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

«Численное интегрирование»

по дисциплине «Численные методы»

Вариант 9

Работу выполнил

студент группы ИУ9-62Б

Егоров Алексей

Москва, 2022

1. Цель работы

Сравнительный анализ методов численного интегрирования: метод средних прямоугольников, метод трапеций, метод Симпсона, метод Монте-Карло. Вычисление уточненного значения интеграла.

2. Постановка задачи

Дано: интеграл $I = \int_a^b f(x)dx$, где $f(x)$ - подынтегральная функция, непрерывная на $[a,\ b]$.

Найти: значение интеграла $I \approx I^*$ с погрешностью $\varepsilon < 0.01$

Индивидуальный вариант: $f(x) = 2sin(\sqrt{x}),\ a = 0,\ b = \frac{\pi^2}{4}$

$$\int_0^{\frac{\pi^2}{4}} 2sin(\sqrt{x})dx = 4$$

3. Теоретические сведения

3.1 Метод средних прямоугольников

Вычисление интеграла основано на приближенном вычислении площади под графиком с помощью суммирования площадей прямоугольников, ширина которых определяется шагом разбиения, а высота - значением подынтегральной функции в узле интегрирования.

$$I^* = h\sum_{i=1}^{n} f(\frac{x_{i-1}+x_i}{2}) = h\sum_{i=1}^{n} f(a + ih + \frac{h}{2}),\ h = \frac{b-a}{n}$$

3.2 Метод трапеций

Вычисление интеграла основано на приближенном вычислении площади под графиком с помощью суммирования площадей прямоугольных трапеций, высота которых определяется шагом разбиения, а основание - значением подынтегральной функции в узле интегрирования.

$$I^* = \frac{h}{2}\sum_{i=0}^{n} f(x_i) = \frac{h}{2}\left(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right),\ h = \frac{b-a}{n}$$

3.3 Метод Симпсона

Метод Симпсона заключается в приближении подынтегральной функции полиномом второй степени $y = a_i x^2 + b_i x + c_i$ на каждом участке разбиения $[a,\ b]$.

$$I^* = \frac{h}{3}\left(f(a) + f(b) + 4\left(f(x_1) + f(x_2) + \ldots + f(x_{n-1})\right) + \right.$$

$$\left. + 2\left(f(x_2) + f(x_4) + \ldots + f(x_{n-2})\right)\right),\ n = 2m$$

3.4 Метод Монте-Карло

Метод Монте-Карло - статистический метод. На отрезке $[a, b]$ строится прямоугольник с высотой $f_{max}(x)$. Случайным образом в этом прямоугольнике выбирается n точек, подсчитывается количество m точек, попавших под график функции.

$$I^* = (b - a) * f_{max}(x) * \frac{m}{n}$$

3.5 Уточнение значение интеграла

$I = I^* + O(h^k)$, где I^* - приближенное значение интеграла, k - порядок точности метода, $O(h^k) = ch^k$, h - шаг, c - константа.

Для методов средних прямоугольников и трапеций $k = 2$, для метода Симпсона $k = 4$.

Из $I = I_h^* + ch^k$ и $I = I_{\frac{h}{2}}^* + c(\frac{h}{2})^k$ следует уточненное значение интеграла: $I = I_{\frac{h}{2}}^* - \frac{I_h^* - I_{\frac{h}{2}}^*}{2^k - 1}$

4. Практическая реализация

Листинг 1. Численное интегрирование

```
package main

import (
 "fmt"
 "math"
 "math/rand"
 "time"
)

func avRect(a, b float64, n int, f func(float64) float64) float64 {
 h := (b - a) / float64(n)
 sum := 0.0

 for i := 1; i <= n; i++ {
   sum += f(a + float64(i)*h - h/2)
 }

 return h * sum
}
```

```
func trapezoid(a, b float64, n int, f func(float64) float64) float64
{
 h := (b - a) / float64(n)
 sum := (f(a) + f(b)) / 2

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

 return h * sum
}

func simps(a, b float64, n int, f func(float64) float64) float64 {
 h := (b - a) / float64(n)
 sum := f(a) + f(b)

 for i := 1; i < n; i++ {
   if i%2 == 0 {
     sum += 2 * f(a+float64(i)*h)
   } else {
     sum += 4 * f(a+float64(i)*h)
   }
 }

 return h / 3 * sum
}

func montekarl(n int, a, b, maxX float64, f func(float64) float64)
float64 {
 type point struct {
   x, y float64
 }
 points := make([]point, 0, n)
 for i := 0; i < n; i++ {
   points = append(points, point{a + rand.Float64()*(b-a),
rand.Float64() * f(maxX)})
 }
 // fmt.Println(points)
 cnt := 0
 for _, p := range points {
   if p.y <= f(p.x) {
     cnt += 1
   }
 }
 return ((b - a) * f(maxX)) * (float64(cnt) / float64(n))
```

```
}

func main() {
 rand.Seed(time.Now().UnixNano())
 rectN := 16
 trapN := 32
 simpN := 16
 e := 4.0
 tf := func(x float64) float64 {
   return 2 * math.Sin(math.Sqrt(x))
 }
 a, b := 0.0, math.Pi*math.Pi/4
 fmt.Printf("n: %d Rect: %.16f diff=%.16f correction=%.16f
corrected=%.16f cor_dif=%.16f\nn: %d Trap: %.16f diff=%.16f
correction=%.16f corrected=%.16f cor_dif=%.16f\nn: %d Simp: %.16f
diff=%.16f correction=%.16f corrected=%.16f cor_dif=%.16f\n",
   rectN,
   avRect(a, b, rectN, tf),
   math.Abs(avRect(a, b, rectN, tf)-e),
   math.Abs(avRect(a, b, rectN, tf)-avRect(a, b, 2*rectN,
tf))/(math.Exp2(2)-1),
   avRect(a, b, 2*rectN, tf)+(avRect(a, b, rectN, tf)-avRect(a, b,
2*rectN, tf))/(math.Exp2(2)-1),
   math.Abs(avRect(a, b, 2*rectN, tf)+(avRect(a, b, rectN,
tf)-avRect(a, b, 2*rectN, tf))/(math.Exp2(2)-1)-e),
   trapN,
   trapezoid(a, b, trapN, tf),
   math.Abs(trapezoid(a, b, trapN, tf)-e),
   math.Abs(trapezoid(a, b, trapN, tf)-trapezoid(a, b, 2*trapN,
tf))/(math.Exp2(2)-1),
   trapezoid(a, b, 2*trapN, tf)+(trapezoid(a, b, trapN,
tf)-trapezoid(a, b, 2*trapN, tf))/(math.Exp2(2)-1),
   math.Abs(trapezoid(a, b, 2*trapN, tf)+(trapezoid(a, b, trapN,
tf)-trapezoid(a, b, 2*trapN, tf))/(math.Exp2(2)-1)-e),
   simpN,
   simps(a, b, simpN, tf),
   math.Abs(simps(a, b, simpN, tf)-e),
   math.Abs(simps(a, b, simpN, tf)-simps(a, b, 2*simpN,
tf))/(math.Exp2(4)-1),
   simps(a, b, 2*simpN, tf)+(simps(a, b, simpN, tf)-simps(a, b,
2*simpN, tf))/(math.Exp2(4)-1),
   math.Abs(simps(a, b, 2*simpN, tf)+(simps(a, b, simpN, tf)-simps(a,
b, 2*simpN, tf))/(math.Exp2(4)-1)-e),
 )
```

```
 eps := 0.01
 cntP := 2
 m := montekarl(cntP, a, b, b, tf)
 for math.Abs(m-e) >= eps {
   cntP *= 2
   m = montekarl(cntP, a, b, b, tf)
 }
 fmt.Printf("n=%d Monte: %.16F diff=%.16f\n", cntP, m, math.Abs(m-e))
}
```

---

5. Результат

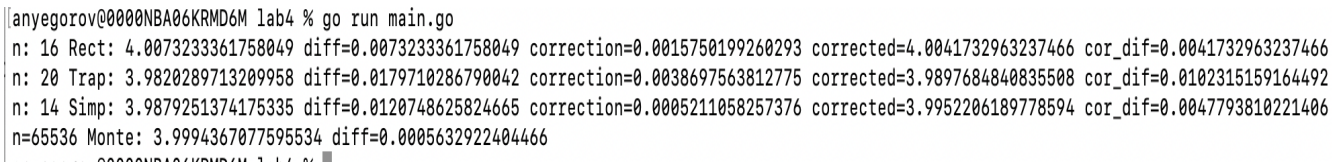

Рисунок 1. Результат работы программы

Метод	n	Вычисленное значение интеграла	Абсолютная погрешность	Уточненное значение интеграла	Абсолютная погрешность
Средних прямоугольников	16	4.0073233361758049	0.0073233361758049	4.0041732963237466	0.0041732963237466
Трапеций	20	3.9911119796005776	0.0088880203994224	3.9949407451996248	0.0050592548003752
Симпсона	14	3.9879251374175335	0.0120748625824665	3.9952206189778594	0.0047793810221406
Монте-Карло	65536	3.9994367077595534	0.0005632922404466	-	-

6. Вывод

В ходе выполнения лабораторной были реализованы 4 метода численного интегрирования: метод средних прямоугольников, метод трапеций, метод Симпсона, метод Монте-Карло. Для каждого метода, кроме метода Монте-Карло, было вычислено уточненное значение. Из первых трех методов самым точным оказался метод Симпсона, а метод трапеций - наименее точным. В методе Монте-Карло для достижения сравнимой точности необходимо выбирать большое количество случайных точек, что негативно сказывается на производительности вычислений.