

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ имени Н.Э.БАУМАНА  
(национальный исследовательский университет)»**

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа № 4**

«Приближенное вычисление определённого интеграла»

по дисциплине «Численные методы»

Вариант 10

Работу выполнил

студент группы ИУ9-62Б

Жук Дмитрий

**Цель работы**

Целью данной работы является изучение способов приближенного вычисления определенного интеграла, а также сравнение их между собой.

**Задание**

1. Построить все заявленные способы, а именно: метод прямоугольников, метод трапеций, формулу Симпсона и метод Монте-Карло.
2. Посчитать значения заданного интеграла и сравнить их.

**Индивидуальный вариант**

**Реализация**

1. Все заявленные способы вычисления определенного интеграла основаны на его геометрическом смысле, а именно на том факте, что значение интеграла для функции есть площадь фигуры под графиком этой самой функции. Достаточно взглянуть на графики и увидеть формулы чтобы понять не только принцип их работы, но и их правильность (рисунки 1, 2, 3, 4):

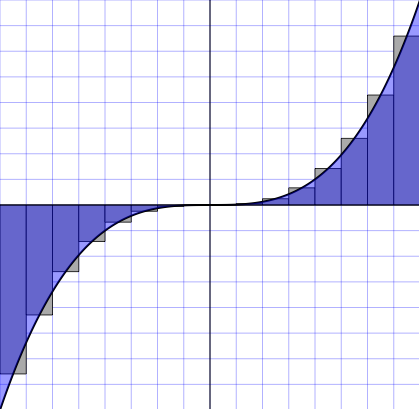


Рисунок 1 – метод прямоугольников,

Изображение выглядит как стрела

Автоматически созданное описание

Рисунок 2 – метод трапеций,

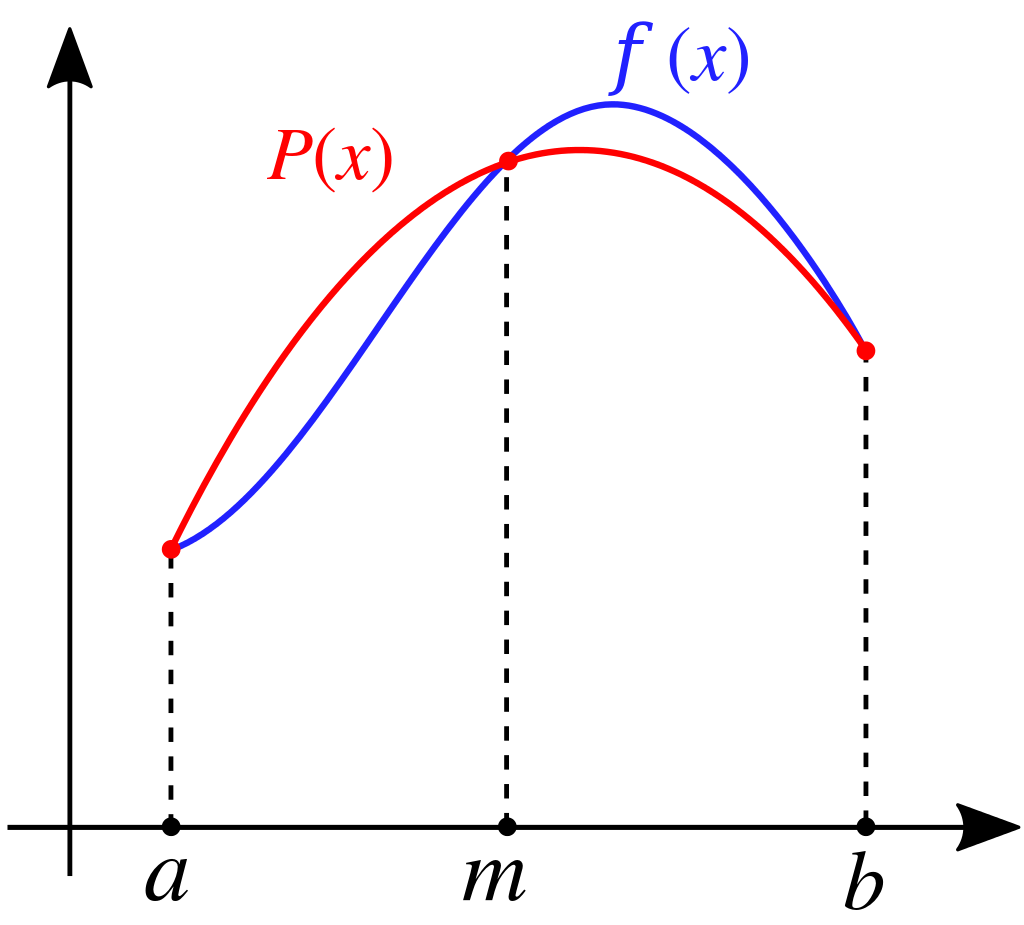


Рисунок 3 – формула Симпсона,

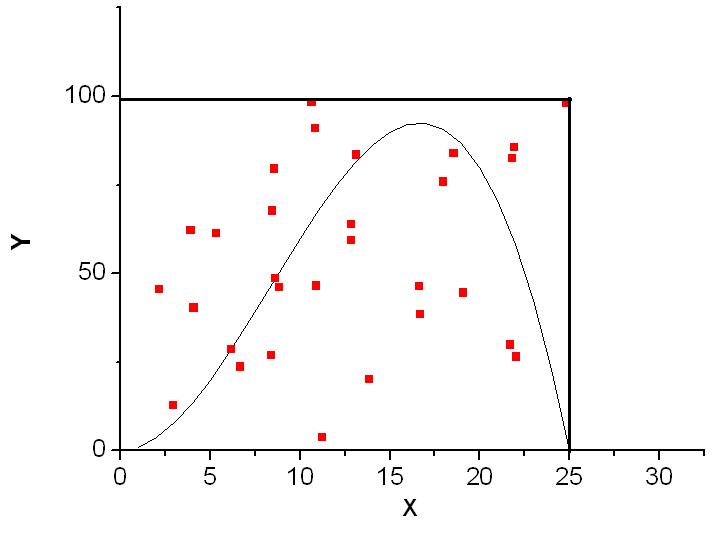


Рисунок 4 – метод Монте-Карло

1. Используя TypeScript (листинг 1), найдем разности в методов (рисунок 5).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – разности методов: название метода, количество разбиений, полученное значение, абсолютное значение, разность

import style from 'chalk';

export const N = 16;

export const F = Math.exp;

export const A = 0;

export const B = 1;

type Input = {

a?: number;

b?: number;

n?: number;

f?: (\_: number) => number;

startSum?: number;

};

const buildMetod = ({ n, f, startSum }: Required<Input>) => Array

.from({ length: n }, (\_, i) => f(i))

.reduce((sum, x) => sum + x, startSum) / n;

const metodRectangles = (f = F) => ({

a = A, b = B, n = N,

}: Input = {}) => buildMetod({

a, b, n, f: (i) => f(a + (b - a) \* (2 \* i + 1) / (2 \* n)), startSum: 0,

}) \* (b - a);

const metodTrapezoids = (f = F) => ({

a = A, b = B, n = N,

}: Input = {}) => buildMetod({

a, b, n, f: (i) => f(a + (b - a) \* i / n) + f(a + (b - a) \* (i + 1) / n), startSum: 0,

}) \* (b - a) / 2;

const metodSimpson1 = (f = F) => ({

a = A, b = B, n = N,

}: Input = {}) => buildMetod({

a,

b,

n,

f: (i) => f(a + (b - a) \* i / n)

+ 4 \* f(a + (b - a) \* (2 \* i + 1) / (2 \* n))

+ f(a + (b - a) \* (i + 1) / n),

startSum: 0,

}) \* (b - a) / 6;

const metodSimpson2 = (f = F) => ({

a = A, b = B, n = N,

}: Input = {}) => buildMetod({

a,

b,

n,

f: (i) => +(i !== 0) \* (i % 2 === 0 ? 2 : 4) \* f(a + (b - a) \* i / n),

startSum: f(a) + f(b),

}) \* (b - a) / 3;

(<[((\_: number) => number), number][]>[

[F, Math.E - 1],

// [function f10(x: number) { return x - Math.trunc(x); }, 1],

]).forEach(([func, orig]) => {

console.log(func.name, 'n:', N);

console.log((<[((i?: Input) => number), string][]>[

[metodRectangles(func), 'metodRectangles'],

[metodTrapezoids(func), 'metodTrapezoids'],

[metodSimpson1(func), 'metodSimpson1'],

[metodSimpson2(func), 'metodSimpson2'],

])

.map(([e, name]) => `${name.padStart(20)}: ${style.yellow(e().toExponential(10))} -> ${style.yellow(Math.abs(orig - e()).toExponential(10))}`).join('\n'));

});

(<[((\_: number) => number), number, number, number, number][]>[

[F, Math.E - 1, 1e-2, 0, 1],

[

function v10(x) { return x \* Math.sin(x) \*\* 2; },

Math.PI \*\* 2 / 4,

1e-2,

0,

Math.PI,

],

// [function f10(x: number) { return x - Math.trunc(x); }, 1],

]).forEach(([func, orig, eps, a, b]) => {

console.log(func.name, eps, orig);

console.log((<[((i?: Input) => number), string, number][]>[

[metodRectangles(func), 'metodRectangles', 2],

[metodTrapezoids(func), 'metodTrapezoids', 2],

[metodSimpson1(func), 'metodSimpson1', 4],

[metodSimpson2(func), 'metodSimpson2', 4],

])

.map(([e, name, step]) => {

for (let n = 2; ; n \*= 2) {

const value = e({ n, a, b });

const epsGot = value - orig;

if (Math.abs(epsGot) < eps) {

const newValue = e({ n: n \* 2, a, b });

const r = (value - newValue) / (2 \*\* step - 1);

const ideal = value - r;

return `${name.padStart(20)}: ${style.yellow(n)} ${style.yellow(value.toExponential(10))} ${style.yellow(newValue.toExponential(10))} ${style.yellow(ideal.toExponential(10))} ${style.yellow(Math.abs(orig - ideal).toExponential(10))}`;

}

}

}).join('\n'));

});

(<[((\_: number) => number), number, number, number, number, number][]>[

[F, Math.E - 1, 1e-2, 0, 1, Math.E],

[

function v10(x) { return x \* Math.sin(x) \*\* 2; },

Math.PI \*\* 2 / 4,

1e-2,

0,

Math.PI,

Math.PI,

],

// [function f10(x: number) { return x - Math.trunc(x); }, 1],

]).forEach(([func, orig, eps, a, b, max]) => {

let correct = 1;

let all = 1;

let n = 2;

for (; Math.abs(correct / all - orig / (max \* (b - a))) > eps; n \*= 2) {

for (let i = 0; i < n; i++) {

if (func(a + (b - a) \* Math.random()) > Math.random() \* max) {

correct++;

}

}

all += n;

}

console.log(func.name, 'Monte Carlo:'.padStart(17), n, all, correct / all \* (max \* (b - a)));

});

Листинг 1 — Метод создания сплайн-интерполяции

**Вывод**

В ходе лабораторной работы были изучены способы вычисления определённого интеграла. Если для функции из индивидуального варианта погрешность отсутствует, то для экспоненты методы дают разную точность: так у методов прямоугольников и трапеций погрешность почти идентичная, а у формулы Симпсона сходимость намного выше и, как следствие, меньше погрешность. Метод же Монте-Карло – очень нестабильный из-за своей природы: случайно брать точки на плоскости нам остается уповать на вероятность. Этот метод можно качественно улучшить если брать точки по некой сетке (матрице) и при недостаточной точности – увеличивать плотность точек.