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)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

Лабораторная работа № 2

«Сравнительный анализ методов численного интегрирования»

по курсу «Численные методы»

Выполнил:

студент группы ИУ9-61Б

Локшин Вячеслав

Проверила:

Домрачева А. Б.

Москва, 2024

1. **Цель**

Целью данной работы является сравнение по быстродействию методов численного интегрирования:

1. Метод центральных прямоугольников
2. Метод трапеций
3. Метод Симпсона

2. **Постановка задачи**

Дано: Интеграл I

$$I = \int_a^b f(x)dx$$

где $f(x)$ – подынтегральная функция, непрерывная на отрезке $[a, b]$.

Найти: Значение интеграла

$$I^* \approx I$$

При заданной точности ε менее 0.01

Индивидуальный вариант: $f(x) = \frac{ln(x)^2}{x}, a = \frac{1}{e},\ b = e$

$$\int_{\frac{1}{e}}^{e} \frac{ln(x)^2}{x} dx = \frac{2}{3} \approx 0.66667$$

3. **Основные теоретические сведения**

3.1 Метод центральных прямоугольников

Метод центральных прямоугольников заключается в вычислении площади под графиком интегрируемой функции путем суммирования площадей прямоугольников с высотой, равной значению функции в центре каждого отрезка разбиения и шириной, равной шагу разбиения.

Для нахождения значения интеграла функции $f(x)$ на отрезке $[a, b]$ нужно разбить отрезок на n равных отрезков длиной $h = \frac{b-a}{n}$. Получаем разбиение данного отрезка точками

$$x_{i-0.5} = a + (i - 0.5) \cdot h \quad i = \overline{1, n}$$

Тогда приближенное значение интеграла на всем отрезке будет равно:

$$I^* = h \cdot \sum_{i=1}^{n} f(x_{i-0.5}) = h \cdot \sum_{i=1}^{n} f(a + (i - 0.5) \cdot h)$$

3.2 Метод трапеций

Суть метода состоит в вычислении площади под графиком интегрируемой функции путем суммирования площадей трапеций. Высота каждой трапеции определяется значением функции в узле интегрирования, а ширина - шагом разбиения.

Для нахождения значения интеграла функции $f(x)$ на отрезке $[a, b]$ нужно разбить отрезок на n равных отрезков длиной $h = \frac{b-a}{n}$. Получаем разбиение данного отрезка точками

$$x_i = a + i \cdot h \quad i = \overline{1, n}$$

Тогда приближенное значение интеграла на всем отрезке будет равно:

$$I^* = h \cdot \left(\frac{f(a) + f(x_1)}{2} + \frac{f(x_1) + f(x_2)}{2} + \cdots + \frac{f(x_{n-1}) + f(b)}{2} \right) =$$

$$= h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) \right)$$

3.3 Метод Симпсона

Метод заключается в приближении функции на отрезке $[a, b]$ интерполяционным многочленом 2 степени функции $P_2(x)$

$$P_2(x) = f_{i-0.5} + \frac{f_i - f_{i-1}}{h}(x_i - x_{i-0.5}) + \frac{f_i - 2f_{i-0.5} + f_{i-1}}{\frac{h^2}{2}}(x_i - x_{i-0.5})^2$$

Тогда приближенное значение интеграла на всем отрезке будет равно:

$$I^* = \frac{h}{6} \cdot \left(f(a) + f(b) + 4 \cdot \sum_{i=1}^{n} f(x_{i-0.5}) + \sum_{i=1}^{n-1} f(x_i) \right)$$

3.4 Уточнение значения интеграла по Ричардсону

$I \approx I_h^* + O(h^k)$, где k – порядок точности метода, I_h^* – приближенное значение интеграла, вычисленного с помощью метода с шагом h. $O(h^k) \approx c \cdot h^k$, где c – некоторая константа, h – шаг.

Считаем, что вычисления проводятся без вычислительной погрешности, тогда можно записать строгое равенство $I = I_h^* + c \cdot h^k$ для шага h. $I = I_{\frac{h}{2}}^* + c \cdot \left(\frac{h}{2}\right)^k$ для шага $\frac{h}{2}$.

Из равенств получаем уточненное значение интеграла:

$$I = I_{\frac{h}{2}}^* + \frac{I_{\frac{h}{2}}^* - I_h^*}{2^k - 1}$$

Где значение R – уточнение по Ричардсону:

$$R = \frac{I_{\frac{h}{2}}^* - I_h^*}{2^k - 1}$$

Данная величина используется для компенсации методологической погрешности численных методов интегрирования.

Чтобы построить процедуру приближенного вычисления интеграла с заданной точностью ε, используется правило Рунге:

$$|R| < \varepsilon$$

4. Реализация

Листинг 1. Численное интегрирование

```
package main

import (
    "fmt"
    "math"
)

const (
    epsilon                   = 0.0001
    lowBorder                 = 1.0 / math.E
    upBorder                  = math.E
    accuracyStepForRichardson = 4
)

// INFO: для смены варианта нужно изменить эту функцию

func f(x float64) float64 {
    return math.Pow(math.Log(x), 2) / x
}

func rectangle(a float64, b float64, n int) float64 {
    h := (b - a) / float64(n)
    var s float64
    for i := 1; i <= n; i++ {
       s += f(a + (float64(i)-0.5)*h)
    }
    return h * s
}

func trapezoid(a float64, b float64, n int) float64 {
    h := (b - a) / float64(n)
    var s float64
    for i := 1; i < n; i++ {
       s += f(a + float64(i)*h)
    }
    return h * ((f(a)+f(b))/2 + s)
}

func simpson(a float64, b float64, n int) float64 {
    h := (b - a) / float64(n)

    var s1, s2, s3 float64
    for i := 1; i <= n; i++ {
       s1 += f(a + float64(i)*h)
    }
    for i := 1; i <= n; i++ {
       s2 += f(a + (float64(i)-0.5)*h)
    }
```

```
    for i := 1; i <= n; i++ {
        s3 += f(a + (float64(i)-1)*h)
    }

    s := s1 + 4*s2 + s3

    return h / 6 * s
}

func getIntegralValue(
    calculate func(float64, float64, int) float64,
    a float64, b float64) ([]int, []float64, []float64) {
    n := 1
    richardson := epsilon * 1000
    result := 0.0
    i := 0

    returnN := make([]int, 0)
    returnResult := make([]float64, 0)
    returnRichardson := make([]float64, 0)

    for math.Abs(richardson) >= epsilon {
        n *= 2
        prevResult := result
        result = calculate(a, b, n)
        richardson = (result - prevResult) / (math.Pow(2,
accuracyStepForRichardson) - 1)
        i++
        returnResult = append(returnResult, result)
        returnRichardson = append(returnRichardson, richardson)
        returnN = append(returnN, n)
    }
    fmt.Printf("n is %d | ", n)
    fmt.Printf("result: %.16f\n", result+richardson)

    return returnN, returnResult, returnRichardson
}

func printMethod(n []int, res, rich []float64, method string) {
    fmt.Println(method)
    fmt.Println("Все n для", method)
    fmt.Println(n)
    fmt.Println("Все res для", method)
    fmt.Println(res)
    fmt.Println("Все rich для", method)
    fmt.Println(rich)
}

func printAll(
    nRec, nTra, nSim []int,
    resRec, resTra, resSim []float64,
    richRec, richTra, richSim []float64) {

    printMethod(nRec, resRec, richRec, "rectangle")
    printMethod(nTra, resTra, richTra, "trapezoid")
    printMethod(nSim, resSim, richSim, "simpson")
}

func main() {

    fmt.Println("epsilon:", epsilon)

    fmt.Println("_____________________________")
```

```
    fmt.Println("Central rectangles method:")
    nRec, resRec, richRec := getIntegralValue(rectangle, lowBorder,
upBorder)
    fmt.Println("_____________________________")
    fmt.Println("Trapezoids method:")
    nTra, resTra, richTra := getIntegralValue(trapezoid, lowBorder,
upBorder)
    fmt.Println("_____________________________")
    fmt.Println("Simpsons method:")
    nSim, resSim, richSim := getIntegralValue(simpson, lowBorder, upBorder)
    fmt.Println("_____________________________")

    printAll(nRec, nTra, nSim, resRec, resSim, resTra, richRec, richTra,
richSim)

}
```

5. Результаты

Для тестирования выбран интеграл из условия и было проверено несколько ε:

$$\varepsilon = 0.001, \qquad \varepsilon = 0.0001, \qquad \varepsilon = 0.00001$$

Рисунок 1 – Пример вывода программы

```
/Users/slavaruswarrior/Library/Caches/JetBrains/GoLand2023.3/tmp/GoLand/___go_build_lab3
epsilon: 0.0001
----------------------------
Central rectangles method:
n is 128 | result: 0.6664160015841885
----------------------------
Trapezoids method:
n is 256 | result: 0.6667920000514594
----------------------------
Simpsons method:
n is 32 | result: 0.6666685330021774
----------------------------
```

Приведем их в таблицу

Таблица 2 – Результаты программы

Метод	Число отрезков равной длины	Значение интеграла с уточнением по Ричардсону
$\varepsilon = 0.001$		

Метод центральных прямоугольников	32	0.6626618168283366
Метод трапеций	64	0.6686718910890977
Метод Симпсона	16	0.6667423970883434
$\varepsilon = 0.0001$		
Метод центральных прямоугольников	128	0.6664160015841885
Метод трапеций	256	0.6667920000514594
Метод Симпсона	32	0.6666685330021774
$\varepsilon = 0.00001$		
Метод центральных прямоугольников	512	0.6666509999904405
Метод трапеций	512	0.6666980000195374
Метод Симпсона	64	0.6666667009952734

6. Вывод

В ходе выполнения лабораторной работы были рассмотрены 3 метода численного интегрирования: метод центральных прямоугольников, метод трапеций и метод Симпсона. Данные методы были реализованы на языке программирования Golang.

Самым точным среди рассмотренных трех методов оказался метод Симпсона, далее идет метод центральных прямоугольников, а в конце метод трапеций.