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

Мегафакультет компьютеных технологий и управления
Факультет программной инженерии и компьютерной техники

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

Вариант 5

Группа: P3218

Студент: Зарубов Егор Николаевич

Преподаватель: Бострикова Дарья Константиновна

Санкт-Петербург
2024

Содержание

1 б№3

$$\int_2^4 (-2x^3-3x^2+x+5)\,dx = -\frac{x^4}{2} - x^3 + \frac{x^2}{2} + 5x \Big|_2^4 =$$

$$= -\frac{4^4}{2} - 4^3 + \frac{4^2}{2} + 5\cdot 4 - \left(-\frac{2^4}{2} - 2^3 + \frac{2^2}{2} + 5\cdot 2\right) =$$

$$= -\frac{256}{2} - 64 + \frac{16}{2} + 20 + \frac{16}{2} + 8 - 2 - 10 = -160$$

Ньютона - Котеса $n = 6$

$h = \frac{b-a}{n} = \frac{2}{6} = \frac{1}{3}$

$x_i = a + i\cdot h$; $x_0 = 2$, $x_1 = 2\frac{1}{3}$, $x_2 = 2\frac{2}{3}$, $x_3 = 3$ $x_4 = 3\frac{1}{3}$ $x_5 = 3\frac{2}{3}$

$x_6 = 4$

$$\sum_{i=0}^{n} f(x_i)\, C_n^i = C_6^0 f(2) + C_6^1 f\left(2\tfrac{1}{3}\right) + C_6^2 f\left(2\tfrac{2}{3}\right) + C_6^3 f(3) +$$

$$+ C_6^4 f\left(3\tfrac{1}{3}\right) + C_6^5 f\left(3\tfrac{2}{3}\right) + C_6^6 f(4) = \frac{41(4-2)}{840}\cdot(-21) +$$

$$+ \frac{216(4-2)}{840}\cdot(-34,41) + \frac{27(4-2)}{840}\cdot(-51,59) + \frac{272(4-2)}{840}\cdot(-73) +$$

$$+ \frac{27(4-2)}{840}\cdot(-99,07) + \frac{216(4-2)}{840}\cdot(-130,26) + \frac{41(4-2)}{840}\cdot(-167) =$$

$$= -160,001$$

$$-160 + 160,001 = 0,01 \approx 0,006\ \%$$

Средние прямоугольники: $n=10$

$h=\frac{b-a}{n}=0,2$ $x_{1-1}+\frac{h}{2}=2,1$ $x_{2-1}+\frac{h}{2}=2,3$ $x_{3-1}+\frac{h}{2}=2,5$

$x_{4-1}+\frac{h}{2}=2,7$ $x_{5-1}+\frac{h}{2}=2,9$ $x_{6-1}+\frac{h}{2}=3,1$ $x_{7-1}+\frac{h}{2}=3,3$

$x_{8-1}+\frac{h}{2}=3,5$ $x_{9-1}+\frac{h}{2}=3,7$ $x_{10-1}+\frac{h}{2}=3,9$

$h\sum_{i=1}^{n} f\left(x_{i-1}+\frac{h}{2}\right)=h\big(f(2,1)+f(2,3)+f(2,5)+f(2,7)+f(2,9)+f(3,1)+$

$+f(3,3)+f(3,5)+f(3,7)+f(3,9)\big)=0,2\big(-24,65-32,90-$

$-42,5-53,54-66,11-80,31-96,244-114-133,68-$

$-155,37\big)=-159,86$

$-160+159,86=0,14\approx 0,088\%$

Трапеции

$h=\frac{4-2}{10}=0,2$ $x_0=2$ $x_1=2,2$ $x_2=2,4$ $x_3=2,6$ $x_4=2,8$

$x_4=3$ $x_5=3,2$ $x_6=3,4$ $x_7=3,8$ $x_9=4$

$\frac{h}{2}\left(y_0+y_n+2\sum_{i=1}^{n-1} y_i\right)=0,1\big(-21f-167+2\big(-28,62-$

$-37,53-47,83-59,62-73-88,06-104,88-123,59-$

$-144,26\big)=-160,278$

$-160+160,278=0,278\approx 0,174\%$

Симпсона: $n=10$

$h=\frac{4-2}{10}=0{,}2$ $x_0=2$ $x_1=2{,}2$ $x_2=2{,}4$ $x_3=2{,}6$ $x_4=2{,}8$

$x_4=3$ $x_5=3{,}2$ $x_6=3{,}4$ $x_7=3{,}8$ $x_9=4$

$\frac{h}{3}\left(y_0+4(y_1+y_3+y_5+y_7+y_9)+2(y_2+y_4+y_6+y_8)+y_{10}\right)=$

$=\frac{0{,}2}{3}\left(-21+4(-28{,}62-47{,}83-73-104{,}88-144{,}26)+\right.$

$\left.+2(-37{,}53-59{,}62-88{,}06-123{,}59)-167\right)=-159{,}997$

$-160+159{,}997=0{,}003\approx 0{,}00188$

1 Цель работы

Цель работы: найти приближенное значение определенного интеграла с требуемой точностью различными численными методами

2 Задание

Написать программу для решения СЛАУ с использованием метода Гаусса. Требования к программе:

- Реализовать в программе методы по выбору пользователя:
- Метод прямоугольников (3 модификации: левые, правые, средние)
- Метод трапеций
- Метод Симпсона
- Методы должны быть оформлены в виде отдельной(ого) функции/класса.
- Вычисление значений функции оформить в виде отдельной(ого) функции/класса.
- Для оценки погрешности и завершения вычислительного процесса использовать правило Рунге.
- Предусмотреть вывод результатов: значение интеграла, число разбиения интервала интегрирования для достижения требуемой точности.
- Установить сходимость рассматриваемых несобственных интегралов 2 рода (2-3 функции). Если интеграл - расходящийся, выводить сообщение: «Интеграл не существует».
- Если интеграл сходящийся, реализовать в программе вычисление несобственных интегралов 2 рода (заданными численными методами).
- Рассмотреть случаи, когда подынтегральная функция терпит

3 Исходный код программы

Репозиторий на GitHub: ссылка

```
def Rectangle_method_left(quation, left_border, right_border, inaccuracy, parts):
    I = 99999

    while I > inaccuracy and parts < 1000000:
        i = 1
        h = (right_border - left_border) / parts
        h2 = (right_border - left_border) / (parts // 2)
        integral = 0
        integral_2 = 0

        for i in range(parts // 2):
            integral_2 += integrand(quation, left_border + (i - 1) * h2)
        integral_2 *= h2
        i = 1
        for i in range(parts):
            integral += integrand(quation, left_border + (i - 1) * h)
        integral *= h
        I = (integral_2 - integral) / (2**2 - 1)
        parts *= 2
    print(f"S = {integral}\nParts = {parts//2}\nInnacuary = {I}")
```

Листинг 1: Метод Прямоугольника левый

```
def Rectangle_method_centre(quation, left_border, right_border, inaccuracy, parts):
    I = 99999

    while I > inaccuracy and parts < 1000000:
        i = 1
        h = (right_border - left_border) / parts
        h2 = (right_border - left_border) / (parts // 2)
        integral = 0
        integral_2 = 0

        for i in range(parts // 2):
            integral_2 += integrand(quation, left_border + (i - 1 ) * h2 + h2 / 2)
        integral_2 *= h2
        i = 1
        for i in range(parts):
            integral += integrand(quation, left_border + (i - 1 ) * h + h / 2)
        integral *= h
        I = abs((integral_2 - integral) / (2**2 - 1))
        parts *= 2
    print(f"S = {integral}\nParts = {parts//2}\nInnacuary = {I}")
```

Листинг 2: Метод Прямоугольника центр

```
def Rectangle_method_right(quation, left_border, right_border, inaccuracy, parts):
    I = 99999

    while I > inaccuracy and parts < 1000000:
        i = 1
        h = (right_border - left_border) / parts
        h2 = (right_border - left_border) / (parts // 2)
        integral = 0
        integral_2 = 0

        for i in range(parts // 2):
            integral_2 += integrand(quation, left_border + (i) * h2)
        integral_2 *= h2
        i = 1
        for i in range(parts):
            integral += integrand(quation, left_border + (i) * h)
        integral *= h
        I = abs((integral_2 - integral) / (2**2 - 1))
        parts *= 2
    print(f"S = {integral}\nParts = {parts//2}\nInnacuary = {I}")
```

Листинг 3: Метод Прямоугольника правый

```
def trapezoidal_method(quation, left_border, right_border, inaccuracy, parts):
    I = 99999

    while I > inaccuracy and parts < 1000000:
        i = 1
        h = (right_border - left_border) / parts
        h2 = (right_border - left_border) / (parts // 2)
        integral = 0.5 * (
            integrand(quation, left_border) + integrand(quation, right_border)
        )
        integral_2 = 0.5 * (
            integrand(quation, left_border) + integrand(quation, right_border)
        )

        for i in range(parts // 2):
            integral_2 += integrand(quation, left_border + i * h2)
        integral_2 *= h2
        i = 1
        for i in range(parts):
            integral += integrand(quation, left_border + i * h)
        integral *= h
        I = (integral_2 - integral) / (2**2 - 1)
        parts *= 2
    print(f"S = {integral}\nParts = {parts//2}\nInnacuary = {I}")
```

Листинг 4: Метод Трапеций

```
def simpson_method(quation, left_border, right_border, inaccuracy, parts):
    I = 99999

    while I > inaccuracy and parts < 1000000:
        # i = 1
        h = (right_border - left_border) / parts
        h2 = (right_border - left_border) / (parts // 2)
        integral = 0
        integral_2 = 0
        x_values = [left_border + i * h for i in range(parts + 1)]
        x_values_2 = [left_border + i * h2 for i in range(parts // 2 + 1)]

        integral_2 = integrand(quation, left_border) + integrand(quation, right_border
        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 = integrand(quation, left_border) + integrand(quation, right_border)
        for i in range(1, parts, 2):
            integral += 4 * integrand(quation, x_values[i])
        for i in range(2, parts - 1, 2):
            integral += 2 * integrand(quation, x_values[i])
        integral *= h / 3
        I = (integral_2 - integral) / (2**4 - 1)
        parts *= 2

        print(f"S = {integral}\nParts = {parts//2}\nInnacuary = {I}")
```

Листинг 5: Метод Симпсона

4 Рассчётные формулы

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.
$\int_a^b f(x)\,dx = \sum_{i=1}^n y_{i-1}$
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
$x_i = a + hi$

Метод Симпсона:

Метод Симпсона - это 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 Симпсона выражается формулой:

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

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

5 Пример вывода программы

(a) (b)

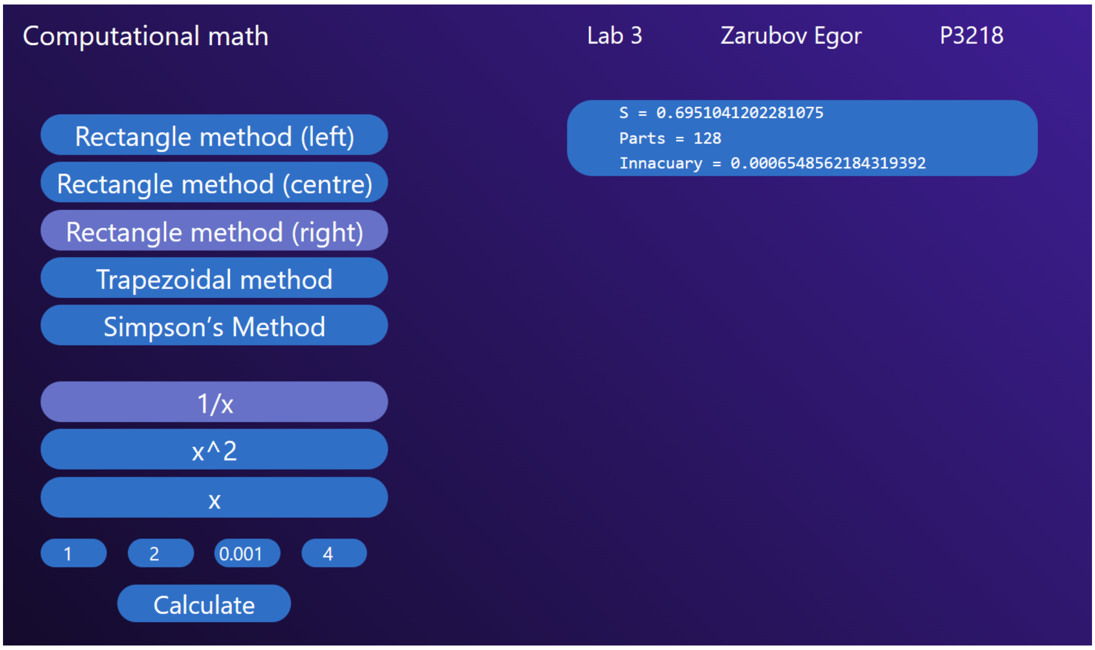

Рис. 1: Площадь методом правых прямоугольников

Рис. 2: Площадь методом симпсона

6 Вывод

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

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

Метод трапеций демонстрирует большую точность по сравнению с методом прямоугольников, так как использует трапеции для аппроксимации функции, 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.