БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа № 3
«Приближенное вычисление интегралов»»

Выполнил
Студент 11 группы
Сергиенко Лев Эдуардович

Минск, 2024

Применяемые составные квадратурные формулы

1. Квадратурная формула трапеций

$$\int_a^b f(x)\,dx \approx h\left(\frac{f(a)+f(b)}{2}+\sum_{i=1}^{N-1} f(x_i)\right), h=\frac{b-a}{N}, x_i \quad = a+i*h$$

2. Квадратурная формула Симпсона с четным N

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left(f(a)+4\sum_{i=1}^{N/2} f(x_{2i-1})+2\sum_{i=1}^{N/2-1} f(x_{2i})+f(b)\right), h=\frac{b-a}{N}, x_i \quad = a+i*h$$

Правило Рунге оценки погрешности

$h_1 = h, h_2 = \frac{h}{2}$

$R^{h_2} = \frac{Q^{h_2} - Q^{h_1}}{2^m - 1}$, где m - порядок точности

m для КФ трапеций = 2

m для КФ Симпсона = 4

Результаты вычислительного эксперимента, оформленные в виде таблицы

Квадратурная формула	Число разбиений	Шаг	Приближенное значение интеграла	Оценка погрешности	Абсолютная погрешность
КФ Трапеций	N = 2 N = 4 N = 8 N = 16 N = 32 N = 64 N = 128 N = 256 N = 512 N = 1024 N = 2048	h = 0.3926990817 h = 0.1963495408 h = 0.0981747704 h = 0.0490873852 h = 0.0245436926 h = 0.0122718463 h = 0.0061359232 h = 0.0030679616 h = 0.0015339808 h = 0.0007669904 h = 0.0003834952	Q = -0.2242578053 Q = -0.1722485846 Q = -0.1582158442 Q = -0.1546293162 Q = -0.1537274883 Q = -0.1535017015 Q = -0.1534452340 Q = -0.1534311159 Q = -0.1534275863 Q = -0.1534267039 Q = -0.1534264833	 R = 0.0173364069 R = 0.0046775801 R = 0.0011955093 R = 0.0003006093 R = 0.0000752623 R = 0.0000188225 R = 0.0000047061 R = 0.0000011765 R = 0.0000002941 R = 0.0000000735	|I - Q| = 0.0000000735
КФ Симпсона	N = 2 N = 4 N = 8 N = 16 N = 32 N = 64	h = 0.3926990817 h = 0.1963495408 h = 0.0981747704 h = 0.0490873852 h = 0.0245436926 h = 0.0122718463	Q = -0.1681107132 Q = -0.1549121778 Q = -0.1535382640 Q = -0.1534338068 Q = -0.1534268791 Q = -0.1534264392	 R = 0.0008799024 R = 0.0000915942 R = 0.0000069638 R = 0.0000004619 R = 0.0000000293	|I - Q| = 0.0000000294

КФ НАСТ с 4 узлами

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{4} C_i f(x_i)$$

Узлы и веса для квадратурной формулы Гаусса-Лежандра
при k = 4 на [-1, 1]
x_i: -0.33998104358485626, -0.86113631159405257,
0.86113631159405257, 0.33998104358485626
C_i: 0.65214515486254614, 0.34785484513745386,
0.34785484513745386, 0.65214515486254614

Узлы и веса для квадратурной формулы Гаусса-Лежандра
при k = 4 на [0, $\boldsymbol{\pi}$/4]
x_i: 0.2591888380879772, 0.054531642918313306,
0.7308665204791349, 0.5262093253094711
C_i: 0.2560968034487941, 0.13660227824993001,
0.13660227824993001, 0.2560968034487941

Приближенное значение интеграла, вычисленное с помощью КФ НАСТ. Сравнение с точным значением I

$Q = -0.1533992901$

$|I - Q| = 0.0000271196$

Выводы

На основании проведенного вычислительного эксперимента можно сделать следующие выводы:

1. Квадратурные формулы Симпсона и трапеций позволяют вычислять приближенные значения интегралов с заданной точностью. При этом, квадратурная формула Симпсона показывает более высокую точность вычислений по сравнению с квадратурной формулой трапеций, также она не требует большого количества разбиений.
2. Квадратурные формулы наивысшей алгебраической степени точности (КФ НАСТ) даже небольшой степени позволяют довольно точно вычислять значения интегралов. Узлы и веса для них можно брать из готовых таблиц, что делает вычисление интегралов быстрым и удобным.
3. Правило Рунге позволяет оценить погрешность вычислений и выбрать оптимальное число разбиений на отрезке.
4. При выборе метода вычисления приближенного значения интеграла необходимо учитывать требуемую точность вычислений и допустимую погрешность.

Листинг программы с комментариями

Структура программы:

```
.
├── internal/
│   └── integral/
│       ├── highestda.go
│       ├── simpson.go
│       └── trapezoid.go
└── main.go
```

highestda.go

```
package integral

import "fmt"

// Узлы и веса для квадратурной формулы Гаусса-Лежандра для n = 3, 4, 5
var gaussLegendreNodes = map[int]struct {
  nodes   []float64
  weights []float64
}{
  3: {
    nodes:   []float64{-0.7745966692414834, 0.0, 0.7745966692414834},
    weights: []float64{0.5555555555555556, 0.8888888888888888,
0.5555555555555556},
  },
  4: {
    nodes:   []float64{-0.3399810435848562 6, -0.8611363115940525 7,
0.8611363115940525 7, 0.3399810435848562 6},
    weights: []float64{0.6521451548625461 4, 0.3478548451374538 6,
0.3478548451374538 6, 0.6521451548625461 4},
  },
  5: {
    nodes:   []float64{-0.5384693101056831, -0.9061798459386640, 0.0,
0.9061798459386640, 0.5384693101056831},
    weights: []float64{0.4786286704993664 6, 0.2369268850561891,
0.5688888888888889, 0.2369268850561891, 0.4786286704993664 6},
  },
}

// Преобразование узлов и весов для отрезка [a, b]
func transformNodes(nodes, weights []float64, a, b float64) ([]float64, []float64) {
  newNodes := make([]float64, len(nodes))
```

```go
  newWeights := make([]float64, len(weights))

  copy(newNodes, nodes)
  copy(newWeights, weights)

  for i := range nodes {
    newNodes[i] = (a+b)/2.0 + (b-a)/2.0*nodes[i]
    newWeights[i] *= (b - a) / 2.0
  }

  return newNodes, newWeights
}

// IntegrateGaussLegendre вычисляет приближенное значение интеграла
функции f от a до b
// с помощью квадратурной формулы Гаусса-Лежандра с n узлами
func IntegrateGaussLegendre(f integrand, a, b float64, n int) float64 {
  if _, ok := gaussLegendreNodes[n]; !ok {
    panic("integral: unknown number of nodes for Gauss-Legendre quadrature")
  }

  nodes, weights := transformNodes(gaussLegendreNodes[n].nodes,
gaussLegendreNodes[n].weights, a, b)

  fmt.Println(nodes)
  fmt.Println(weights)

  sum := 0.0
  for i := range nodes {
    sum += weights[i] * f(nodes[i])
  }

  return sum
}
```

simpson.go

```go
package integral

import (
  "fmt"
  "math"
  "os"
)

// rungeErrorSimpson вычисляет оценку погрешности методом Рунге для
формулы Симпсона
```

```go
func rungeErrorSimpson(q2, q float64) float64 {
  return (q2 - q) / 15
}

// simpsonRule вычисляет приближенное значение интеграла функции f от a до b с помощью кф Симпсона
func simpsonRule(f integrand, a, b float64, n int) float64 {
  h := (b - a) / float64(n)
  sum := 0.0

  // Вычисляем сумму значений функции f в точках x с соответствующими коэффициентами
  for i := 1; i <= n-1; i++ {
    x := a + float64(i)*h
    coeff := 0.0

    if i%2 == 0 {
      coeff = 2.0
    } else {
      coeff = 4.0
    }

    sum += coeff * f(x)
  }

  return (h / 3.0) * (f(a) + f(b) + sum)
}

// IntegrateSimpson вычисляет приближенное значение интеграла функции f от a до b с заданной точностью e
// с помощью кф Симпсона и метода Рунге для оценки погрешности
func IntegrateSimpson(f integrand, a, b float64, e float64, logFile *os.File) float64 {
  n := 2
  prevResult := 0.0
  result := simpsonRule(f, a, b, n)

  // Вычисляем приближенное значение интеграла с заданной точностью e с помощью метода Рунге
  for math.Abs(rungeErrorSimpson(result, prevResult)) > e {
    fmt.Fprintf(logFile, "h = %.10f, Q = %.10f, R = %.10f, n = %d\n",
      (b-a)/float64(n), result, math.Abs(rungeErrorSimpson(result, prevResult)), n)

    prevResult = result
    n *= 2
    result = simpsonRule(f, a, b, n)
```

```
	}

	// Записываем результаты в лог-файл
	fmt.Fprintf(logFile, "h = %.10f, Q = %.10f, R = %.10f, n = %d\n",
		(b-a)/float64(n), result, math.Abs(rungeErrorSimpson(result, prevResult)), n)

	return result
}
```

trapezoid.go

```
package integral

import (
	"fmt"
	"math"
	"os"
)

// integrand представляет функцию, которую необходимо интегрировать
type integrand func(float64) float64

// rungeErrorTrapezoid вычисляет оценку погрешности методом Рунге
func rungeErrorTrapezoid(q2, q float64) float64 {
	return (q2 - q) / 3
}

// trapezoidalRule вычисляет приближенное значение интеграла функции f от a до b
func trapezoidalRule(f integrand, a, b float64, n int) float64 {
	h := (b - a) / float64(n)
	sum := 0.0

	// Вычисляем сумму значений функции f в точках x
	for i := 1; i <= n-1; i++ {
		x := a + float64(i)*h
		sum += f(x)
	}

	return 0.5 * h * (f(a) + 2*sum + f(b))
}

// IntegrateTrapezoidal вычисляет приближенное значение интеграла функции f от a до b с заданной точностью e
// с помощью кф трапеций и метода Рунге для оценки погрешности
func IntegrateTrapezoidal(f integrand, a, b float64, e float64, logFile *os.File) float64
{
```

```
  n := 2
  prevResult := 0.0
  result := trapezoidalRule(f, a, b, n)

  // Вычисляем приближенное значение интеграла с заданной точностью e с помощью метода Рунге
  for math.Abs(rungeErrorTrapezoid(result, prevResult)) > e {
    fmt.Fprintf(logFile, "h = %.10f, Q = %.10f, R = %.10f, n = %d\n",
      (b-a)/float64(n), result, math.Abs(rungeErrorTrapezoid(result, prevResult)), n)

    prevResult = result
    n *= 2
    result = trapezoidalRule(f, a, b, n)
  }

  // Записываем результаты в лог-файл
  fmt.Fprintf(logFile, "h = %.10f, Q = %.10f, R = %.10f, n = %d\n",
    (b-a)/float64(n), result, math.Abs(rungeErrorTrapezoid(result, prevResult)), n)

  return result
}
```

main.go

```
package main

import (
  "fmt"
  "math"
  "mv2.3/internal/integral"
  "os"
)

const (
  e = 1e-7
  a = 0
  b = math.Pi / 4.0
)

var (
  f = func(x float64) float64 {
    return math.Pow((math.Sin(x)-math.Cos(x))/(math.Sin(x)+math.Cos(x)), 3.0)
  }

  I = math.Log(math.Sqrt2) - 0.5
)
```

```go
func main() {
   logFileTrapezoid, _ := os.OpenFile("trapezoid.txt",
os.O_CREATE|os.O_WRONLY|os.O_TRUNC, 0644)
   logFileSimpson, _ := os.OpenFile("simpson.txt",
os.O_CREATE|os.O_WRONLY|os.O_TRUNC, 0644)

   resultTrapezoid := integral.IntegrateTrapezoidal(f, a, b, e, logFileTrapezoid)
   resultSimpson := integral.IntegrateSimpson(f, a, b, e, logFileSimpson)
   resultGaussLegendre := integral.IntegrateGaussLegendre(f, a, b, 4)

   fmt.Printf("Trapezoid res: %.10f\nerror: %.10f\n", resultTrapezoid,
math.Abs(I-resultTrapezoid))
   fmt.Printf("Simpson res: %.10f\nerror: %.10f\n", resultSimpson,
math.Abs(I-resultSimpson))
   fmt.Printf("GaussLegendre res: %.10f\nerror: %.10f\n", resultGaussLegendre,
math.Abs(I-resultGaussLegendre))
}
```