Федеральное государственное
автономное учебное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники

Отчёт
по лабораторной работе №3
по дисциплине «Вычислительная математика»

Вариант №4

Группа: P3218

Студент: Горло Евгений Николаевич

Преподаватель: Бострикова Дарья Константиновна

Санкт-Петербург
2024

Содержание

1 Цель лабораторной работы

Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

2 Задание

Программная реализация

1. Реализовать в программе методы по выбору пользователя:
   - Метод прямоугольников (3 модификации: левые, правые, средние)
   - Метод трапеций
   - 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

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.

Интеграл из варианта:

$$\int_{-3}^{-1} \left(-2x^3 - 4x^2 + 8x - 4\right)\, dx$$

3 Рабочие формулы методов

Метод прямоугольников:

На каждом шаге интегрирования функция аппроксимируется полиномом нулевой степени - отрезком, параллельным оси абсцисс. Площадь криволинейной трапеции приближенно заменяется площадью многоугольника, составленного из n - прямоугольников. Таким образом, вычисление определенного интеграла сводится к нахождению суммы n-элементарных прямоугольников.

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} yi - 1$$

$h = (b-a)/n$
$x_i = a + hi$ для правых
$x_i = a + hi_{-1}$ для левых
$x_i = a + hi_{-1} + h/2$ для центральных

Метод трапеции:

Подынтегральную функцию на каждом отрезке $[x_i; x_{i+1}]$ заменяют интерполяционным многочленом первой степени: $f(x) = ax + b$ Используют линейную интерполяцию, т.е. график функции $y = f(x)$ представляется в виде ломаной, соединяющий точки (x_i, y_i)

$$\int_a^b f(x)\,dx = h((y_0 + y_n)/2 + \sum_{i=1}^{n} y_{i-1})$$

$h = (b-a)/n$
$x_i = a + h_i$

Метод Симпсона:

Метод Симпсона - численный метод для приближенного вычисления определенных интегралов. Он основан на аппроксимации подынтегральной функции квадратичной функцией на каждом интервале интегрирования. Для данного отрезка $[a, b]$ с равномерной сеткой, где n - четное число подотрезков, метод Симсона выражается формулой:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}[f(a) + 4\sum_{i-1}^{n/2} f(x_{2i-1}) + 2\sum_{i-1}^{n/2-1} f(x_{2i}) + f(b)]$$

где h = $\frac{b-a}{n}$ - шаг интегрирования, $x_i = a + ih$ - узлы сетки.

4 Листинг программы

Ссылка на Github репозиторий

Метод прямоугольников (левый):

```
def rectangle_method_left(quation, a, b, e, parts):
    integral = 99999
    answer = []

    while integral > e and parts < 10000000:
        h = (b - a) / parts
        h2 = (b - a) / (parts // 2)

        integral_1 = sum(integrand(quation, a + (i - 1) * h) for i in range(parts
            ))
        integral_2 = sum(integrand(quation, a + (i - 1) * h2) for i in range(
            parts // 2))

        integral_1 *= h
        integral_2 *= h2

        integral = abs((integral_2 - integral_1) / (2 ** 2 - 1))
        parts *= 2

    if integral_1 < float("inf"):
        answer.append(integral_1)
        answer.append(parts // 2)
        answer.append(integral)
        return answer
    else:
        return "The integral doesn't converge."
```

Метод прямоугольников (средний):

```
def rectangle_method_centre(quation, a, b, e, parts):
    integral = 99999
    answer = []

    while integral > e and parts < 10000000:
        h = (b - a) / parts
        h2 = (b - a) / (parts // 2)

        integral_1 = sum(integrand(quation, a + (i - 1 + h / 2) * h) for i in
            range(parts))
        integral_2 = sum(integrand(quation, a + (i - 1 + h2 / 2) * h2) for i in
            range(parts // 2))

        integral_1 *= h
        integral_2 *= h2

        integral = abs((integral_2 - integral_1) / (2 ** 2 - 1))
        parts *= 2

    if integral_1 < float("inf"):
        answer.append(integral_1)
        answer.append(parts // 2)
        answer.append(integral)
        return answer
    else:
        return "The integral doesn't converge."
```

Метод прямоугольников (правый):

```
def rectangle_method_right(quation, a, b, e, parts):
    integral = 99999
    answer = []
    while integral > e and parts < 10000000:
        h = (b - a) / parts
        h2 = (b - a) / (parts // 2)

        integral_1 = sum(integrand(quation, a + (i) * h) for i in range(parts))
        integral_2 = sum(integrand(quation, a + (i) * h2) for i in range(parts //
            2))

        integral_1 *= h
        integral_2 *= h2

        integral = abs((integral_2 - integral_1) / (2 ** 2 - 1))
        parts *= 2

    if integral_1 < float("inf"):
        answer.append(integral_1)
        answer.append(parts // 2)
        answer.append(integral)
        return answer
    else:
        return "The integral doesn't converge."
```

Метод трапеций:

```
def trapezoidal_method(quation, a, b, e, parts):
    integral = 99999
    answer = []

    while integral > e and parts < 10000000:
        i = 1
        h = (b - a) / parts
        h2 = (b - a) / (parts // 2)
        integral_1 = 0.5 * (
                integrand(quation, a) + integrand(quation, b)
        )
        integral_2 = 0.5 * (
                integrand(quation, a) + integrand(quation, b)
        )

        for i in range(parts // 2):
            integral_2 += integrand(quation, a + i * h2)
        integral_2 *= h2
        i = 1

        for i in range(parts):
            integral_1 += integrand(quation, a + i * h)
        integral_1 *= h

        integral = abs((integral_2 - integral_1) / (2 ** 2 - 1))
        parts *= 2

    if integral_1 < float("inf"):
        answer.append(integral_1)
        answer.append(parts // 2)
        answer.append(integral)
        return answer
```

```
    else:
        return "The integral doesn't converge."
```

Метод Симпсона:

```
def simpson_method(quation, a, b, e, parts):
    integral = 99999
    answer = []

    while integral > e and parts < 10000000:
        h = (b - a) / parts
        h2 = (b - a) / (parts // 2)
        integral_1 = 0
        integral_2 = 0
        x_values = [a + i * h for i in range(parts + 1)]
        x_values_2 = [a + i * h2 for i in range(parts // 2 + 1)]

        integral_2 = integrand(quation, a) + integrand(quation, b)
        for i in range(1, parts // 2, 2):
            integral_2 += 4 * integrand(quation, x_values_2[i])

        for i in range(2, parts // 2 - 1, 2):
            integral_2 += 2 * integrand(quation, x_values_2[i])
        integral_2 *= h2 / 3

        integral_1 = integrand(quation, a) + integrand(quation, b)
        for i in range(1, parts, 2):
            integral_1 += 4 * integrand(quation, x_values[i])
        for i in range(2, parts - 1, 2):
            integral_1 += 2 * integrand(quation, x_values[i])
        integral_1 *= h / 3
        integral = abs((integral_2 - integral_1) / (2 ** 2 - 1))
        parts *= 2

    if integral_1 < float("inf"):
        answer.append(integral_1)
        answer.append(parts // 2)
        answer.append(integral)
        return answer
    else:
        return "The integral doesn't converge."
```

5 Результаты выполнения программы

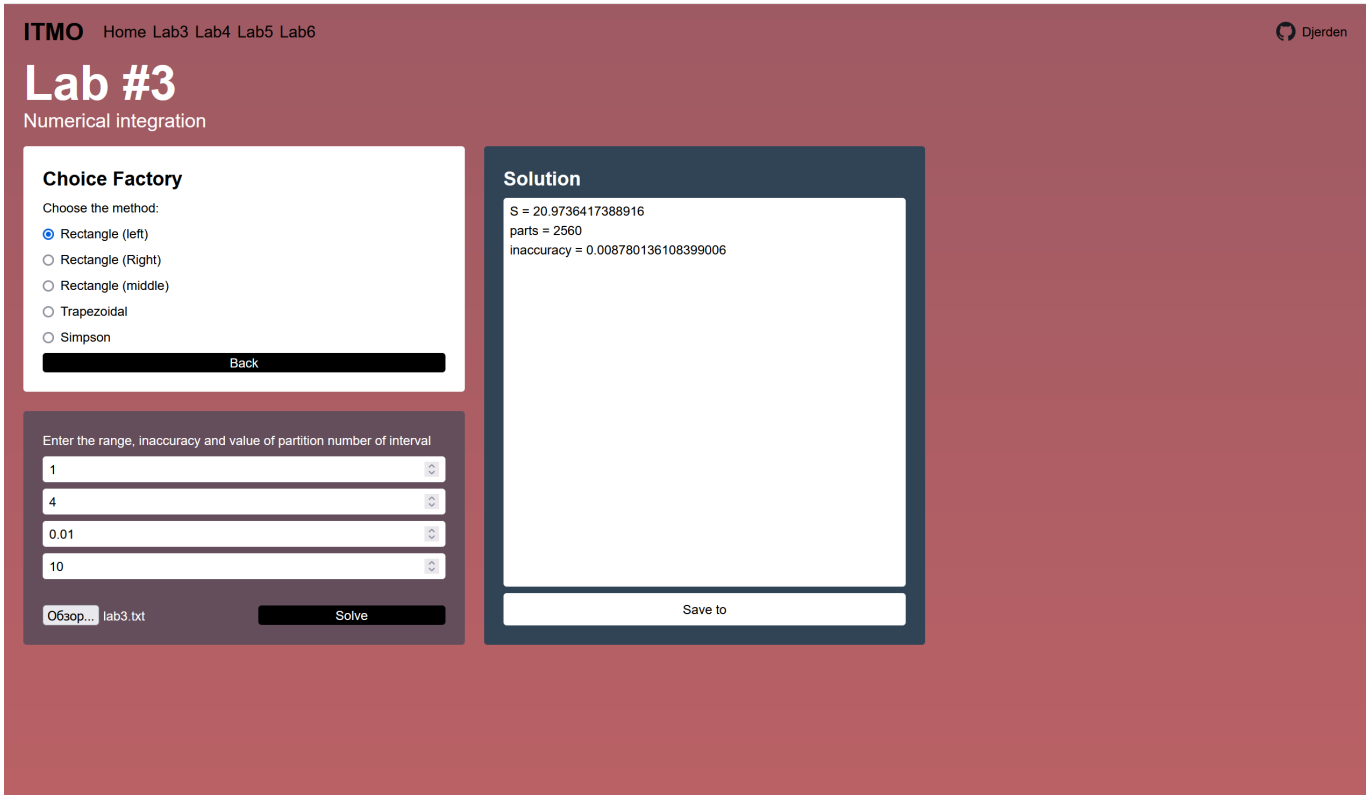

Рис. 1: Пример работы программы

6 Вычисление заданного интеграла

$$\int_{-3}^{-1} \left(-2x^3 - 4x^2 + 8x - 4\right)\, dx$$

6.1 Точное вычисление интеграла:

Найдем первообразную подынтегральной функции:

$$F(x) = -\frac{1}{2}x^4 - \frac{4}{3}x^3 + 4x^2 - 4x$$

По формуле Ньютона-Лейбница вычислим определенный интеграл:
$F(-1) = -\frac{1}{2} + \frac{4}{3} + 4 + 4 = \frac{53}{6} \approx 8,84$
$F(-3) = -\frac{81}{2} + 72 + 12 = \frac{87}{2} \approx 43,5$
$F(-1) - F(-3) = 8,84 - 43,5 = -34,66667$
Ответ: $-34,66667$

6.2 Вычисление интеграла по формуле Ньютона-Котеса:

$n = 6$

Разбиение интервала и определение узлов:

Разбиваем интервал на 6 равных подотрезков, это нам даст 7 узлов (точек)

$x_0 = -3$
$x_1 = -2,6667$
$x_2 = -2,3333$
$x_3 = -2$
$x_4 = -1,6667$
$x_5 = -1,3333$
$x_6 = -1$

Вычисление значений функций в узлах:

$y_0 = f(-3) = -10$
$y_1 = f(-2,6667) = -15,8518$
$y_2 = f(-2,3333) = -19,0372$
$y_3 = f(-2) = -20$
$y_4 = f(-1,6667) = -19,1851$
$y_5 = f(-1,3333) = -17,037$
$y_6 = f(-1) = -14$

Вычисление коэффициентов Ньютона-Котеса для n = 6:

$c_6^0 = \frac{41(b-a)}{840} = \frac{82}{840} \approx 0.09761905$
$c_6^1 = \frac{216(b-a)}{840} = \frac{432}{840} \approx 0.51428571$
$c_6^2 = \frac{27(b-a)}{840} = \frac{54}{840} \approx 0.06428571$
$c_6^3 = \frac{272(b-a)}{840} = \frac{544}{840} \approx 0.64761905$
$c_6^4 = c_6^2$
$c_6^5 = c_6^1$
$c_6^6 = c_6^0$

Подстановка значений в формулу Ньютона-Котеса:

$$\int_{-3}^{-1} (-2x^3 - 4x^2 + 8x - 4)\,dx \approx \sum_{i=0}^{6} c_i \cdot f(x_i)$$

$$\frac{82}{840} \cdot (-10) + \frac{432}{840} \cdot (-15,8514) + \frac{54}{840} \cdot (-19,0372) + \frac{544}{840} \cdot (-20) + \frac{54}{840} \cdot (-19,1853) +$$
$$+ \frac{432}{840} \cdot (-17,0367) + \frac{82}{840} \cdot (-14) = -34.66667$$

Вычислим погрешность интеграла:

$\Delta I_n = I - I_n = 34,66667 - 34,66667 = 0 (\approx 0\%)$

Ответ: -34,66667

6.3 Вычисление интеграла по формуле средних прямоугольников:

$n = 10$
Разобьем отрезок интегрирования на 10 равных частей:
$h = \frac{b-a}{n} = 0,2$

i	1	2	3	4	5	6	7	8	9	10
$x_{i-\frac{1}{2}}$	-2,9	-2,7	-2,5	-2,3	-2,1	-1,9	-1,7	-1,5	-1,3	-1,1
$y_{i-\frac{1}{2}}$	55,218	42,926	32,25	23,094	15,362	8,958	3,786	-0,25	-3,246	-5,298

Вычислим значение интеграла:
$I_{avg} = h\sum_{i=1}^{n} y_{i-\frac{1}{2}} = 34,56$ **Вычислим погрешность интеграла:**
$\Delta I_{avg} = I - I_{avg} = 34,66667 - 34,56 = 0,10667(\approx 0,31\%)$
Ответ: 34,56

6.4 Вычисление интеграла по формуле трапеций:

$n = 10$
$h = \frac{b-a}{n} = 0,2$

i	0	1	2	3	4	5	6	7	8	9	10
$x_{i-\frac{1}{2}}$	-3	-2,8	-2,6	-2,4	-2,2	-2	-1,8	-1,6	-1,4	-1,2	-1
$y_{i-\frac{1}{2}}$	62	48,864	37,392	27,488	19,056	12	6,224	1,632	-1,872	-4,384	-6

Вычислим значение интеграла:
$I_{trap} = \frac{1}{2}\sum_{i=0}^{n} h_i(y_{i-1} + y_i) = 34,88$
Вычислим погрешность интеграла:
$\Delta I_{trap} = I - I_{trap} = 34,66667 - 34,88 = 0,021333(\approx 0,615\%)$
Ответ: 34,88

6.5 Вычисление интеграла по формуле Симпсона:

$n = 10$
$h = \frac{b-a}{n} = 0,2$

i	0	1	2	3	4	5	6	7	8	9	10
$x_{i-\frac{1}{2}}$	-3	-2,8	-2,6	-2,4	-2,2	-2	-1,8	-1,6	-1,4	-1,2	-1
$y_{i-\frac{1}{2}}$	62	48,864	37,392	27,488	19,056	12	6,224	1,632	-1,872	-4,384	-6

Вычислим значение интеграла:
$I_{sim} = \frac{h}{3}[(y_0 + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2}) + y_n)] = 34,66667$
$\Delta I_{sim} = I - I_{sim} = 34,66667 - 34,66667 = 0(\approx 0\%)$
Ответ: 34,66667

7 Вывод

В ходе выполнения лабораторной работы были исследованы и реализованы три численных метода для приближенного вычисления определенных интегралов: метод прямоугольников (левый, правый, средний), метод трапеций и метод Симпсона.

Метод прямоугольниов (в частности центральный) обеспечивает наименьшую точность приближенного вычисления интегралов сравнительно с методом трапеций и Симпсона. Это связано с тем, что при использовании метода прямоугольников аппроксимация подынтегральной функции происходит с помощью прямоугольников, что может приводить к значительной потери точности, особенно на функциях с большими изменениями.

Метод трапеций демонстриурет большую точность по сравнению с методом прямоугольников, так как использует трапеции для аппроксимации функций, что более точно приближает интеграл. Однако, он все еще может оказаться менее точным по сравнению с методом Симпсона.

Метод Симпсона, использующий квадратичные интерполяционные полиномы для аппроксимации функции, предоставляет наибольшую точность среди рассмотренных методов. Он обеспечивает хорошее приближение к интегралу даже на функциях с большими изменениями и устойчив к различным формам функций.