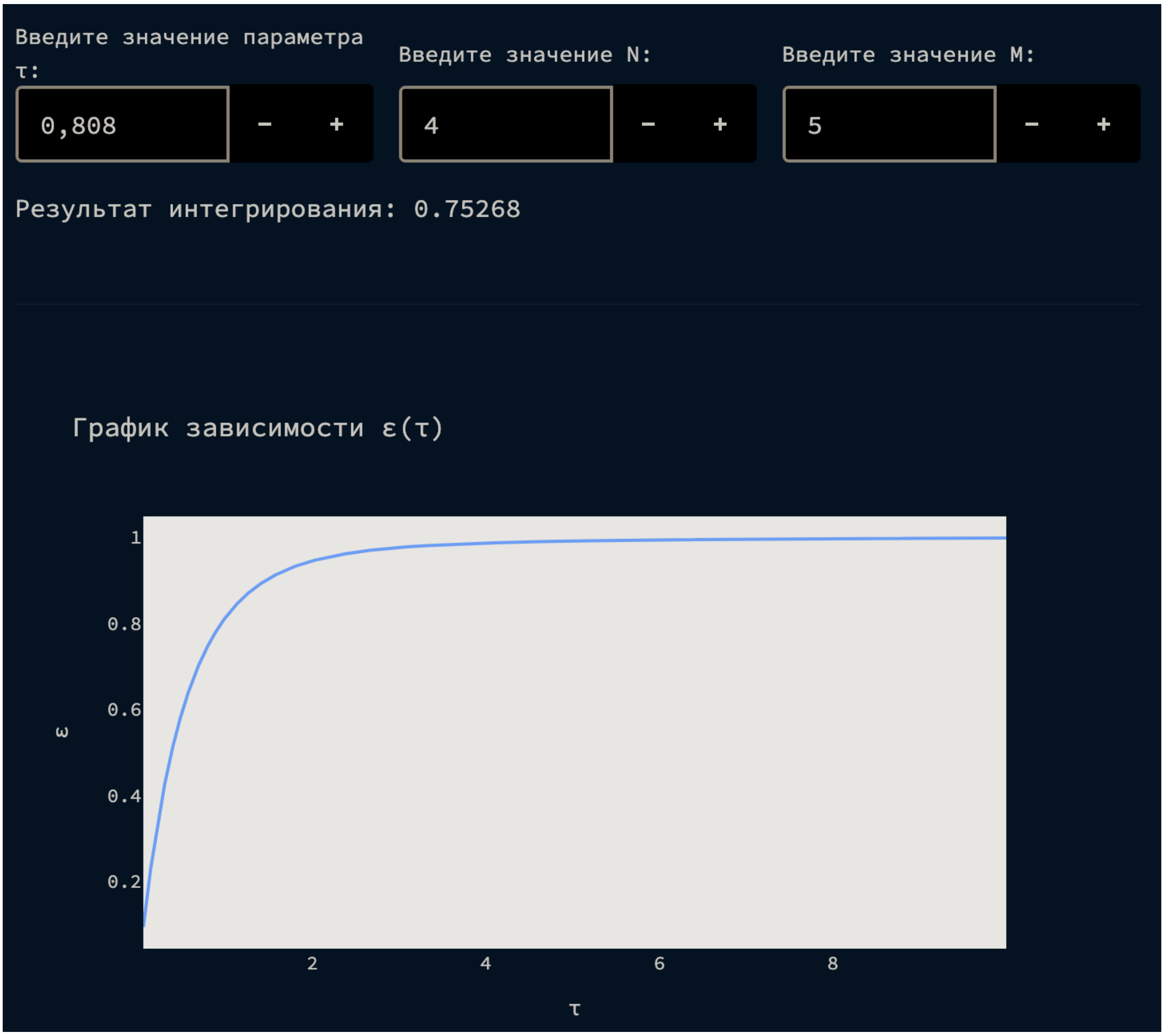
   Построим таблицу значений интеграла при фиксированном τ = 0.808:

M | N	1	2	3	4
1	1.23244	0.69883	0.74321	0.74626
2	1.23475	0.71082	0.75136	0.75251
3	1.23467	0.70987	0.75156	0.75295
4	1.23467	0.70961	0.75117	0.7528

   Из наблюдений можем сделать вывод, что зависимость ошибки от параметра N не является существенной, из чего следует, что метод Гаусса выдаёт более высокую точность в сравнении с методом Симпсона.

3. **Построить график зависимости ε(τ) в диапазоне изменения τ = 0.05-10. Указать при каком количестве узлов получены результаты.**

   График построен со значениями τ=0.808, N = 4, M = 5



Работающую программу можно увидеть и протестировать через web-интерфейс по следующей ссылке:

https://share.streamlit.io/dkosarevsky/math_modelling/main.py

Программа написана на языке программирования Python 3.9.7 с использованием следующих библиотек:

- streamlit
- plotly
- numpy

Код

```
import plotly.graph_objects as go
import streamlit as st
import numpy as np

from typing import List, Callable


def simpson_integrate(function: Callable, a: float, b: float, n: int) -> float:
    """ интегрирование методом Симпсона """
    h, res = (b - a) / n, 0
    x = a
    while x < b:
        next_x = x + h
        res += function(x) + 4 * function(x + h * .5) + function(next_x)
        x = next_x
    return h / 6 * res


def legendre_polynomial(n: int, x: float) -> float:
    """ значение полинома Лежандра n-го порядка """
    if n < 2:
        return [1, x][n]
    p1, p2 = legendre_polynomial(n - 1, x), legendre_polynomial(n - 2, x)
    return ((2 * n - 1) * x * p1 - (n - 1) * p2) / n


def derivative_legendre_polynomial(n: int, x: float) -> float:
    """ значение производной полинома Лежандра """
    p1, p2 = legendre_polynomial(n - 1, x), legendre_polynomial(n, x)
    return n / (1 - x * x) * (p1 - x * p2)


def roots_legendre_polynomial(n: int, eps: float = 1e-12) -> List[float]:
    """ корни полинома Лежандра n-го порядка """
    roots = [np.cos(np.pi * (4 * i + 3) / (4 * n + 2)) for i in range(n)]
    for i, root in enumerate(roots):  # уточнение корней
        root_val = legendre_polynomial(n, root)
        while abs(root_val) > eps:
            root -= root_val / derivative_legendre_polynomial(n, root)
            root_val = legendre_polynomial(n, root)
        roots[i] = root
    return roots


def gauss_integrate_norm(f: Callable, n: int) -> float:
    """ интегрирование методом Гаусса на промежутке [-1, 1] """
    t = roots_legendre_polynomial(n)
    T = np.array([[t_i ** k for t_i in t] for k in range(n)])

    tk = lambda k: 2 / (k + 1) if k % 2 == 0 else 0
    b = np.array([tk(k) for k in range(n)])
    A = np.linalg.solve(T, b)  # решение СЛАУ

    return sum(A_i * f(t_i) for A_i, t_i in zip(A, t))
```

```
def gauss_integrate(f: Callable, a: float, b: float, n: int) -> float:
    """ интегрирование методом Гаусса на произвольном промежутке """
    mean, diff = (a + b) * .5, (b - a) * .5
    g = lambda t: f(mean + diff * t)
    return diff * gauss_integrate_norm(g, n)


def composite_integrate(f: Callable, a1: float, b1: float, a2: float, b2: float,
                        method_1: Callable, method_2: Callable, n1: int, n2: int) -> float:
    func = lambda y: method_1(lambda x: f(x, y), a1, b1, n1)
    return method_2(func, a2, b2, n2)


def function_integrator(f: Callable, a: float, b: float, c: float, d: float, n: int, m: int) ->
float:
    return composite_integrate(f, a, b, c, d, gauss_integrate, simpson_integrate, n, m)


def integrate_function(t: float, n: int, m: int) -> float:
    l_r = lambda theta, phi: 2 * np.cos(theta) / (1 - np.sin(theta) ** 2 * np.cos(phi) ** 2)
    f = lambda theta, phi: (1 - np.exp(-t * l_r(theta, phi))) * np.cos(theta) * np.sin(theta)

    return 4 / np.pi * function_integrator(f, 0, np.pi * .5, 0, np.pi * .5, n, m)


def plot(func: Callable, n: int, m: int, test: bool = False, test_func: Callable = None,
test_func2: Callable = None):
    """ отрисовка графика """
    st.markdown("---")
    x = np.arange(.05, 10, .001) if not test else np.arange(-10, 10, .001)
    y = [func(t, n, m) for t in x] if not test else test_func2(x) - test_func2(0 * x)

    fig = go.Figure()
    fig.add_trace(go.Scatter(
        x=x,
        y=y,
        mode='lines',
    ))

    if test:
        def test_y(n_test):
            return [func(test_func, 0, x_test, n_test) for x_test in x]

        fig.add_trace(go.Scatter(
            x=x,
            y=test_y(1),
            mode='lines',
        ))

    fig.update_layout(
        title_text=f"График зависимости ε(τ)" if not test else "График тестирования",
        xaxis_title="τ" if not test else "",
        yaxis_title="ε" if not test else "",
        showlegend=False
    )

    st.write(fig)
    st.markdown("---")
```

```
def test_function(func: Callable):
    def test_f(x):
        return .3 * (x - 1) ** 2 - .1 * (x + 4) ** 2 - x

    def act_int_f(x):
        return .1 * (x - 1) ** 3 - .1 / 3 * (x + 4) ** 3 - .5 * x ** 2

    plot(func, 0, 0, test=True, test_func=test_f, test_func2=act_int_f)


def main():
    st.markdown("### Лабораторная работа №2")
    st.markdown("**Тема:** Построение и программная реализация алгоритмов численного
интегрирования.")
    st.markdown("""**Цель работы:** Получение навыков построения алгоритма вычисления двукратного
        интеграла с использованием квадратурных формул Гаусса и Симпсона.""")

    st.write("---")

    c0, c1, c2 = st.columns(3)
    tau = c0.number_input("Введите значение параметра τ:", min_value=.0, max_value=1.,
value=.808, format="%.3f")
    N = c1.number_input("Введите значение N:", min_value=1, max_value=100, value=4, step=1)
    M = c2.number_input("Введите значение M:", min_value=1, max_value=100, value=5, step=1)

    result = integrate_function(tau, N, M)
    st.write(f"Результат интегрирования: {round(result, 5)}")

    plot(integrate_function, N, M)

    test = st.selectbox("Выберите метод тестирования", ("Гаусс", "Симпсон"))
    if test == "Симпсон":
        test_function(simpson_integrate)
    elif test == "Гаусс":
        test_function(gauss_integrate)


if __name__ == "__main__":
    main()
```

Вопросы

1. В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается?

Теоретический порядок квадратурных формул численного интегрирования может не достигаться в случаях больших значений производной интегрируемой функции, т.е. при наличии резких скачков функции, а также в случае, если подынтегральная функция не имеет производных.

2. Построить формулу Гаусса численного интегрирования при одном узле.

$$\int_{-1}^{1} f(t)dt = A_1 f(t_1)$$

$$A_1 = 2$$

$$\int_{-1}^{1} f(t)dt = A_1 f(t_1) = 2f(0)$$

на произвольном промежутке [a, b]:

$$x_1 = (b + a)/2 + (b - a)t_1/2 = (b + a)/2$$

$$\int_{a}^{b} f(x)dx = (b - a)/2 * (A_1 f(x_1)) = (b - a)/2 * 2f((b + a)/2) = (b - a)f((b +$$

3. Построить формулу Гаусса численного интегрирования при двух узлах.

$$\int_{-1}^{1} f(t)dt = A_1 f(t_1) + A_2 f(t_2)$$

$$\{A_1 + A_2 = 2$$

$$\{A_1 t_1 + A_2 t_2 = 0$$

$$P_2(x) = 1/2(3x^2 - 1) \Rightarrow x_{1,2} \pm 1/\sqrt{3}$$

$$\{A_1 + A_2 = 2$$

$$\{- 1/\sqrt{3} * A_1 + 1/\sqrt{3} = 0 \Rightarrow A_1 = A_2 = 1$$

$$\int_{-1}^{1} f(t)dt = f(-1/\sqrt{3}) + f(1/\sqrt{3})$$

на произвольном промежутке [a, b]:

$$x_1 = (b + a)/2 + (b - a)t_1/2 = (b + a)/2 - (b - a)/2 * 1/\sqrt{3}$$

$$x_2 = (b + a)/2 + (b - a)t_2/2 = (b + a)/2 - (b - a)/2 * 1/\sqrt{3}$$

4. Получить обобщенную кубатурную формулу, для вычисления двойного интеграла методом последовательного интегрирования на основе формулы трапеций с тремя узлами по каждому направлению.

$$\int_c^d \int_a^b f(x, y)dxdy = \int_a^b dx \int_c^d f(x, y)dy = \int_a^b F(x)dx = h_x(1/2F(x_0) + F(x_1) + 1/2F(x_2)) =$$

$$= h_x h_y[1/2(1/2f(x_0, y_0) + f(x_0, y_1) + 1/2f(x_0, y_2)) + 1/2f(x_1, y_0) + f(x_1, y_1) + 1/2f(x_1, y_2) +$$

$$+ 1/2(1/2f(x_2, y_0) + f(x_2, y_1) + 1/2f(x_2, y_2)],$$ где $h_x = (b - a)/2$, $h_y = (d - c)/2$