Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет ПИиКТ

Дисциплина: Математический анализ

Лабораторная работа №2
Приблизительное вычисление интеграла с погрешностью ε = 0,00001 различными методами: прямоугольников, трапеций, Симпсона

Вариант 14

Выполнил: Михайлов Петр Сергеевич
Группа: Мат Ан Прод 11.4
Преподаватель: доцент, кандидат технических наук
Холодова Светлана Евгеньевна

Санкт-Петербург 2025г.

Содержание

Задание

Составить программу на основе формул прямоугольников, трапеций и Симпсона и, используя её, найти приближенно значение определённого интеграла с погрешностью ε.

Вариант №14: $\int_0^1 \cos x^3 dx$.

Ход работы

Вычисление методом прямоугольников

$$I \approx I^{\Pi} := h\sum_{i=1}^{n} f\left(x_{i-1}+\frac{h}{2}\right) = h\sum_{i=1}^{n} f\left(x_i-\frac{h}{2}\right).$$

$$r^{\Pi}(h) = \frac{b-a}{24} f''(\xi_{\Pi})h^2 \ , \qquad \xi_{\Pi} \in (a, b).$$

Программу для вычисления интеграла методом прямоугольника работает по тому принципу, что разбивает его n-ое число отрезков и вычисляет сумму площадей этих прямоугольников (сумма по значению середины каждого из отрезка разбиения).

Функция-метод по формуле прямоугольников

Составим функцию на языке программирования Python.

```python
# Метод прямоугольников
def rectangle_method(n):
    h = (b - a) / n
    total = 0.0
    for i in range(n):
        x_mid = a + (i + 0.5) * h
        total += f(x_mid)
    return h * total
```

Добавим проверку точности для этого метода.

```python
# Проверка точности для прямоугольников
h_rect = (b - a)/n_rect
error_rect = (b - a)/24 * M2 * h_rect**2
print("\n" + "="*75)
if error_rect > epsilon:
    print(f"Для метода прямоугольников (n = {n_rect}) теоретическая погрешность {error_rect:.2e} > ε!")
else:
    I_rect = rectangle_method(n_rect)
    print(f"Метод прямоугольников (n = {n_rect}): {I_rect:.10f}")
    print(f"Отклонение от эталона: {abs(I_rect - reference):.2e}")
```

Вычисление методом трапеций

$$I \approx I^{T} := h\left(\frac{y_0+y_n}{2}+y_1+y_2+\ldots+y_{n-1}\right),$$

$$r^{T} := I - I^{T} = -\frac{b-a}{12}h^2 f''(\xi_T), \ \ \xi_T \in (a, b).$$

Изменим код метода прямоугольников. В данном методе вычисления значений при очередном разбиении n будут начинаться не с того, что мы берем центральное значение отрезка, а полусумму двух крайних значений.

Функция-метод по формуле трапеций

Составим функцию на языке программирования Python.

```python
# Метод трапеций
def trapezoid_method(n):
    h = (b - a) / n
    total = (f(a) + f(b)) / 2
    for i in range(1, n):
        total += f(a + i * h)
    return h * total
```

Добавим проверку точности для этого метода.

```python
# Проверка точности для трапеций
h_trap = (b - a)/n_trap
error_trap = (b - a)/12 * M2 * h_trap**2
print("\n" + "="*75)
if error_trap > epsilon:
    print(f"Для метода трапеций (n={n_trap}) теоретическая погрешность {error_trap:.2e} > ε!")
else:
    I_trap = trapezoid_method(n_trap)
    print(f"Метод трапеций (n = {n_trap}): {I_trap:.10f}")
    print(f"Отклонение от эталона: {abs(I_trap - reference):.2e}")
```

Вычисление методом Симпсона

$$I \approx I^C := \frac{h}{3}(y_0 + y_{2m} + 4\sigma_1 + 2\sigma_2),$$

$$r^C := I - I^C = -\frac{h^4}{180}\sum_{i=1}^{m} 2h\, f^{(4)}(\xi_i) = -\frac{b-a}{180} h^4 f^{(4)}(\xi_C)$$

Вычисление методом Симпсона отличается от других тем, что ведется аппроксимация исходной функции по четным и нечетным узлам с определенными коэффициентами. На формуле это все отражено.

Функция-метод по формуле Симпсона

Составим функцию на языке программирования Python.

```python
# Метод Симпсона
def simpson_method(m):
    n = 2 * m
    h = (b - a) / n
    total = f(a) + f(b)
    # Сумма нечётных узлов
    for i in range(1, m + 1):
        total += 4 * f(a + (2 * i - 1) * h)
    # Сумма чётных узлов
    for i in range(1, m):
        total += 2 * f(a + 2 * i * h)
    return (h / 3) * total
```

Добавим проверку точности для этого метода.

```python
# Проверка точности для Симпсона
h_simp = (b - a)/(2*m_simp)
error_simp = (b - a)/180 * M4 * h_simp**4
print("\n" + "="*75)
if error_simp > epsilon:
    print(f"Для метода Симпсона (m = {m_simp}) теоретическая погрешность {error_simp:.2e} > ε!")
else:
    I_simp = simpson_method(m_simp)
    print(f"Метод Симпсона (m = {m_simp}): {I_simp:.10f}")
    print(f"Отклонение от эталона: {abs(I_simp - reference):.2e}")
```

Добавление оставшихся функций и вывода

Сделаем ввод пользователя для погрешности (по умолчанию 0,00001) и проверку ввода. Также сделаем ввод количества разбиений для каждого метода и вывод работы программы.

Полная программа выглядит так:

```python
import numpy as np
import sympy as sp

def f(x):
    return np.cos(x**3)

a, b = 0, 1
reference = 0.931704440591544

# Вычисление производных с помощью SymPy
x = sp.Symbol('x')
f_sym = sp.cos(x**3)

# Вторая производная
f2_sym = sp.diff(f_sym, x, 2)
f2_lambda = sp.lambdify(x, f2_sym, 'numpy')

# Четвертая производная
f4_sym = sp.diff(f_sym, x, 4)
f4_lambda = sp.lambdify(x, f4_sym, 'numpy')
```

```python
# Находим максимумы производных на [0, 1]
x_vals = np.linspace(0, 1, 1000)
M2 = max(abs(f2_lambda(x_vals)))
M4 = max(abs(f4_lambda(x_vals)))

# Функция для ввода чисел с плавающей точкой
def input_epsilon():
    while True:
        try:
            value = input("Введите точность ε (формат: 1e-5, 0.00001; по умолчанию 
1e-5): ").strip()
            if not value:
                return 1e-5
            return float(value)
        except ValueError:
            print("Ошибка: введите число в формате 0.0001 или 1e-4!")

# Функция для безопасного ввода целых чисел
def input_int(prompt):
    while True:
        try:
            value = int(input(prompt))
            if value <= 0:
                raise ValueError
            return value
        except ValueError:
            print("Ошибка: введите целое положительное число!")

# Ввод параметров
epsilon = input_epsilon()
print(f"\nУстановленная точность: ε = {epsilon:.0e}" if epsilon < 0.001 else 
f"\nУстановленная точность: ε = {epsilon}")

n_rect = input_int("Введите количество интервалов n для метода прямоугольников: ")
n_trap = input_int("Введите количество интервалов n для метода трапеций: ")
m_simp = input_int("Введите количество пар интервалов m для метода Симпсона (n = 
2m): ")

# Метод прямоугольников
def rectangle_method(n):
    h = (b - a) / n
    total = 0.0
    for i in range(n):
        x_mid = a + (i + 0.5) * h
        total += f(x_mid)
    return h * total

# Проверка точности для прямоугольников
h_rect = (b - a)/n_rect
error_rect = (b - a)/24 * M2 * h_rect**2
print("\n" + "="*75)
if error_rect > epsilon:
    print(f"Для метода прямоугольников (n = {n_rect}) теоретическая погрешность 
{error_rect:.2e} > ε!")
else:
    I_rect = rectangle_method(n_rect)
    print(f"Метод прямоугольников (n = {n_rect}): {I_rect:.10f}")
    print(f"Отклонение от эталона: {abs(I_rect - reference):.2e}")

# Метод трапеций
def trapezoid_method(n):
    h = (b - a) / n
    total = (f(a) + f(b)) / 2
    for i in range(1, n):
```

```python
        total += f(a + i * h)
    return h * total

# Проверка точности для трапеций
h_trap = (b - a)/n_trap
error_trap = (b - a)/12 * M2 * h_trap**2
print("\n" + "="*75)
if error_trap > epsilon:
    print(f"Для метода трапеций (n={n_trap}) теоретическая погрешность
{error_trap:.2e} > ε!")
else:
    I_trap = trapezoid_method(n_trap)
    print(f"Метод трапеций (n = {n_trap}): {I_trap:.10f}")
    print(f"Отклонение от эталона: {abs(I_trap - reference):.2e}")

# Метод Симпсона
def simpson_method(m):
    n = 2 * m
    h = (b - a) / n
    total = f(a) + f(b)
    # Сумма нечётных узлов
    for i in range(1, m + 1):
        total += 4 * f(a + (2 * i - 1) * h)
    # Сумма чётных узлов
    for i in range(1, m):
        total += 2 * f(a + 2 * i * h)
    return (h / 3) * total

# Проверка точности для Симпсона
h_simp = (b - a)/(2*m_simp)
error_simp = (b - a)/180 * M4 * h_simp**4
print("\n" + "="*75)
if error_simp > epsilon:
    print(f"Для метода Симпсона (m = {m_simp}) теоретическая погрешность
{error_simp:.2e} > ε!")
else:
    I_simp = simpson_method(m_simp)
    print(f"Метод Симпсона (m = {m_simp}): {I_simp:.10f}")
    print(f"Отклонение от эталона: {abs(I_simp - reference):.2e}")

# Итоговое сравнение (Вывод эталонного значения)
print("\n" + "="*75)
print(f"\nЭталонное значение: {reference:.15f}")
```

Вывод работы программы для ε = 0,00001

```
Введите точность ε (формат: 1e-5, 0.00001; по умолчанию 1e-5):

Установленная точность: ε = 1e-05
Введите количество интервалов n для метода прямоугольников: 115
Введите количество интервалов n для метода трапеций: 333
Введите количество пар интервалов m для метода Симпсона (n = 2m): 10

===========================================================================
Для метода прямоугольников (n = 115) теоретическая погрешность 3.12e-05 > ε!

===========================================================================
Метод трапеций (n = 333): 0.9317025435
Отклонение от эталона: 1.90e-06

===========================================================================
Метод Симпсона (m = 10): 0.9317040174
Отклонение от эталона: 4.23e-07

===========================================================================

Эталонное значение: 0.931704440591544
Press any key to continue . . .
```

Как видно, программа, если верно выбрано количество интервалов для определённого метода, выводит приближенное значение, а также отклонение от эталонного значения (в примере выше – методы трапеций и Симпсона). Но если для заданном ε выбрано слишком малое количество разбиений, то программа выводит погрешность и указывает на то, что она больше ε (в примере выше – метод прямоугольников).

Заключение

В результате выполнения лабораторной работы, я познакомился с методами по приблизительному вычислению интеграла, научился реализовывать эти методы с помощью языка программирования Python.

Литература

1) Математический анализ [Электронный ресурс]: Конспект лекций по математическому анализу на платформе Notion. – Режим доступа: https://clck.ru/3FC9Hk (дата обращения: 12.12.2024).
2) Т.В. Родина, Е.С. Трифанова Курс лекций по математическому анализу - I (для напр. «Прикладная математика и информатика»). Учебное пособие. – СПб: СПбГУ ИТМО, 2010. –183с. – Режим доступа: https://books.ifmo.ru/file/pdf/649.pdf (дата обращения: 12.12.2024).
3) Т.В. Родина, Е.С.Трифанова Задачи и упражнения по математическому анализу I (для спец. «Прикладная математика и информатика»). Учебное пособие. – СПб: СПбГУ ИТМО, 2011. –208с. – Режим доступа: https://books.ifmo.ru/file/pdf/835.pdf (дата обращения 12.12.2024).

Приложение

1) Ссылка на репозиторий GitHub, содержащий исходные коды всех составленных программ:

   https://clck.ru/3LafoB

2) Для сравнения прилагаю приблизительное значение интеграла, найденное с помощью калькулятора:

$$x \approx 0.9317044405915442260769263906807884349 93\ \ldots$$

   Синим выделены первые 15 цифр после запятой.