Федеральное государственное автономное

образовательное учреждение высшего образования

Национальный Исследовательский Университет ИТМО

Лабораторная работа 3

«Численное интегрирование»

Дисциплина: Вычислительная математика

Вариант 13

Выполнил: Терехин Никита Денисович

Факультет: Программной инженерии и компьютерной техники

Группа: P3208

Преподаватель: Машина Екатерина Алексеевна

г. Санкт-Петербург, 2024 год

Оглавление

Цель работы

Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами

Текст задания

Исходные данные:

1. Пользователь выбирает функцию, интеграл которой требуется вычислить (3-5 функций), из тех, которые предлагает программа.
2. Пределы интегрирования задаются пользователем.
3. Точность вычисления задается пользователем.
4. Начальное значение числа разбиения интервала интегрирования: n=4.
5. Ввод исходных данных осуществляется с клавиатуры.

Программная реализация задачи:

1. Реализовать в программе методы по выбору пользователя:
    a. Метод прямоугольников (3 модификации: левые, правые, средние)
    b. Метод трапеций
    c. Метод Симпсона
2. Методы должны быть оформлены в виде отдельной(ого) функции/класса.
3. Вычисление значений функции оформить в виде отдельной(ого) функции/класса.
4. Для оценки погрешности и завершения вычислительного процесса использовать правило Рунге.
5. Предусмотреть вывод результатов: значение интеграла, число разбиения интервала интегрирования для достижения требуемой точности.

Вычислительная реализация задачи:

1. Вычислить интеграл, приведенный в таблице 1, точно.
2. Вычислить интеграл по формуле Ньютона – Котеса при $n = 6$.
3. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при $n = 10$.
4. Сравнить результаты с точным значением интеграла.
5. Определить относительную погрешность вычислений для каждого метода.

6. В отчете отразить последовательные вычисления.

Рабочие формулы методов

Методы прямоугольников

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} y_{i-1} - \text{Левые}$$

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} y_{i} - \text{Правые}$$

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} f(\frac{x_i + x_{i-1}}{2}) - \text{Средние}$$

Метод трапеций

$$\int_a^b f(x)dx = h\left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i\right)$$

Метод Симпсона

$$\int_a^b f(x)dx = \frac{h}{3}\left(y_0 + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2}) + y_n\right)$$

Программная реализация

Листинг программы

```
# классы методов

class Method(Describable):
   option_name = 'method'

   def __init__(self, description: str):
       super().__init__(description)
       self.function: Callable[[float], float] | None = None
       self.partition: int = 4
       self.prev_value: float = math.inf
       self.accuracy_order: int = 2

   @abstractmethod
   def calculate_integral(self, a: float, b: float, eps: float) ->
float:
       pass

   def check_end_condition(self, a: float, b: float, ans: float,
eps: float) -> float:
```

```
        if abs((ans - self.prev_value) / (2 ** self.accuracy_order
- 1)) < eps:
            return ans
        self.prev_value = ans
        self.partition *= 2
        return self.calculate_integral(a, b, eps)

    def set_function(self, function: Callable[[float], float]) ->
None:
        self.function = function


class RectangleMethod(Method):
    def __init__(self, description: str):
        super().__init__(description)

    @abstractmethod
    def get_interval_partition(self, a: float, b: float) ->
list[float]:
        pass

    def calculate_integral(self, a: float, b: float, eps: float) ->
float:
        h: float = (b - a) / self.partition
        x: list[float] = self.get_interval_partition(a, b)
        if self.function is not None:
            y: list[float] = [self.function(num) for num in x]
            if math.nan in y:
                raise TypeError("Integral is undefined on this
interval")
        else:
            raise TypeError('Function is not defined')
        ans: float = sum(y) * h
        return self.check_end_condition(a, b, ans, eps)


class LeftRectangleMethod(RectangleMethod):
    def __init__(self):
        super().__init__('Left Rectangle Method')

    def get_interval_partition(self, a: float, b: float) ->
list[float]:
        h: float = (b - a) / self.partition
        return [a + h * i for i in range(0, self.partition)]


class RightRectangleMethod(RectangleMethod):
    def __init__(self):
```

```
        super().__init__('Right Rectangle Method')

    def get_interval_partition(self, a: float, b: float) -> 
list[float]:
        h: float = (b - a) / self.partition
        return [a + h * i for i in range(1, self.partition + 1)]


class MiddleRectangleMethod(RectangleMethod):
    def __init__(self):
        super().__init__('Middle Rectangle Method')

    def get_interval_partition(self, a: float, b: float) -> 
list[float]:
        h: float = (b - a) / self.partition
        return [a + h * i + h / 2 for i in range(0, 
self.partition)]


class TrapezeMethod(Method):
    def __init__(self):
        super().__init__('Trapeze Method')

    def calculate_integral(self, a: float, b: float, eps: float) -> 
float:
        h: float = (b - a) / self.partition
        x: list[float] = [a + h * i for i in range(self.partition + 
1)]
        if self.function is not None:
            y: list[float] = [self.function(num) for num in x]
            if math.nan in y:
                raise TypeError("Integral is undefined on this 
interval")
            if math.nan in y:
                raise ValueError("Integral is undefined on this 
interval")
        else:
            raise TypeError('Function is not defined')
        ans: float = (sum(y) - (y[0] + y[-1]) / 2) * h
        return self.check_end_condition(a, b, ans, eps)


class SimpsonsMethod(Method):
    def __init__(self):
        super().__init__("Simpson's Method")
        self.accuracy_order = 4
```

```python
    def calculate_integral(self, a: float, b: float, eps: float) -> float:
        h: float = (b - a) / self.partition
        x: list[float] = [a + h * i for i in range(0, self.partition + 1)]
        if self.function is not None:
            y: list[float] = [self.function(num) for num in x]
            if math.nan in y:
                raise TypeError("Integral is undefined on this interval")
        else:
            raise TypeError('Function is not defined')
        ans: float = (4 * sum(y[1:-1:2]) + 2 * sum(y[2:-1:2]) + y[0] + y[-1]) * h / 3
        return self.check_end_condition(a, b, ans, eps)


METHODS: Final[list[Method]] = [
    LeftRectangleMethod(),
    RightRectangleMethod(),
    MiddleRectangleMethod(),
    TrapezeMethod(),
    SimpsonsMethod()
]
```

```python
# класс функций

class Integral(Describable):
    option_name = 'function to calculate integral'

    def __init__(self, description: str, function: Callable[[float], float]):
        super().__init__(description)
        self.function: Callable[[float], float] = function


def condition_function(x: float) -> float:
    if x <= -2:
        return -2 / x
    if x <= 2:
        return abs(x) - 1
    return (x - 2)**0.5 + 1


INTEGRALS: Final[list[Integral]] = [
    Integral('-2x^3 - 5x^2 + 7x - 13', lambda x: -2 * x**3 - 5 * x**2 + 7 * x - 13),
```

```
    Integral('atan(sqrt(6x - 1))', lambda x: math.atan((6 * x -
1)**0.5) if 6 * x > 1 else math.nan),
    Integral('cos(x) / (e^x + 4)', lambda x: math.cos(x) /
(math.e**x + 4)),
    Integral('tan(sqrt(x)) / sqrt(x)', lambda x: math.tan(x**0.5) /
x**0.5 if x > 0 else math.nan),
    Integral('| -2 / x ,  x <= -2\n   | |x| - 1 ,  -2 < x <= 2\n
| sqrt(x - 2) + 1 ,  x > 2', condition_function)
]
```

```
if __name__ == '__main__':
    reader: AbstractReader = ConsoleReader('Read integral')
    integral: Integral = request_from_list(INTEGRALS)
    b_lim, t_lim, precision = reader.read_tuple('Input integral
limits using two numbers: ')
    method: Method = request_from_list(METHODS)
    method.set_function(integral.function)
    try:
        ans: float = method.calculate_integral(b_lim, t_lim,
precision)
        print('Calculated answer is: ', round(ans,
get_precision(precision)))
        print('Partition intervals number: ', method.partition)
    except (ZeroDivisionError, TypeError) as e:
        print(e)
        print('Please, try again')
```

Результаты работы программы

```
1. -2x^3 - 5x^2 + 7x - 13
2. atan(sqrt(6x - 1))
3. cos(x) / (e^x + 4)
4. tan(sqrt(x)) / sqrt(x)
5. | -2 / x ,  x <= -2
   | |x| - 1 ,  -2 < x <= 2
   | sqrt(x - 2) + 1 ,  x > 2
Choose function to calculate integral:
3
Input integral limits using two numbers:
-5 5
Input precision:
0.001
1. Left Rectangle Method
2. Right Rectangle Method
3. Middle Rectangle Method
4. Trapeze Method
5. Simpson's Method
```

```
Choose method:
4
Calculated answer is:  -0.17625
Partition intervals number:  64
```

```
1. -2x^3 - 5x^2 + 7x - 13
2. atan(sqrt(6x - 1))
3. cos(x) / (e^x + 4)
4. tan(sqrt(x)) / sqrt(x)
5. | -2 / x ,  x <= -2
   | |x| - 1 ,  -2 < x <= 2
   | sqrt(x - 2) + 1 ,  x > 2
Choose function to calculate integral:
4
Input integral limits using two numbers:
-5 5
Input precision:
0.0001
1. Left Rectangle Method
2. Right Rectangle Method
3. Middle Rectangle Method
4. Trapeze Method
5. Simpson's Method
Choose method:
5
Integral is undefined on this interval
Please, try again
```

Вычислительная реализация

Интеграл для вычислительной части лабораторной работы

13	$\int_{1}^{3}(2x^3 - 5x^2 + 7x - 13)\,dx$

С использованием формулы Ньютона-Лейбница

$$I_{\text{точн}} = \int_{1}^{3}(2x^3 - 5x^2 + 7x - 13)\,dx = \left(\frac{1}{2}x^4 - \frac{5}{3}x^3 + \frac{7}{2}x^2 - 13x\right)\Big|_1^3 =$$

$$= -\frac{4}{3} = -1.(3)$$

По формуле Ньютона-Котеса

$$\int_a^b f(x)dx \approx \int_a^b L(x)dx = \sum_{i=0}^{n} c_n^i f(x_i) \qquad n = 6$$

$$h = \frac{3-1}{6} = \frac{1}{3}$$

$$I_{cotes} = \int_1^3 f(x)dx \approx c_6^0 f(x_0) + c_6^1 f(x_1) + c_6^2 f(x_2) + c_6^3 f(x_3) + c_6^4 f(x_4) +$$

$$+ c_6^5 f(x_5) + c_6^6 f(x_6) = (\frac{3-1}{840})(41(f(1) + f(3)) + 216(f(\frac{4}{3}) + f(\frac{8}{3})) +$$

$$+ 27(f(\frac{5}{3}) + f(\frac{7}{3})) + 272f(2)) =- 1.3333333333333415$$

По формуле средних прямоугольников

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} f(\frac{x_i+x_{i-1}}{2}) \qquad n = 10$$

$$h = \frac{3-1}{10} = \frac{1}{5}$$

$$I_{прям} = \int_1^3 f(x)dx = h\,(f(11/10) + f(13/10) + f(15/10) + f(17/10) +$$

$$+ f(19/10) + f(21/10) + f(23/10) + f(25/10) + f(27/10) + f(29/10)) =$$

$$= - 1.38$$

По формуле трапеций

$$\int_a^b f(x)dx = h\left(\frac{y_0+y_n}{2} + \sum_{i=1}^{n-1} y_i\right) \qquad n = 10$$

$$I_{трап} = \int_1^3 f(x)dx = h(\frac{f(1)+f(3)}{2} + f(6/5) + f(7/5) + f(8/5) + f(9/5) +$$

$$+ f(2) + f(11/5) + f(12/5) + f(13/5) + f(14/5)) = - 1.24$$

По формуле Симпсона

$$\int_a^b f(x)dx = \frac{h}{3}\left(y_0 + 4(y_1 + y_3 +\ldots+ y_{n-1}) + 2(y_2 + y_4 +\ldots+ y_{n-2}) + y_n\right)$$

$$I_{simp} = \int_1^3 f(x)dx = \frac{h}{3}\,(f(1) + 4(f(6/5) + f(8/5) + f(2 + f(12/5) +$$

$$+ f(14/5)) + 2(f(7/5) + f(9/5) + f(11/5) + f(13/5)) + f(3)) =$$

$= \ -1.3333333333333357$

Погрешности методов

$\Delta I = I_{\text{точн}} - I_{cotes} = 8.22e - 15 \qquad \varepsilon = (6.16e - 13)\%$

$\Delta I = I_{\text{точн}} - I_{\text{прям}} = 0.04(6) \qquad \varepsilon = 3.5\%$

$\Delta I = I_{\text{точн}} - I_{\text{трап}} = 0.09(3) \qquad \varepsilon = 7\%$

$\Delta I = I_{\text{точн}} - I_{simp} = 2.44e - 15 \qquad \varepsilon = (1.83e - 13)\%$

Выводы

В ходе выполнения работы удалось найти приближенное значение определенного интеграла с требуемой точностью методами Ньютона-Котеса, Симпсона, трапеций и прямоугольников, а также реализовать их программно
В конечном итоге получилось, что самым точным является метод Котеса, что неудивительно, ведь это обобщение всех остальных методов
Метод Симпсона при этом имеет наименьшую погрешность, однако это обусловлено большей плотностью разбиения в сравнении с методом Котеса
Метод трапеций имеет наивысшую погрешность. К плюсам данного метода можно отнести простоту. При увеличении количества интервалов нет необходимости пересчитывать старые значения функции