**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное автономное 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 | ИУ6 Компьютерные системы и сети |
| Группа | ИУ6-63Б |

# ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

## Отчет по домашнему заданию N4:

## Численное интегрирование

## Вариант 10


| | | |
|---|---|---|
| Студент: | | В.К. Залыгин |
| | дата, подпись | Ф.И.О. |
| Преподаватель: | | Я.Ю. Павловский |
| | дата, подпись | Ф.И.О. |


*Москва, 2025*

## Задание

Цель домашней работы: изучения методов численного интегрирования, изучения метода Рунге.

Для достижения цели необходимо решить следующие задачи:

1. Найти по формуле Ньютона-Лейбница точное значение заданного определенного интеграла и внести полученное значение в отчет с 4-мя верными знаками после запятой.
2. Реализовать в среде MatLab или на языке Python вычисление данного определенного интеграла по формуле центральных прямоугольников и трапеции и найти его значения с выбором шага (вторая производная) для удолетворении заданной точности .
3. Реализовать в среде MatLab или на языке Python вычисление данного определенного интеграла по трапеции и Симпсона и найти его значения с автоматическим выбором шага по правилу Рунге для удовлетворения заданной точности.
4. Сравнить полученные значения с точным значением определенного интеграла, убедится в том, что результаты удовлетворяют заданной точности.

**Методы численного интегрирования**

### Метод центральных прямоугольников

Приближает интеграл суммой площадей прямоугольников, высоты которых равны значению функции в середине каждого подотрезка.

Формула:

$$I \approx h \sum_{i=0}^{n-1} f\left(a + h\left(i + \frac{1}{2}\right)\right),$$

### Метод трапеций

Аппроксимирует функцию на каждом подотрезке отрезком прямой и вычисляет сумму площадей трапеций:

$$I \approx h\left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(a + ih)\right),$$

**Правило Рунге для метода трапеций**

Позволяет автоматически подобрать число разбиений nnn для достижения заданной точности ε. Погрешность оценивается по разности интегралов на сетках с n и 2n узлами:

$$\left| \frac{I_n - I_{2n}}{3} \right| \leq \varepsilon.$$

**Метод Симпсона (параболический метод)**

Приближает интеграл суммой площадей парабол, проходящих через три последовательные точки. Требует чётное число разбиений nnn. Формула:

$$I \approx \frac{h}{3} \left( f(a) + f(b) + 4 \sum_{\text{нечетные } i} f(a+ih) + 2 \sum_{\text{четные } i} f(a+ih) \right).$$

**Правило Рунге для метода Симпсона**

Для метода Симпсона точность оценивается по правилу Рунге с другим коэффициентом:

$$\left| \frac{I_n - I_{2n}}{15} \right| \leq \varepsilon.$$

**Погрешность**

Для каждого метода дополнительно рассчитывается абсолютная ошибка как разность между численным и точным значениями интеграла.

Значения для индивидуального варианта указаны в таблице 1.

Таблица 1 – Значения индивидуального варианта

| № Варианта | $y(x)$ | Отрезок интегрирования [a;b] |
|---|---|---|
| 10 | $\frac{1}{2x + \sqrt{3x+1}}$ | [0;5] |

Решением интеграла $\int \frac{1}{2x+\sqrt{3x+1}}dx$ является семейство функций $\frac{1}{5}(4\ln(\sqrt{3x+1}+2)) + \ln(2\sqrt{3x+1}-1)) + C.$

Листинг 1 – Код программы main.py

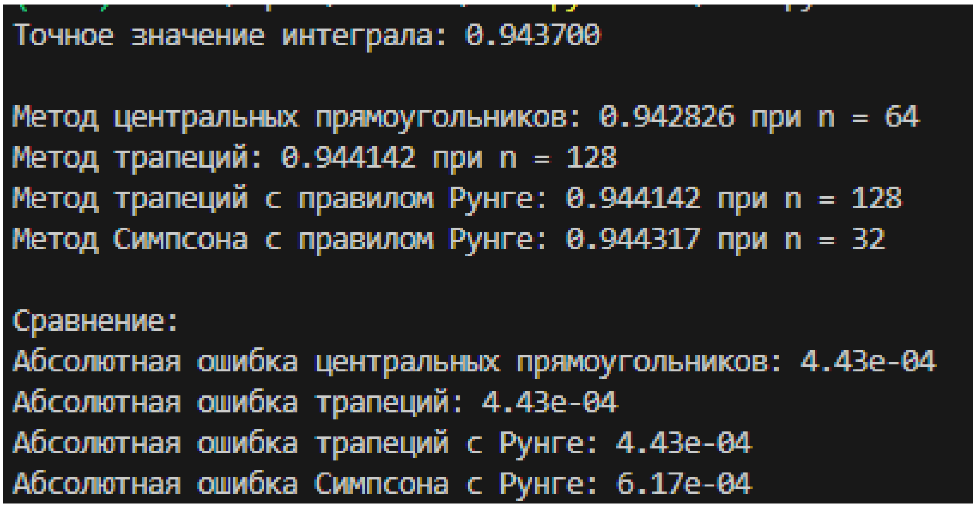
```
from data_input import get_parameters, exact_integral
from methods import (
    central_rectangles,
    trapezoidal,
    runge_trapezoidal,
    runge_simpson,
)

def main():
    a, b, eps = get_parameters()
    exact = exact_integral()

    print(f"Точное значение интеграла: {exact:.6f}\n")

    # Центральные прямоугольники
    n = 2
    while True:
        integral = central_rectangles(a, b, n)
        next_integral = central_rectangles(a, b, n * 2)
        if abs(integral - next_integral) / 3 <= eps:
            break
        n *= 2
    print(f"Метод центральных прямоугольников: {next_integral:.6f}
при n = {n * 2}")

    # Метод трапеций с шагом по второй производной
    n = 2
    while True:
        integral = trapezoidal(a, b, n)
        next_integral = trapezoidal(a, b, n * 2)
        if abs(integral - next_integral) / 3 <= eps:
            break
        n *= 2
    print(f"Метод трапеций: {next_integral:.6f} при n = {n * 2}")

    # Метод трапеций с Рунге
    integral_runge, n_runge = runge_trapezoidal(a, b, eps)
    print(f"Метод трапеций с правилом Рунге: {integral_runge:.6f}
при n = {n_runge}")

    # Метод Симпсона с Рунге
    integral_simpson, n_simpson = runge_simpson(a, b, eps)
    print(f"Метод Симпсона с правилом Рунге: {integral_simpson:.6f}
при n = {n_simpson}")

    print("\nСравнение:")
    print(f"Абсолютная ошибка центральных прямоугольников:
{abs(next_integral - exact):.2e}")
```

```
    print(f"Абсолютная ошибка трапеций: {abs(next_integral - 
exact):.2e}")
    print(f"Абсолютная ошибка трапеций с Рунге: {abs(integral_runge 
- exact):.2e}")
    print(f"Абсолютная ошибка Симпсона с Рунге: 
{abs(integral_simpson - exact):.2e}")

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

Листинг 2 – Код программы methods.py

```
from data_input import f

def central_rectangles(a, b, n):
    """ Метод центральных прямоугольников. """

    h = (b - a) / n
    result = 0
    for i in range(n):
        xi = a + h * (i + 0.5)
        result += f(xi)
    return result * h

def trapezoidal(a, b, n):
    """ Метод трапеций. """

    h = (b - a) / n
    result = (f(a) + f(b)) / 2
    for i in range(1, n):
        xi = a + h * i
        result += f(xi)
    return result * h

def runge_trapezoidal(a, b, eps):
    """ Метод трапеций с автоматическим выбором шага по правилу 
Рунге. """

    n = 2
    I_n = trapezoidal(a, b, n)
    n *= 2
    I_2n = trapezoidal(a, b, n)
    while abs(I_n - I_2n) / 3 > eps:
        n *= 2
        I_n = I_2n
        I_2n = trapezoidal(a, b, n)
    return I_2n, n

def simpson(a, b, n):
    """ Метод Симпсона. """

    if n % 2 != 0:
        n += 1  # Симпсон требует чётное число отрезков
    h = (b - a) / n
    result = f(a) + f(b)
    for i in range(1, n, 2):
```

```
        result += 4 * f(a + i * h)
    for i in range(2, n-1, 2):
        result += 2 * f(a + i * h)
    return result * h / 3

def runge_simpson(a, b, eps):
    """ Метод Симпсона с автоматическим выбором шага по правилу
Рунге. """

    n = 2
    I_n = simpson(a, b, n)
    n *= 2
    I_2n = simpson(a, b, n)
    while abs(I_n - I_2n) / 15 > eps:
        n *= 2
        I_n = I_2n
        I_2n = simpson(a, b, n)
    return I_2n, n
```



```
Точное значение интеграла: 0.943700

Метод центральных прямоугольников: 0.942826 при n = 64
Метод трапеций: 0.944142 при n = 128
Метод трапеций с правилом Рунге: 0.944142 при n = 128
Метод Симпсона с правилом Рунге: 0.944317 при n = 32

Сравнение:
Абсолютная ошибка центральных прямоугольников: 4.43e-04
Абсолютная ошибка трапеций: 4.43e-04
Абсолютная ошибка трапеций с Рунге: 4.43e-04
Абсолютная ошибка Симпсона с Рунге: 6.17e-04
```

Рисунок 1 – Результат работы программы

## Вывод

В ходе работы были изучены и реализованы численные методы вычисления определённых интегралов: метод центральных прямоугольников, метод трапеций, метод трапеций с использованием правила Рунге и метод Симпсона с правилом Рунге. Для интегрирования заданной функции на отрезке [0,5] с точностью $\varepsilon = 10^{-3}$ были получены численные результаты, а также рассчитаны абсолютные ошибки относительно точного значения интеграла. Анализ показал, что методы с использованием правила Рунге (особенно метод Симпсона) обеспечивают более высокую точность при меньшем числе разбиений по сравнению с базовыми методами. Метод центральных прямоугольников и метод трапеций сходятся медленнее, но тоже достигают требуемой точности при достаточно большом числе разбиений. Таким образом, использование адаптивных методов (с контролем погрешности) позволяет оптимизировать расчёты и получать высокоточные результаты при меньших вычислительных затратах.