Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

Вариант 2

Численное решение нелинейных уравнений и систем

Выполнил:

Брагин Роман Андреевич

Проверил:

Рыбаков Степан Дмитриевич

г. Санкт-Петербург

2025

Цель работы:

Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

Вычислительная часть

Необходимо вычислить:

$$\int_{-3}^{-1} (-3x^3 - 5x^2 + 4x - 2)dx$$

Точное значение

$\int_{-3}^{-1} (-3x^3 - 5x^2 + 4x - 2)dx = -\frac{10}{3} \approx -3.3(3)$

1 Метод Ньютона-Котеса (шестого порядка при $n = 6$)

Используем формулу Ньютона-Котеса шестого порядка для 7 узлов:

$$I \approx \frac{h}{140}[41f(x_0) + 216f(x_1) + 27f(x_2) + 272f(x_3) + 27f(x_4) + 216f(x_5) + 41f(x_6) \tag{1}$$

Возьмем семь равноотстоящих точек:

$$\begin{aligned}
x_0 &= 0, & f(x_0) &= 1,\\
x_1 &= 0.1667, & f(x_1) &= 1.3611,\\
x_2 &= 0.3333, & f(x_2) &= 1.7778,\\
x_3 &= 0.5, & f(x_3) &= 2.25,\\
x_4 &= 0.6667, & f(x_4) &= 2.7778,\\
x_5 &= 0.8333, & f(x_5) &= 3.3611,\\
x_6 &= 1, & f(x_6) &= 4.
\end{aligned}$$

Подставляем в формулу и вычисляем:

$$\begin{aligned}
I_{\text{НК}} &= \frac{0.1667}{140}(41 + 293.78 + 48 + 612 + 75 + 725.4 + 164)\\
&= \frac{0.1667}{140} \times 1959.18\\
&= \frac{326.41}{140}\\
&\approx -3.3.
\end{aligned}$$

2 Методы численного интегрирования при $n = 10$

Рассмотрим интеграл:

$$I = \int_0^1 (x^2 + 2x + 1)\, dx. \tag{2}$$

2.1 Метод средних прямоугольников

$$I_{\text{СР}} = h \sum_{i=1}^{n} f\left(x_i^*\right). \tag{3}$$

2.1 Метод средних прямоугольников

$$I_{\text{СР}} = h \sum_{i=1}^{n} f\left(x_i^*\right). \tag{3}$$

Вычисляем сумму:

$$S = f(0.05) + f(0.15) + \cdots + f(0.95) = -33.05.$$

Приближенное значение:

$$I_{\text{СР}} = 0.1 \times (-33.05) = -3.305. \tag{4}$$

2.2 Метод трапеций

$$I_{\text{Тр}} = \frac{h}{2} \left[f(0) + 2 \sum_{i=1}^{n-1} f(x_i) + f(1) \right]. \tag{5}$$

Вычисляем сумму:

$$S = f(0) + 2 \sum f(x_i) + f(1) = -33.1.$$

Приближенное значение:

$$I_{\text{Тр}} = \frac{0.1}{2} \times (-33.1) = -3.31. \tag{6}$$

2.3 Метод Симпсона

$$I_{\text{Симп}} = \frac{h}{3} \left[f(0) + 4 \sum_{i=1,\ \text{нечёт}}^{n-1} f(x_i) + 2 \sum_{i=2,\ \text{чёт}}^{n-2} f(x_i) + f(1) \right]. \tag{7}$$

Вычисляем сумму:

$$S = f(0) + 4 \sum f(x_i^{\text{неч}}) + 2 \sum f(x_i^{\text{чёт}}) + f(1) = -33.$$

Приближенное значение:

$$I_{\text{Тр}} = \frac{0.1}{2} \times (-33.1) = -3.31. \tag{6}$$

2.3 Метод Симпсона

$$I_{\text{Симп}} = \frac{h}{3}\left[f(0) + 4\sum_{i=1,\ \text{нечёт}}^{n-1} f(x_i) + 2\sum_{i=2,\ \text{чёт}}^{n-2} f(x_i) + f(1)\right]. \tag{7}$$

Вычисляем сумму:

$$S = f(0) + 4\sum f(x_i^{\text{нечт}}) + 2\sum f(x_i^{\text{чёт}}) + f(1) = -33.$$

Приближенное значение:

$$I_{\text{Симп}} = \frac{0.1}{3} \times (-33) = -3.3. \tag{8}$$

3 Сравнение методов

Метод	Численное значение I	Абсолютная ошибка
Средних прямоугольников	-3.305	0.005
Трапеций	-3.31	0.01
Симпсона	-3.3	0.000
Ньютона-Котеса ($n = 6$)	-3.3	0.000

4 Выводы

1. Метод Симпсона и Ньютона-Котеса дают точный результат.
2. Метод трапеций менее точен, чем метод средних прямоугольников.
3. Увеличение n улучшает точность приближенного интегрирования.

Код

```
package main

import (
  "fmt"
  "math"
)

type Integral struct {
  Name     string
  Function func(float64) float64
  Exact    float64
}

func main() {
  integrals := []Integral{
```

```go
	{
		Name: "-x³ - x² - 2x + 1 на [0,2]",
		Function: func(x float64) float64 {
			return -math.Pow(x, 3) - math.Pow(x, 2) - 2*x + 1
		},
		Exact: -26.0 / 3.0,
	},
	{
		Name: "-3x³ - 5x² + 4x - 2 на [-3,-1]",
		Function: func(x float64) float64 {
			return -3*math.Pow(x, 3) - 5*math.Pow(x, 2) + 4*x - 2
		},
		Exact: -10.0 / 3.0,
	},
	{
		Name: "-x³ - x² + x + 3 на [0,2]",
		Function: func(x float64) float64 {
			return -math.Pow(x, 3) - math.Pow(x, 2) + x + 3
		},
		Exact: 4.0 / 3.0,
	},
}

fmt.Println("Выберите функцию для интегрирования:")
for i, integral := range integrals {
	fmt.Printf("%d) %s\n", i+1, integral.Name)
}
var choice int
fmt.Print("Ваш выбор: ")
fmt.Scan(&choice)
f := integrals[choice-1].Function

var a, b, epsilon float64
fmt.Print("Введите нижний предел интегрирования (a): ")
fmt.Scan(&a)
fmt.Print("Введите верхний предел интегрирования (b): ")
fmt.Scan(&b)
fmt.Print("Введите точность вычисления (ε): ")
fmt.Scan(&epsilon)

fmt.Println("Выберите метод интегрирования:")
fmt.Println("1) Метод левых прямоугольников")
fmt.Println("2) Метод правых прямоугольников")
fmt.Println("3) Метод средних прямоугольников")
fmt.Println("4) Метод трапеций")
fmt.Println("5) Метод Симпсона")
```

```go
  fmt.Print("Ваш выбор: ")
  var method int
  fmt.Scan(&method)

  var result float64
  var n int
  switch method {
  case 1:
    result, n = rectangleLeft(f, a, b, epsilon)
  case 2:
    result, n = rectangleRight(f, a, b, epsilon)
  case 3:
    result, n = rectangleMid(f, a, b, epsilon)
  case 4:
    result, n = trapezoidal(f, a, b, epsilon)
  case 5:
    result, n = simpson(f, a, b, epsilon)
  default:
    fmt.Println("Неверный выбор метода")
    return
  }

  fmt.Printf("\nРезультат интегрирования: %.6f\n", result)
  fmt.Printf("Число разбиений: %d\n", n)
  fmt.Printf("Приблизительное значение: %.6f\n", integrals[choice-1].Exact)
  fmt.Printf("Абсолютная погрешность: %.6f\n", math.Abs(result-
integrals[choice-1].Exact))
  fmt.Printf("Относительная погрешность: %.6f%%\n",
    math.Abs(result-integrals[choice-1].Exact)/math.Abs(integrals[choice-
1].Exact)*100)
}

func rectangleLeft(f func(float64) float64, a, b, epsilon float64) (float64, int) {
  n := 4
  var prev, result float64
  for {
    h := (b - a) / float64(n)
    sum := 0.0
    for i := 0; i < n; i++ {
      x := a + float64(i)*h
      sum += f(x)
    }
    result = sum * h

    if n > 4 && math.Abs(result-prev) < epsilon {
      break
```

```
        }
        prev = result
        n *= 2
    }
    return result, n
}

func rectangleRight(f func(float64) float64, a, b, epsilon float64) (float64, int) {
    n := 4
    var prev, result float64
    for {
        h := (b - a) / float64(n)
        sum := 0.0
        for i := 1; i <= n; i++ {
            x := a + float64(i)*h
            sum += f(x)
        }
        result = sum * h

        if n > 4 && math.Abs(result-prev) < epsilon {
            break
        }
        prev = result
        n *= 2
    }
    return result, n
}

func rectangleMid(f func(float64) float64, a, b, epsilon float64) (float64, int) {
    n := 4
    var prev, result float64
    for {
        h := (b - a) / float64(n)
        sum := 0.0
        for i := 0; i < n; i++ {
            x := a + (float64(i)+0.5)*h
            sum += f(x)
        }
        result = sum * h

        if n > 4 && math.Abs(result-prev) < epsilon {
            break
        }
        prev = result
        n *= 2
    }
```

```go
	return result, n
}

func trapezoidal(f func(float64) float64, a, b, epsilon float64) (float64, int) {
	n := 4
	var prev, result float64
	for {
		h := (b - a) / float64(n)
		sum := (f(a) + f(b)) / 2
		for i := 1; i < n; i++ {
			x := a + float64(i)*h
			sum += f(x)
		}
		result = sum * h

		if n > 4 && math.Abs(result-prev) < epsilon {
			break
		}
		prev = result
		n *= 2
	}
	return result, n
}

func simpson(f func(float64) float64, a, b, epsilon float64) (float64, int) {
	n := 4
	var prev, result float64
	for {
		h := (b - a) / float64(n)
		sum := f(a) + f(b)
		for i := 1; i < n; i++ {
			x := a + float64(i)*h
			if i%2 == 0 {
				sum += 2 * f(x)
			} else {
				sum += 4 * f(x)
			}
		}
		result = sum * h / 3

		if n > 4 && math.Abs(result-prev) < epsilon {
			break
		}
		prev = result
		n *= 2
	}
```

```
    return result, n
}
```