Вычислительная математика

Раевский Григорий, группа 3.2

06.04.2024

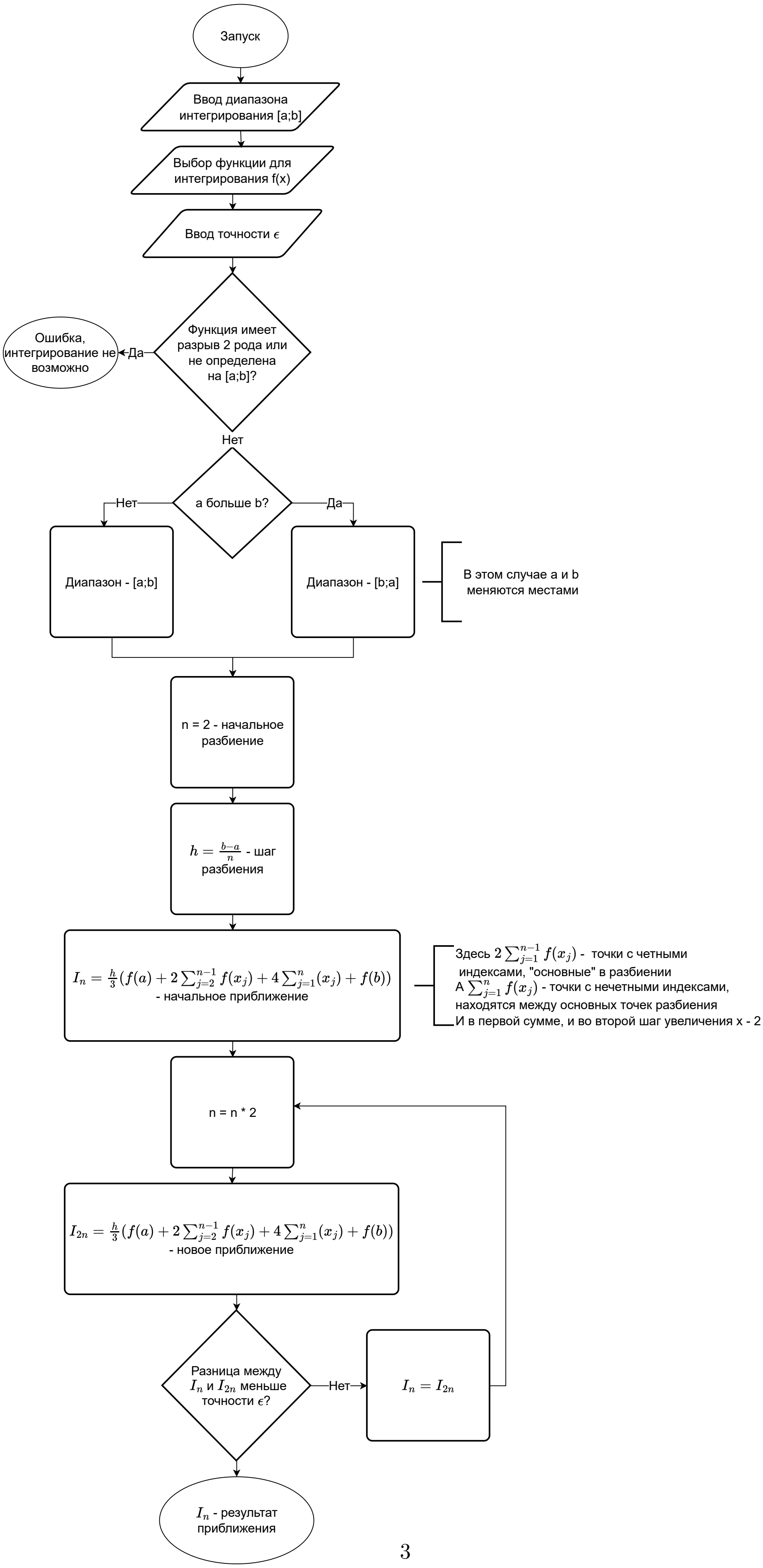
Отчет по лабораторной работе 4

Содержание

Описание численного метода

Интегрирование с помощью метода Симпсона — численный метод поиска значения интеграла $\int f(x)dx$ на промежутке [a;b] с заданной точностью. Метод основан на итеративном приближении функции f(x) параболами. Итеративность достигается так, что количество разбиений на элементарные отрезки увеличивается каждый раз после расчета приближений только в том случае, если разница между двумя последними приближениями оказалась больше заданной точности ϵ. Для каждой группы из 3 точек интервала (начало, середина, конец) высчитывается парабола, проходящая через эти точки, после чего значения интегралов в этих параболах вычисляются и суммируются, что и дает приближение к исходному интервалу. Если функция имеет разрыв 2 рода, или не определена, то метод Симпсона, как и другие методы, не способен рассчитать значение интеграла на заданном промежутке [a;b].

Диаграмма


Запуск
Ввод диапазона интегрирования [a;b]
Выбор функции для интегрирования f(x)
Ввод точности ϵ
Функция имеет разрыв 2 рода или не определена на [a;b]?
Да
Ошибка, интегрирование не возможно
Нет
a больше b?
Нет
Да
Диапазон - [a;b]
Диапазон - [b;a]
В этом случае a и b меняются местами
n = 2 - начальное разбиение
$h = \frac{b-a}{n}$ - шаг разбиения
$I_n = \frac{h}{3}(f(a) + 2\sum_{j=2}^{n-1} f(x_j) + 4\sum_{j=1}^{n}(x_j) + f(b))$ - начальное приближение
Здесь $2\sum_{j=1}^{n-1} f(x_j)$ - точки с четными индексами, "основные" в разбиении
А $\sum_{j=1}^{n} f(x_j)$ - точки с нечетными индексами, находятся между основных точек разбиения
И в первой сумме, и во второй шаг увеличения x - 2
n = n * 2
$I_{2n} = \frac{h}{3}(f(a) + 2\sum_{j=2}^{n-1} f(x_j) + 4\sum_{j=1}^{n}(x_j) + f(b))$ - новое приближение
Разница между I_n и I_{2n} меньше точности ϵ?
Нет
$I_n = I_{2n}$
I_n - результат приближения

Листинг кода

```
def check_discontinuity(a, b, func):
    dx = (b - a) / 1000
    try:
        test_points = [a + dx * i for i in range(1001)]
        for point in test_points:
            func(point)
    except:
        Result.error_message = "Integrated function has discontinuity or does not defined in current interval"
        Result.has_discontinuity = True
    return Result.has_discontinuity

def calculate_simpson(a, b, n, func):
    h = (b - a) / n

    first_sum = 0
    for i in range(2, n - 1, 2):
        first_sum += func(a + h * i)

    second_sum = 0
    for i in range(1, n, 2):
        second_sum += func(a + h * i)

    integral = (func(a) + 2 * first_sum + 4 * second_sum + func(b)) * (h / 3)
    return integral

def calculate_integral(a, b, f, epsilon):
    if (epsilon <= 0):
        raise Exception("Epsilon should be greater than 0!")

    func = Result.get_function(f)
    if Result.check_discontinuity(a, b, func):
        return None

    n = 2
```

```
        result_previous = Result.calculate_simpson(a, b, n, func)
        while True:
            n *= 2
            result_next = Result.calculate_simpson(a, b, n, func)
            if abs(result_previous - result_next) < epsilon:
                break
            result_previous = result_next

        if a > b:
            return -result_previous
        return result_previous
```

Примеры работы программы

Стандартный случай

stdin:

```
1
2
3
0.001
```

stdout:

```
4.333333333333333
```

Для функции $f(x) = x^2 + 2$ приближенное значение интеграла было найдено успешно.

Малый интервал

stdin:

```
0.99
1.01
1
0.0001
```

stdout:

```
0.020000666733340017
```

Для функции $f(x) = \frac{1}{x}$ код отработал корректно, найдя приближение с высокой точностью.

Разрыв

stdin:

```
-1
1
1
0.001
```

stdout:

```
Integrated function has discontinuity or does not defined in current interval
```

Функция $f(x) = \frac{1}{x}$ имеет неустранимый разрыв в 0, следовательно, значение интеграла не определено.

Устранимый разрыв

stdin:

```
-1
1
2
0.01
```

stdout:

```
Integrated function has discontinuity or does not defined in current interval
```

Программа выдала ошибку, так как посчитала, что разрыв неустраним. Однако это произошло из-за того, что на code-n-test значение Result.eps не было определено при создании кода. Я этого не заметил, так как копировал напрямую с сайта из блока Code edit, предполагая, что нужно изменять только calculate_integral, а остальной код корректен. Если значение epsilon определить, например, $eps = 10^{-7}$**, то код отработает корректно и выдаст значение 1.8943139898719288 для этих тестовых данных. Я так же предполагаю, что из-за этого и не прошел тест 1. Если возможно, хотел повторного тестирования моего кода с исправлением баллов...**

Обратный интервал

stdin:

```
3
1
4
0.001
```

stdout:

```
12.0
```

Код отработал не совсем корректно, он выдал значение 12, хотя должно было быть -12. Все дело в последнем блоке кода, в комментариях я приложил исправленный код.

Выводы

Программа работает стабильно на данных, если функция не имеет разрыва 2 рода (если Result.eps определен, то разрывы 1 рода будут корректно отрабатываться) и корректно находит приближенное значение интеграла с заданной точностью. Однако программа выдаст ошибку, если функция будет иметь неустранимый разрыв или не будет определена в заданном диапазоне.

Метод Симпсона обладает высокой точностью и скоростью получения значения интеграла. Это возможно благодаря аппроксимации функции $f(x)$ с помощью парабол. Он обеспечивает лучшую точность по сравнению с методом трапеций, так как последний использует линейную аппроксимацию. Так же, метод Симпсона достаточно просто реализовать. Метод прямоугольников реализовать еще проще, но у него страдает точностью, так как используются константы вместо парабол.

Точность метода Симпсона зависит от вычислений значений функции в 3 точках для каждого интервала, что может повлечь к накоплению численной ошибки. Однако он хорошо подходит для широкого спектра задач, так как остается довольно точным и обладает алгоритмической сложностью $O(n \log n)$, где n - количество разбиений интервала.

P.S.

Прошу обратить особое внимание на 4 пример работы