Санкт-Петербургский *Национальный Исследовательский Университет ИТМО*
Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №3

Вариант: метод Симпсона

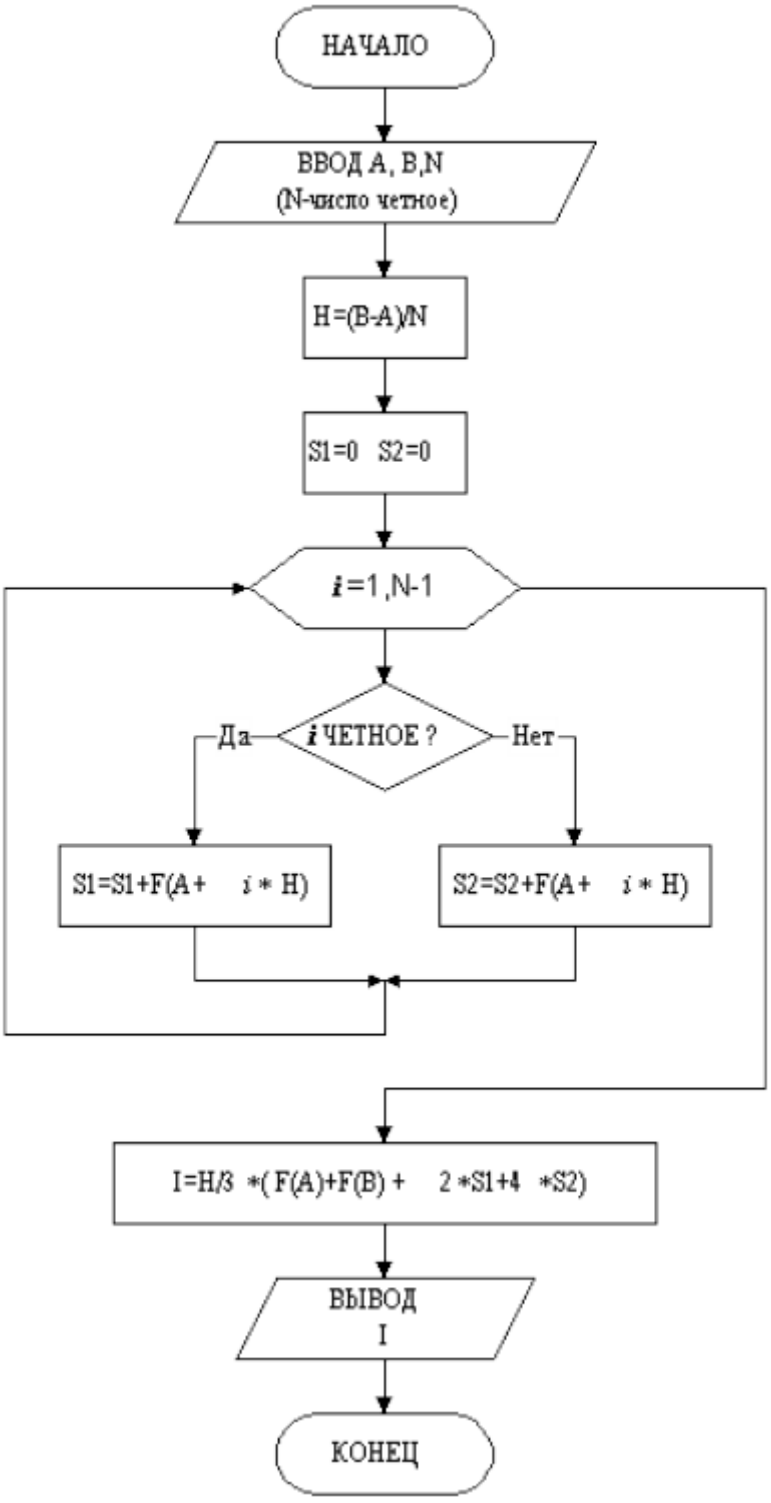
Выполнил:
студент группы P3231
Нестеров Иван Алексеевич

Преподаватель:
Перл Ольга Вячеславовна

г. Санкт-Петербург
2022 г.

Цель работы: реализовать метод Симпсона для численного интегрирования заданных функций.

Блок-схема указанного метода:



Описание метода:

В основе метода лежит квадратичная интерполяция подынтегральной функции на отрезке [a; b] по трем равноотстоящим узлам. Для поиска решения разобьем интервал интегрирования [a; b] на четное число n равных отрезков с шагом h. Соответственно, примем, что:

x_0 = a

$x_1 = x_0 + h$

…

$x_n = x_0 + nh = b$

Значения функций в точках обозначим следующим образом:

y_0 = f(a),

y_1 = f(x_1),

y_1 = f(x_2),

…

y_n = f(x_n).

Далее воспользуемся следующей формулой из лекции для вычисления значения интеграла.

$$\int_a^b f(x)\,dx = \frac{h}{3}\left(f(a) + 4f\left(\frac{b-a}{2}\right) + f(b)\right) - R^+ + R^-$$

$$R = R^+ - R^-$$

Листинг вычислительного метода:

```
public int integrate() {
    double firstBorder;
    double secondBorder;
    double height;
    if (topLimit != bottomLimit) {
        for (int n = 4; n <= 10000; n += 2) {
            double sum1 = 0;
            double sum2 = 0;
            height = (topLimit - bottomLimit) / n;
            for (int index = 1; index < n; index++) {
                sum1 += 4 * function.calculateFunction(bottomLimit + index *
height);
                index++;
                sum1 += 2 * function.calculateFunction(bottomLimit + index *
height);
```

```
            }
            firstBorder = (sum1 + function.calculateFunction(bottomLimit) -
function.calculateFunction(topLimit))
                    * height / 3;

            height = (topLimit - bottomLimit) / (2 * n);
            for (int index = 1; index < 2 * n; index++) {
                sum2 += 4 * function.calculateFunction(bottomLimit + index *
height);
                index++;
                sum2 += 2 * function.calculateFunction(bottomLimit + index *
height);
            }
            secondBorder = (sum2 + function.calculateFunction(bottomLimit) -
function.calculateFunction(topLimit))
                    * height / 3;
            if (Math.abs(secondBorder - firstBorder) / 15 < accuracy) {
                result = secondBorder;
                steps = n;
                error = Math.abs(secondBorder - firstBorder) / 15;
                return 0;
            }
            if (n == 10000) {
                steps = 0;
                return -1;
            }
        }
    }
    else {
        result = 0;
        steps = 0;
        return 1;
    }
    return -1;
}
```

Демонстрация работы программы:

```
Simpson method solver system has been started!
List of functions:
1) y = x^2
2) y = sin(x)
3) y = 1/x
Choose a number of function for integrating: 3
Enter bottom limit: 10
Enter top limit: 20
Enter necessary accuracy: 0.1
Result: 0.693
Error: 0.0000066292
Steps amount: 4

Process finished with exit code 0
```

```
Simpson method solver system has been started!
List of functions:
1) y = x^2
2) y = sin(x)
3) y = 1/x
Choose a number of function for integrating: 2
Enter bottom limit: 20
Enter top limit: 10
Enter necessary accuracy: notNumber
Accuracy is a double number less than 0.
0.0001
Result: -1.247
Error: 0.0000694425
Steps amount: 16

Process finished with exit code 0
```

Вывод:

Изучая численное интегрирование, я понял механизмы вычисления интегралов в языках программирования, понял то, как непрерывную функцию, можно задать очень схожей с ней «дискретной» функцией. Для вычисления

используется разбиение площади под графиком на небольшие блоки, площадь которых можно посчитать. Сложение же всех «площадей» приблизительно даст значение интеграла. Изучая теорию, убедился, что методы интегрирования концептуально работают примерно одинаково, и различия лишь в реализации разбиения заданной площади на блоки и вычисления «площадей» этих блоков.