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

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа **№3**
«Численное интегрирование»

по дисциплине «Вычислительная математика»

Вариант: **4**

Преподаватель:
Малышева Татьяна Алексеевна

Выполнил:
Касымов Тимур Шавкатович

Санкт-Петербург, 2024 г.

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

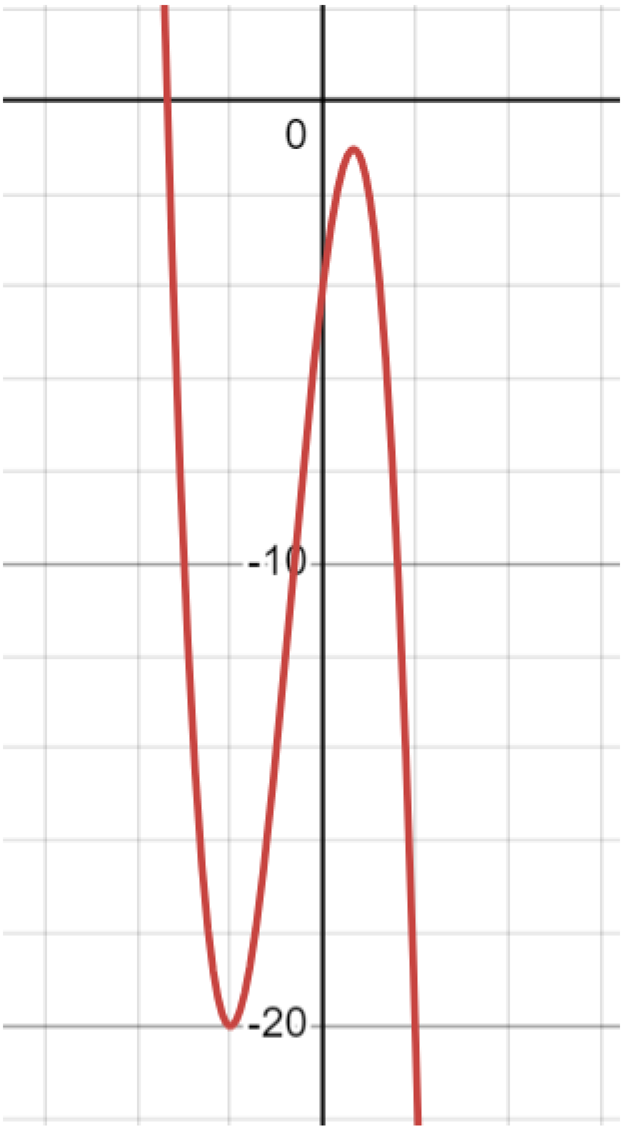
1. Вычислительная реализация задачи

1. **Вычислить интеграл**, приведенный в таблице 1, **точно:**

$$\int_{-3}^{-1} (-2 * x^3 - 4 * x^2 + 8 * x - 4)dx =$$

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

$$I_{точн} = F(-1) - F(-3) = -\frac{128}{3} + 8 = -\frac{104}{3}$$



2. **Вычислить интеграл по формуле Ньютона–Котеса** при $n = 6$:

$$\int_{-3}^{-1} (-2 * \mathrm{x}^3 - 4 * \mathrm{x}^2 + 8 * \mathrm{x} - 4)\mathrm{dx} =$$

$$\int_{-3}^{-1} \frac{n * h}{C_n} \sum_{i=0}^{n} c_n^i * f(x_i) =$$

$$\frac{5 * 0.4}{288} * \big(19 * f(-3) + 75 * f(-2.6) + 50 * f(-2.2) + 50 * f(-1.8) + 75 * f(-1.4) + 19 * f(-1)\big) =$$

$$= 0.00694 * 19 * (-10) + 75 * (-16.69) + 50 * (-19.664) + 50 * (-19.696) + 75 * (-17.552) + 19 * (-14) = -34.645521$$

3. **Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона** при $n = 10$:

$$h = \frac{b - a}{n} = \frac{-1 - (-3)}{10} = \frac{1}{5}$$

- **Метод средних прямоугольников**:

$$I_{\text{ср.прям}} = h \sum_{i=1}^{n} y_{i-\frac{1}{2}} = h \cdot \left(f\left(a + \frac{h}{2}\right) + f\left(a + \frac{3h}{2}\right) + f\left(a + \frac{5h}{2}\right) + f\left(a + \frac{7h}{2}\right) + f\left(a + \frac{9h}{2}\right) + f\left(a + \frac{11h}{2}\right) + f\left(a + \frac{13h}{2}\right) + f\left(a + \frac{15h}{2}\right) + f\left(a + \frac{17h}{2}\right) + f\left(a + \frac{19h}{2}\right)\right) =$$
$$= 0.2\big(f(-3 + 0.1) + f(-3 + 0.3) + f(-3 + 0.5) + f(-3 + 0.7) + f(-3 + 0.9) + f(-3 + 1.1) + f(-3 + 1.3) + f(-3 + 1.5) + f(-3 + 1.7) + f(-3 + 1.9)\big) = \mathbf{-34.62}$$

- **Метод трапеций**:

$$I_{\text{трапеция}} = h \cdot \left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i\right)$$
$$I_{\text{трапеция}} = 0.2\left(\frac{f(-3) + f(-1)}{2} + f(-3 + 0.2) + f(-3 + 0.4) + f(-3 + 0.6) + f(-3 + 0.8) + f(-3 + 1) + f(-3 + 1.2) + f(-3 + 1.4) + f(-3 + 1.6) + f(-3 + 1.8)\right) = \mathbf{-34.66}$$

- **Метод Симпсона**:

$$I_{\text{Симпсона}} = \frac{h}{3} \cdot \left(y_0 + \ 4 \sum_{i=1}^{n-1} y_{\text{нечёт}} + 2 \sum_{i=2}^{n-2} y_{\text{чёт}} + y_n \right)$$

$$I_{\text{Симпсона}} = \frac{0.2}{3}\big(f(-3) + 4 * \big(f(-3+0.2) + f(-3+0.6) + f(-3+1) + f(-3+1.4) + f(-3+1.8)\big) + 2 * (f(-3+0.4) + f(-3+0.8) + f(-3+1.2) + \ f(-3+1.6)) + f(-1)\big) = \mathbf{-34.66}$$

4. Сравнить результаты с точным значением интеграла:

Точное значение интеграла на интервале вычислено как -104/3

1. Для метода **Ньютона–Котеса** при $n = 6$: $I_{\text{точн}} = I_{cotes} = -34.645521$, **значения совпадают**.
   $R = |I_{\text{точн}} - I_{cotes}| = \left|\frac{-104}{3} - (-34.64)\right| = 0.026$
2. Для метода **средних прямоугольников** при $n = 10$: $I_{\text{ср.прям}} = \mathbf{-34.62}$.
   $R = \left|I_{\text{точн}} - I_{\text{ср.прям}}\right| = \left|\frac{-104}{3} - (\mathbf{-34.62})\right| = 0.04(6)$
3. Для метода **трапеций** при $n = 10$: $I_{\text{трапеция}} = \mathbf{-34.66}$.
   $R = \left|I_{\text{точн}} - I_{\text{трапеция}}\right| = \left|\frac{-104}{3} - (\mathbf{-34.66}(\dots))\right| = 0$
4. Для метода **Симпсона** при $n = 10$: $I_{\text{точн}} = I_{\text{Симпсона}} = \mathbf{-34.66}$, **значения совпадают**.
   $R = |I_{\text{точн}} - I_{cotes}| = \left|\frac{-104}{3} - (\mathbf{-34.66}(\dots))\right| = 0$

5. Определить относительную погрешность вычислений для каждого метода.

1. Для метода **Ньютона–Котеса**: $\Delta = \frac{\left|\frac{-104}{3} - (-34.64)\right|}{\left|\frac{-104}{3}\right|} \approx \frac{1}{1300}\%$.
2. Для метода **средних прямоугольников**: $\Delta = \frac{\left|\frac{-104}{3} - (\mathbf{-34.62})\right|}{\left|\frac{-104}{3}\right|} \approx \frac{7}{5200}\%$
3. Для метода **трапеций**: $R = 0$ → **погрешности нет**
4. Для метода **Симпсона**: $R = 0$ → **погрешности нет.**

Как видно из результатов, все методы дали относительно малую погрешность, особенно при использовании формулы **трапеций** и Симпсона. Наилучший результат был получен при использовании формулы **трапеций** с $n = 10$ и формулы Симпсона с $n = 10$, при которых значения интеграла полностью совпали.

2. Программная реализация задачи

```
function sqr_l(eqInd) {
    debugger
    console.log("left squares method starting...");
    [bounds, pres] = getSuite()
    let numOfIntervals = 4

    const sqr_l_with_iters = (interN) => {
        let s = "0", step = _(`(${bounds[1]} - ${bounds[0]}) / ${interN}`)
        for (pointer = bounds[0]; _(`${pointer} < ${bounds[1]}`); pointer =
_(`${pointer}+${step}`)) {
            s = _(`${s} + ${step}*${evalExpr(eqInd, pointer)}`)
        }
        return s
    }

    while (numOfIntervals < 100) {
        let i_h = sqr_l_with_iters(numOfIntervals)
        let i_h_half = sqr_l_with_iters(numOfIntervals / 2)
        let err = _(`abs(${i_h} - ${i_h_half}) / (2^2 - 1)`)
        if (_(`${err} < ${pres}`)) return [i_h, numOfIntervals];
        numOfIntervals+=2
    }
    return [sqr_l_with_iters(numOfIntervals), numOfIntervals];
}

function sqr_r(eqInd) {
    console.log("right squares method starting...");
    [bounds, pres] = getSuite()
    let numOfIntervals = 4

    const sqr_r_with_iters = (interN) => {
        let s = "0", step = _(`(${bounds[1]} - ${bounds[0]}) / ${interN}`)
        for (pointer = _(`${bounds[0]}+${step}`); _(`${pointer} <=
${bounds[1]}`); pointer = _(`${pointer}+${step}`)) {
            s = _(`${s} + ${step}*${evalExpr(eqInd, pointer)}`)
        }
        return s
    }

    while (numOfIntervals < 100) {
        let i_h = sqr_r_with_iters(numOfIntervals)
        let i_h_half = sqr_r_with_iters(numOfIntervals / 2)
        let err = _(`abs(${i_h} - ${i_h_half}) / (2^2 - 1)`)
        if (_(`${err} < ${pres}`)) return [i_h, numOfIntervals];
        numOfIntervals+=2
    }
    return [sqr_l_with_iters(numOfIntervals), numOfIntervals];
}
```

```
function sqr_c(eqInd) {
    console.log("center squares method starting...");
    [bounds, pres] = getSuite()
    let numOfIntervals = 4

    const sqr_c_with_iters = (interN) => {
        let s = "0", step = _(`(${bounds[1]} - ${bounds[0]}) / ${interN}`)
        for (pointer = _(`(${bounds[0]}+${step})/2`); _(`${pointer} <=
${bounds[1]}`); pointer = _(`${pointer}+${step}`)) {
            s = _(`${s} + ${step}*${evalExpr(eqInd, pointer)}`)
        }
        return s
    }

    while (numOfIntervals < 100) {
        let i_h = sqr_c_with_iters(numOfIntervals)
        let i_h_half = sqr_c_with_iters(numOfIntervals / 2)
        let err = _(`abs(${i_h} - ${i_h_half}) / (2^2 - 1)`)
        if (_(`${err} < ${pres}`)) return [i_h, numOfIntervals];
        numOfIntervals+=2
    }
    return [sqr_c_with_iters(numOfIntervals), numOfIntervals];
}

function trap(eqInd) {
    console.log("trap method starting...");
    [bounds, pres] = getSuite()
    let numOfIntervals = 4

    const trap_with_iters = (interN) => {
        let s = "0", step = _(`(${bounds[1]} - ${bounds[0]}) / ${interN}`)
        for (let i = 1; i < interN; i++) {
            let x = _(`${bounds[0]} + ${i} * ${step}`)
            s = _(`${s} + ${evalExpr(eqInd, x)}`)
        }
        let end = _(`${bounds[0]} + ${interN} * ${step}`)
        s = _(`(${s}*2 + ${evalExpr(eqInd, bounds[0])} + ${evalExpr(eqInd,
end)})*${step}*0.5`)

        return s
    }

    while (numOfIntervals < 100) {
        let i_h = trap_with_iters(numOfIntervals)
        let i_h_half = trap_with_iters(numOfIntervals / 2)
        let err = _(`abs(${i_h} - ${i_h_half}) / (2^2 - 1)`)
        if (_(`${err} < ${pres}`)) return [i_h, numOfIntervals];
        numOfIntervals+=2
    }
    return [trap_with_iters(numOfIntervals), numOfIntervals];
```

```
}

function simp(eqInd) {
    console.log("simp method starting...");
    [bounds, pres] = getSuite()
    let numOfIntervals = 4

    const simp_with_iters = (interN) => {
        let step = _(`(${bounds[1]} - ${bounds[0]}) / ${interN}`)
        let fs = "0", ss = "0"
        for (let i = 1; i < interN; i++) {
            let y = evalExpr(eqInd, _(`${bounds[0]} + ${i}*${step}`))
            if(i % 2 == 0) {
                ss = _(`${ss} + ${y}`)
            } else {
                fs = _(`${fs} + ${y}`)
            }
        }
        let yn = evalExpr(eqInd, _(`${bounds[0]} + ${step} * ${interN}`))
        let s = _(`${step}/3 * (${evalExpr(eqInd, bounds[0])} + 4*(${fs}) +
2*(${ss}) + ${yn})`)
        return s
    }

    while (numOfIntervals < 100) {
        debugger
        let i_h = simp_with_iters(numOfIntervals)
        let i_h_half = simp_with_iters(numOfIntervals / 2)
        let err = _(`abs(${i_h} - ${i_h_half}) / (2^4 - 1)`)
        if (_(`${err} < ${pres}`)) return [i_h, numOfIntervals];
        numOfIntervals += 2
    }
    return [simp_with_iters(numOfIntervals), numOfIntervals];
}
```

Вывод

В ходе выполнения лабораторной работы были изучены численные методы интегрирования с использованием Python. В результате работы были рассмотрены различные численные методы вычисления определенных интегралов: метод прямоугольников (левых, правых, средних), метод трапеций, метод Ньютона-Котеса и метод Симпсона.

Была реализована программа, позволяющая выбрать одну из предложенных функций, задать пределы интегрирования, точность и начальное значение числа разбиения интервала интегрирования. Написав реализации всех трех методов решения интегралов, можно сделать вывод, что самым точным и быстрым является метод Симпсона.

В ходе вычислительной реализации задачи были рассчитаны интегралы различными методами и проведено сравнение результатов с точными значениями интегралов.

Также была выполнена дополнительная задача по установлению сходимости рассматриваемых несобственных интегралов 2 рода и их вычислению заданными численными методами в случаях, когда подынтегральная функция терпит бесконечный разрыв в точке a, в точке b или на отрезке интегрирования.