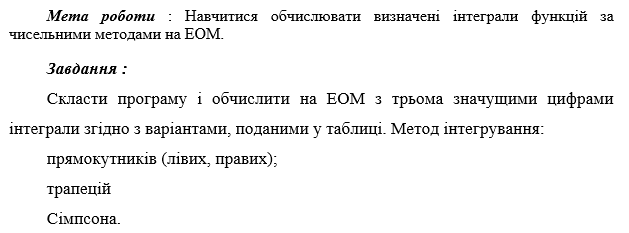
**Лабораторна робота №3**

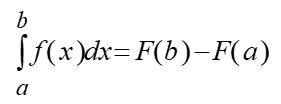
**Чисельне інтегрування фунцій**





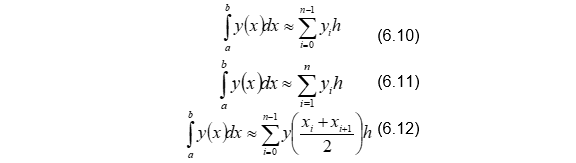
Короткі теоретичні відомості

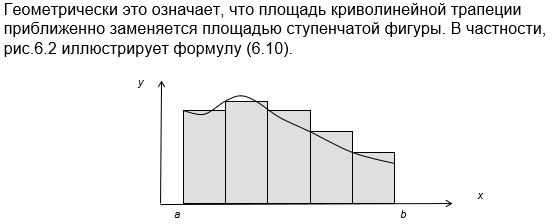
Точное решение определенного интеграла:



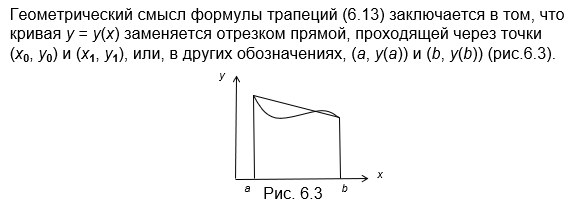
Метод левых, правых и средних прямоугольников:

Определенный интеграл приближенно заменяется интегральной суммой:



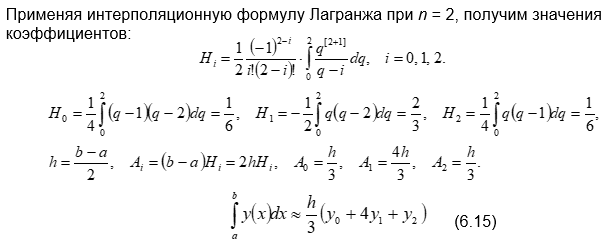


Метод трапеций:

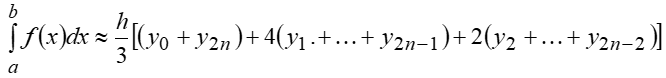


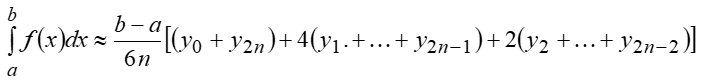


Метод Симпсона









**Хід роботи**

1. Текст програми

/\*\*

\* Интегрирует функцию f на интервале [a, b] методом Симпсона

\* @param {(x: number) => number} f интегрируемая функция

\* @param {number} a нижная граница интегрирования

\* @param {number} b верхняя граница интегрирования

\* @param {number} n количество шагов

\*/

function integrateBySimpson(f, a, b, n) {

if (a > b) [a, b] = [b, a];

const h = (b - a) / (2 \* n);

let s = f(a) + f(b);

let k = 1;

for (let i = 1; i <= 2 \* n - 1; i++) {

s += f(a + h \* i) \* (3 + k);

k = -k;

}

return Math.abs((h / 3) \* s);

}

/\*\*

\* Интегрирует функцию f на интервале [a, b]

\* @param {(x: number) => number} f интегрируемая функция

\* @param {number} a нижная граница интегрирования

\* @param {number} b верхняя граница интегрирования

\* @param {number} n количество шагов

\* @param {'left-rect'|'middle-rect'|'right-rect'|'trapezoidal'} method метод интегрирования

\*/

function integrateByPolygon(f, a, b, n, method) {

if (a > b) [a, b] = [b, a];

const h = (b - a) / n;

if (method === 'left-rect') {

return getSum(f, a, n, h, f(a), 0);

} else if (method === 'middle-rect') {

return getSum(f, a, n, h, f(a + h / 2), h / 2);

} else if (method === 'right-rect') {

return getSum(f, a, n, h, f(b), 0);

} else if (method === 'trapezoidal') {

return getSum(f, a, n, h, (f(a) + f(b)) / 2, 0);

} else {

throw SyntaxError(`Wrong integration method (${method})`);

}

}

/\*\*

\* Вычисляет общую часть интеграла функции f

\* @param {(x: number) => number} f интегрируемая функция

\* @param {number} a нижная граница интегрирования

\* @param {number} h шаг интегрирования

\* @param {number} s начальное значение результата

\* @param {number} offset смещение шага интегрирования

\*/

function getSum(f, a, n, h, s, offset) {

for (let i = 1; i < n; i++) {

s += f(a + i \* h + offset);

}

return Math.abs(s \* h);

}

/\*\*

\* Интегрирует функцию до точности precision

\* @param {(x: number) => number} f интегрируемая функция

\* @param {number} a нижняя граница интегрирования

\* @param {number} b верхняя граница интегрирования

\* @param {'left-rect'|'middle-rect'|'right-rect'|'trapezoidal'|'simpson'} method метод интегрирования

\* @param {number} precision точность вычисления

\* @returns {{n: number, val: number}[]} массив значение всех итераций

\* интегрирования(значения указаны в порядке возрастания

\* точности - наибольшая точно в последнем элементе)

\*/

function integrate(f, a, b, method, precision = 0.00001) {

const epsilon = precision \* ((method === 'simpson') ? 15 : 3);

const res = [];

let n = 2;

let prev = getNextValue(f, a, b, n, method);

res.push({ n, val: prev });

while (true) {

n \*= 2;

const next = getNextValue(f, a, b, n, method);

res.push({ n, val: next });

if (Math.abs(next - prev) <= epsilon) {

break

}

prev = next;

}

return res;

}

/\*\*

\* Интегрирует функцию f на интервале [a, b]

\* по методу method, и количеству разбиения n

\* @param {(x: number) => number} f интегрируемая функция

\* @param {number} a нижняя граница интегрирования

\* @param {number} b верхняя граница интегрирования

\* @param {'left-rect'|'middle-rect'|'right-rect'|'trapezoidal'|'simpson'} method метод интегрирования

\* @param {number} precision точность вычисления

\* @returns {{n: number, val: number}[]} массив значение всех итераций

\* интегрирования(значения указаны в порядке возрастания

\* точности - наибольшая точно в последнем элементе)

\*/

function getNextValue(f, a, b, n, method) {

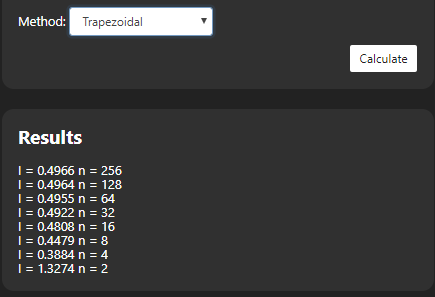
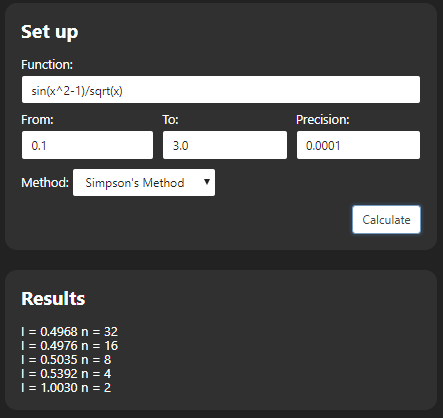
return (method === 'simpson') ?

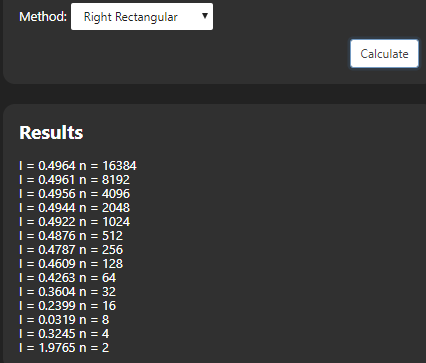
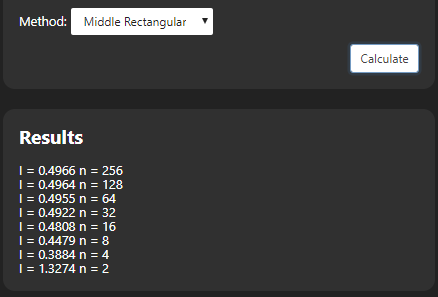
integrateBySimpson(f, a, b, n) :

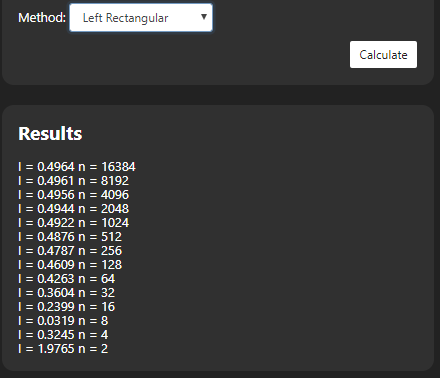
integrateByPolygon(f, a, b, n, method);

}

1. Результат виконання







**Висновок:** на цій лабораторній роботі навчився розв’язувати визначенні інтеграли за допомогою числених методів на ЕОМ. Проаналізувавши результати можна стверджувати, що найбільш еффективний метод – метод Сімпсона. Після нього метод трапецій, та метод середніх трекутників, і найповільніші методи – метод лівих і правих прямокутників.