Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа №4

по курсу «Численные методы»

«Сравнение приближенных методов решения нелинейных уравнений»

Студент: Дьячков Е.С.

Группа: ИУ9-61Б

Преподаватель: Домрачева А.Б.

Москва 2025

1 Постановка задачи

Дано: Уравнение $f(x) = 0$, где:

$$f(x) = 2x^3 + 9x^2 - 21$$

Найти:

1. Найти все корни уравнения с точностью 0.001 методами:
   - деления отрезка пополам (бисекции)
   - Ньютона (касательных)
2. Определить количество приближений для каждого метода
3. Сравнить полученные результаты

2 Основные теоретические сведения

2.1 Метод деления отрезка пополам

Для функции $f(x)$, непрерывной на $[a, b]$, где $f(a)f(b) < 0$:

1. Вычисляем $c = \frac{a+b}{2}$
2. Если $|f(c)| < \varepsilon$, корень найден
3. Иначе выбираем подотрезок $[a, c]$ или $[c, b]$, где функция меняет знак. Если $f(a)f(c) < 0$, то этот отрезок $[a, c]$. В противном случае это отрезок $[c, b]$.

2.2 Метод Ньютона

Итерационная формула:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Условия сходимости:

- $f'(x) \neq 0$ на исследуемом отрезке
- $f''(x)$ сохраняет знак
- Начальное приближение x_0 выбирается из условия $f(x_0)f''(x_0) > 0$

3 Графическое определение интервалов корней

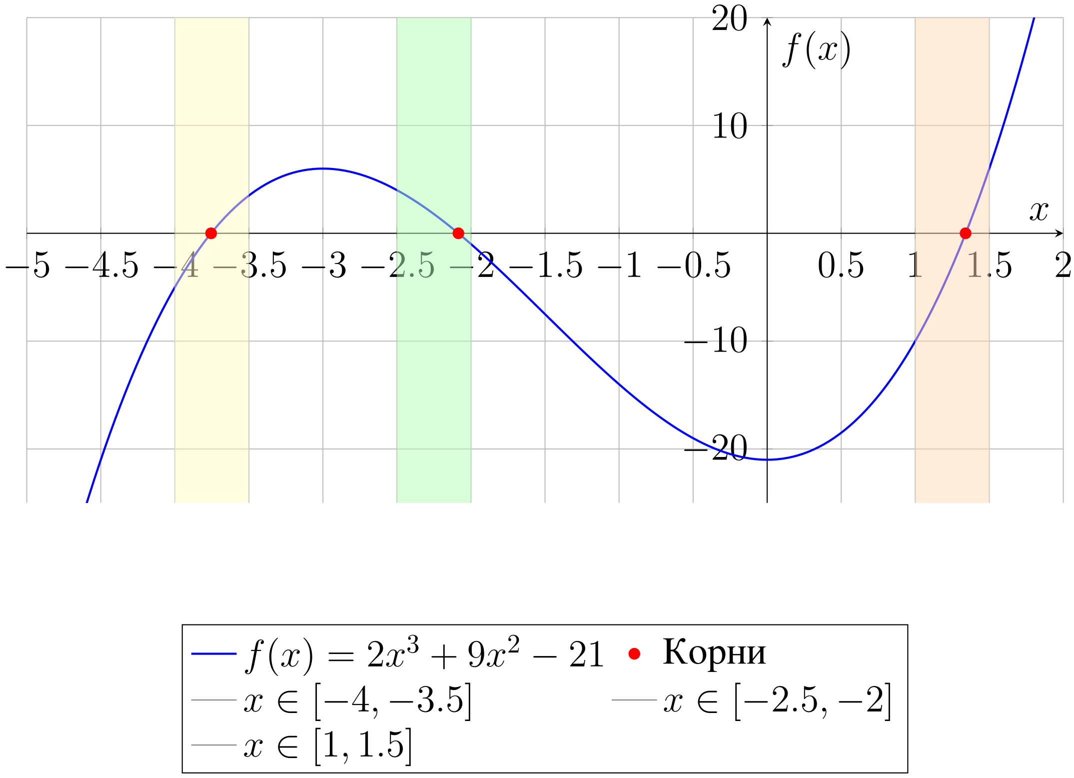

Рис. 1 — График функции с выделенными интервалами изоляции корней

По графику четко видны три интервала, содержащие корни:

- Первый корень: $x \in [-4.0; -3.5]$ (желтая область)
- Второй корень: $x \in [-2.5; -2.0]$ (зеленая область)
- Третий корень: $x \in [1.0; 1.5]$ (оранжевая область)

4 Реализация

```go
package main

import (
  "fmt"
  "math"
)

// Функция для варианта 4: f4(x) = 2³x + 9²x - 21
func f4(x float64) float64 {
  return 2*math.Pow(x, 3) + 9*math.Pow(x, 2) - 21
}

// Первая производная: f4'(x) = 6²x + 18x
func df(x float64) float64 {
  return 6*math.Pow(x, 2) + 18*x
}

// Вторая производная: ”f4(x) = 12x + 18
func d2f(x float64) float64 {
  return 12*x + 18
}

func bisection(a, b, epsilon float64) (float64, int, error) {
  if f4(a)*f4(b) >= 0 {
    return 0, 0, fmt.Errorf("неверный интервал [%.2f, %.2f]", a, b)
  }

  iterations := 0
  for {
    c := (a + b) / 2
    iterations++
    if math.Abs(f4(c)) < epsilon {
      return c, iterations, nil
    }
    if f4(a)*f4(c) < 0 {
      b = c
    } else {
      a = c
    }
  }
}

func newton(epsilon float64) ([]float64, []int, error) {
  roots := []float64{}
  iterations := []int{}

```

```
  for _, x0 := range []float64{-4.0, -2.0, 1.5} {
    if f4(x0)*d2f(x0) <= 0 {
      continue
    }

    x := x0
    iter := 0
    for {
      fx := f4(x)
      dfx := df(x)
      iter++

      if math.Abs(fx) < epsilon {
        roots = append(roots, x)
        iterations = append(iterations, iter)
        break
      }

      if dfx == 0 {
        roots = append(roots, 0)
        iterations = append(iterations, iter)
        return nil, nil, fmt.Errorf("производная равна нулю в точке x=%.4f", x)
      }
      x = x - fx/dfx
    }
  }
  return roots, iterations, nil
}

func main() {
  epsilon := 0.001
  fmt.Printf("Решаем уравнение: 2³x + 9²x - 21 = 0 ( = %.3f)\n\n", epsilon)

  intervals := [][2]float64{
    {-4, -3.5},
    {-2.5, -2},
    {1, 1.5},
  }
  fmt.Println("Найденные интервалы с корнями:", intervals)

  fmt.Println("\Методn бисекции:")
  for _, interval := range intervals {
    root, iter, err := bisection(interval[0], interval[1], epsilon)
    if err == nil {
      fmt.Printf("Корень: %.4f, итераций: %d, |f(x)|: %.6f\n", root, iter, math.Abs(f4(root)))
```

```
    } else {
      fmt.Println("Ошибка:", err)
    }
  }

  fmt.Println("\Методn Ньютона:")
  roots, iters, err := newton(epsilon)
  if err == nil {
    for i := range roots {
      fmt.Printf("Корень: %.4f, итераций: %d, |f(x)|: %.6f\n", roots[i], iters[i],
    math.Abs(f4(roots[i])))
    }
  } else {
    fmt.Println("Ошибка:", err)
  }
}
```

Листинг 1: Методы решения нелинейных уравнений

5 Результаты

Таблица 1: Сравнение методов бисекции и Ньютона

Метод	Корень	Значение x	Итераций	$\lvert f(x)\rvert$
Бисекция	1	-3.7555	12	0.000645
Бисекция	2	-2.0852	11	0.000556
Бисекция	3	1.3408	13	0.000888
Ньютона	1	-3.7555	4	0.000013
Ньютона	2	-2.0853	3	0.000013
Ньютона	3	1.3408	4	0.000000

6 Вывод

Оба метода успешно нашли все три корня уравнения с требуемой точностью $\varepsilon = 0.001$. Метод Ньютона показал преимущество в эффективности: для всех корней потребовалось в 3–4 раза меньше итераций по сравнению с методом бисекции, при этом максимальная ошибка $|f(x)|$ составила $1.3 \cdot 10^{-5}$, тогда как у метода бисекции она достигала $8.9 \cdot 10^{-4}$. Метод бисекции продемонстрировал стабильность работы: количество итераций для разных корней варьировалось в узком диапазоне (11–13), а ошибка $|f(x)|$ для всех корней оставалась близкой к заданному значению ε. Начальные приближения для метода Ньютона были выбраны оптимально: $x_1 = -4.0$, $x_2 = -2.0$ и $x_3 = 1.5$, что обеспечило быструю сходимость метода.

Полученные результаты подтверждают, что метод Ньютона является предпочтительным выбором и обеспечивает высокую скорость сходимости и малую ошибку. Ключевым фактором успешного применения метода Ньютона является корректный выбор начального приближения, поэтому для применения данного метода важен предварительный анализ функции перед проведением вычислений.