**Лабораторная работа № 4. Часть 1. Численные методы решения нелинейных уравнений**

**Цель**: **Изучить численные методы решения нелинейных уравнений**

## Задание

1. Найти корень уравнения по варианту методом половинного деления, методом простых итераций, методом Ньютонас погрешностью ε < 0.000001
2. Выполнить п.1 для ε < 0.00000000001

3. Результаты выполнения представить в виде таблицы

**Численные методы решения нелинейных уравнений**

**Существуют уравнения, которые нельзя решить аналитически. Для решения таких уравнений применяются численные методы**

1. **Метод половинного деления**

**Метод половинного деления - ч**исленный метод для нахождения корней нелинейных уравнений вида

f(x) = 0,

где f(x) — непрерывная функция на отрезке [a;b].

Метод основан на **теореме Больцано-Коши.**

### Теорема Больцано-Коши (теорема о промежуточном значении)

Если функция непрерывна на отрезке и принимает на его концах значения разных знаков, то существует точка внутри этого отрезка, в которой функция обращается в ноль. Эта точка и является корнем уравнения.

## Пример решения с ограничением по количеству итераций (требуется найти решение за 5 итераций):

### Найти корень уравнения: ****x² - 3 = 0**** на интервале [1, 2]

### Проверка условия существования корня:

f(1) = 1² - 3 = -2 < 0

f(2) = 2² - 3 = 1 > 0

f(1)×f(2) = (-2)×1 = -2 < 0 -> Корень существует

Пусть:

### а - левая граница интервала

Начальное значение: **a₀** (левый конец начального интервала)

На каждой итерации **a** обновляется

**Свойство**: f(a) всегда имеет один и тот же знак (в примере - отрицательный)

### ****b**** - правая граница интервала

Начальное значение: **b₀** (правый конец начального интервала)

На каждой итерации **b** обновляется

**Свойство**: f(b) всегда имеет один и тот же знак (в примере - положительный)

### ****c**** - середина текущего интервала

c = (a + b) / 2

**Является кандидатом на корень** на текущей итерации

**Является ключевой точкой**: по знаку f(c) выбираем новый интервал

### Итерация 1

**a = 1.000000**, **b = 2.000000**

**c = (1 + 2)/2 = 1.500000**

**f(c) = 1.5² - 3 = 2.25 - 3 = -0.75 < 0**

**Выбор**: f(c)×f(b) < 0 → корень в [1.5, 2.0]

**Новый интервал**: a = 1.5, b = 2.0

### Итерация 2

**a = 1.500000**, **b = 2.000000**

**c = (1.5 + 2)/2 = 1.750000**

**f(c) = 1.75² - 3 = 3.0625 - 3 = 0.0625 > 0**

**Выбор**: f(a)×f(c) < 0 → корень в [1.5, 1.75]

**Новый интервал**: a = 1.5, b = 1.75

### Итерация 3

**a = 1.500000**, **b = 1.750000**

**c = (1.5 + 1.75)/2 = 1.625000**

**f(c) = 1.625² - 3 = 2.6406 - 3 = -0.3594 < 0**

**Выбор**: f(c)×f(b) < 0 → корень в [1.625, 1.75]

**Новый интервал**: a = 1.625, b = 1.75

### Итерация 4

**a = 1.625000**, **b = 1.750000**

**c = (1.625 + 1.75)/2 = 1.687500**

**f(c) = 1.6875² - 3 = 2.8477 - 3 = -0.1523 < 0**

**Выбор**: f(c)×f(b) < 0 → корень в [1.6875, 1.75]

**Новый интервал**: a = 1.6875, b = 1.75

### Итерация 5

**a = 1.687500**, **b = 1.750000**

**c = (1.6875 + 1.75)/2 = 1.718750**

**f(c) = 1.71875² - 3 = 2.9541 - 3 = -0.0459 < 0**

**Выбор**: f(c)×f(b) < 0 → корень в [1.71875, 1.75]

**Новый интервал**: a = 1.71875, b = 1.75

|  n | a | b | c | f(c) | Интервал |
| --- | --- | --- | --- | --- | --- |
|  0 | 1.000000 | 2.000000 | 1.500000 | -0.750000 | [1.0, 2.0] |
|  1 | 1.500000 | 2.000000 | 1.750000 | +0.062500 | [1.5, 2.0] |
|  2 | 1.500000 | 1.750000 | 1.625000 | -0.359375 | [1.5, 1.75] |
|  3 | 1.625000 | 1.750000 | 1.687500 | -0.152344 | [1.625, 1.75] |
|  4 | 1.687500 | 1.750000 | 1.718750 | -0.045898 | [1.6875, 1.75] |

## Результат после 5 итераций:

**Лучшее приближение**: **c = 1.718750**

**Значение функции**: **f(c) = -0.045898**

**Интервал**: [1.71875, 1.75] длиной **0.03125**

**Погрешность**: ≤ 0.015625

## Проверка точности

**Точный корень**: √3 ≈ 1.7320508 ( в данном случае мы можем найти корень еще и аналитичиски)

**Наше приближение**: 1.718750

**Абсолютная погрешность**: |1.732051 - 1.718750| ≈ 0.0133

### Алгоритм метода

#### Шаг 1: Подготовка

1. Выбрать начальный интервал [a₀,b₀] такой, что f(a₀)×f(b₀) < 0
2. Задать точность ε > 0
3. Задать максимальное число итераций Nmax

#### Шаг 2: Итерационный процесс

Для n = 0, 1, 2, ..., Nmax:

1. **Вычислить середину интервала**:
cₙ = (aₙ + bₙ) / 2
2. **Проверить точность**:
Если |bₙ - aₙ| < 2ε или |f(cₙ)| < ε → КОРЕНЬ НАЙДЕН
3. **Выбрать новый интервал**:
	* Если f(aₙ)×f(cₙ) < 0 → корень в [aₙ, cₙ] → bₙ₊₁ = cₙ, aₙ₊₁ = aₙ
	* Если f(cₙ)×f(bₙ) < 0 → корень в [cₙ, bₙ] → aₙ₊₁ = cₙ, bₙ₊₁ = bₙ
	* Если f(cₙ) = 0 → корень найден

### Критерии остановки

1. **По длине интервала**: |bₙ - aₙ| < 2ε
2. **По значению функции** |f(cₙ)| < ε
3. **По количеству итераций**: n ≥ Nmax

### Оценка погрешности

После n итераций погрешность оценивается как:
|ξ - cₙ| ≤ (b₀ - a₀) / 2ⁿ⁺¹

где ξ — точный корень.

## ****Рекомендации по выбору ε:****

### ****Требования задачи****

**Инженерные расчеты**: ε = 10⁻³ ÷ 10⁻⁶ (0.001 ÷ 0.000001)

**Научные исследования**: ε = 10⁻⁶ ÷ 10⁻¹²

**Учебные примеры**: ε = 10⁻² ÷ 10⁻⁴ (0.01 ÷ 0.0001)

### ****Вычислительные ресурсы****

*# Зависимость итераций от ε для метода половинного деления:*

ε = 0.1 → ~4 итерации

ε = 0.01 → ~7 итераций

ε = 0.001 → ~10 итераций

ε = 0.0001 → ~14 итераций

ε = 0.000001 → ~20 итераций

### ****Общие правила:****

*# Учебные задачи*

ε = 1e-3 *# Хороший баланс точности и наглядности*

*# Инженерные расчеты*

ε = 1e-6 *# Стандартная точность*

*# Научные исследования*

ε = 1e-12 *# Высокая точность*

1. **Метод простой итерации**

**Метод простой итерации** - численный метод для нахождения корней нелинейных уравнений вида f(x) = 0, где f(x) — непрерывная функция. Метод основан на приведении уравнения к виду x = φ(x) и построении итерационной последовательности.

**Теорема о сжатом отображении**
Если функция φ(x) является сжатием на отрезке [a,b] (т.е. |φ'(x)| ≤ q < 1 для всех x ∈ [a,b]), то итерационный процесс xₙ₊₁ = φ(xₙ) сходится к единственному корню на этом отрезке.

Алгоритм:

### ****ШАГ 1: ПОДГОТОВКА****

1. **Преобразование исходного уравнения f(x) = 0** к виду **x = φ(x)**

Пример: x² - 3 = 0 → x = (x + 3/x)/2

(здесь для преобразования используем формулу Ньютона:

Формула Ньютона:

xₙ₊₁ = xₙ - f(xₙ)/f'(xₙ)

Для f(x) = x² - 3:

f'(x) = 2x

xₙ₊₁ = xₙ - (xₙ² - 3)/(2xₙ)

xₙ₊₁ = (xₙ + 3/xₙ)/2

1. **Проверка условия сходимости**

Вычисляем производную φ'(x)

**Теорема:** Если |φ'(x)| < 1 в окрестности корня, то метод сходится

Проверяем: |φ'(x)| < 1 в окрестности корня

Если условие выполняется → метод сходится

φ(x) = (x + 3/x)/2

φ'(x) = (1 - 3/x²)/2

**Находим окрестность корня:**

* Сначала находим грубый интервал расположения корня:

f(1) = 1² - 3 = -2 < 0

f(2) = 2² - 3 = 1 > 0

→ Корень между 1 и 2

* Проверяем |φ'(x)| на этом интервале:

φ'(1) = (1 - 3/1)/2 = -1 → |φ'(1)| = 1

φ'(1.5) = (1 - 3/2.25)/2 = (1 - 1.333)/2 = -0.167 → |φ'(1.5)| = 0.167

φ'(2) = (1 - 3/4)/2 = (1 - 0.75)/2 = 0.125 → |φ'(2)| = 0.125

* Выбираем "хороший" подотрезок где **|φ'(x)| < 1**:

На [1.5, 2.0]: max|φ'(x)| = 0.167 < 1

Метод гарантированно сходится при начальном приближении из [1.5, 2.0]

1. **Выбор начального приближения**

**Правила выбора x₀:**

* Должен лежать в области где |φ'(x)| < 1
* Желательно ближе к предполагаемому корню
* Для надежности берем середину "хорошего" интервала

x₀ = 1.7 (лежит в [1.5, 2.0] где |φ'(x)| < 1)

1. **Задание параметров точности**

ε - требуемая точность

Nmax - максимальное число итераций

### ****ШАГ 2: ИТЕРАЦИОННЫЙ ПРОЦЕСС****

Для n = 0, 1, 2, ..., Nmax:

1. **Вычисляем новое приближение:**

xₙ₊₁ = φ(xₙ)

1. **Проверяем точность:**

Вычисляем разность: Δ = |xₙ₊₁ - xₙ|

Вычисляем: R = |f(xₙ₊₁)|

Если **Δ < ε** или **R < ε** → КОРЕНЬ НАЙДЕН

1. **Переходим к следующей итерации:**

xₙ = xₙ₊₁

n = n + 1

**На нашем примере:**

**Итерация 0 (n = 0):**

x₀ = 1.700000

x₁ = (1.7 + 3/1.7)/2 = (1.7 + 1.764706)/2 = 1.732353

Δ = |1.732353 - 1.700000| = 0.032353 > 0.001 (здесь ε = 0.001) (условие не выполняется)

R = |1.732353² - 3| = |3.001041 - 3| = 0.001041 > 0.001 (условие не выполняется)

**Итерация 1 (n = 1):**

x₁ = 1.732353

x₂ = (1.732353 + 3/1.732353)/2 = (1.732353 + 1.732051)/2 = 1.732202

Δ = |1.732202 - 1.732353| = 0.000151 > 0.001 (условие не выполняется)

R = |1.732202² - 3| ≈ 0.000000 < 0.001 (условие выполняется)

Корень найден на итерации 1 по критерию 2

## ****Критерии останова****

1. **По изменению приближения:** |xₙ₊₁ - xₙ| < ε
2. **По значению функции:** |f(xₙ₊₁)| < ε
3. **По числу итераций:** n ≥ Nmax

**Рекомендуется использовать первые два критерия вместе.**

1. **Метод Ньютона**

## ****Алгоритм:****

## ****ШАГ 1: ПОДГОТОВКА****

### ****1.1. ПРОВЕРКА УСЛОВИЙ ПРИМЕНИМОСТИ****

**Необходимые условия:**

1. **f(x) ∈ C²[a,b]** - функция дважды непрерывно дифференцируема на [a,b]
2. **f(a)×f(b) < 0** - на концах отрезка разные знаки
3. **f'(x) ≠ 0** на [a,b] - производная не обращается в ноль
4. **f''(x)** не меняет знак на [a,b] - вторая производная знакопостоянна

### ****1.2. ВЫБОР НАЧАЛЬНОГО ПРИБЛИЖЕНИЯ****

**Правило выбора x₀:**

Выбираем тот конец отрезка [a,b], где:

f(x₀) × f''(x₀) > 0

### ****1.3. ЗАДАНИЕ ПАРАМЕТРОВ ТОЧНОСТИ****

ε = 0.001 # Требуемая точность

Nmax = 20 # Максимальное число итераций

## ****ШАГ 2: ИТЕРАЦИОННЫЙ ПРОЦЕСС****

### ****2.1. ОСНОВНАЯ ФОРМУЛА МЕТОДА НЬЮТОНА****

 f(xₙ)

xₙ₊₁ = xₙ - —————

 f'(xₙ)

### ****2.2. Алгоритм вычислений****

Для **n = 0, 1, 2, ..., Nmax**:

#### ****1. Вычисление значений функции и производной****

yₙ = f(xₙ)

y'ₙ = f'(xₙ)

#### ****2. Проверка ошибок****

Если y'ₙ = 0 → ОШИБКА: производная равна нулю

Если |y'ₙ| очень мала → ВОЗМОЖНА расходимость

#### ****3. Вычисление нового проиближения****

text

 yₙ

xₙ₊₁ = xₙ - ———

 y'ₙ

#### ****4. Проверка точности****

**Вычисляем критерии:**

Δ = |xₙ₊₁ - xₙ| # Изменение приближения

R = |f(xₙ₊₁)| # Невязка (значение функции)

**Проверяем условия останова:**

* Если **Δ < ε** → корень найден
* Если **R < ε** → корень найден
* Если **n ≥ Nmax** → достигнут предел итераций

## Пример:

**Уравнение:** x² - 3 = 0
**Производная:** f'(x) = 2x
**Начальное приближение:** x₀ = 2.0 (т.к. f(2)>0 и f''(2)=2>0)
**Точность:** ε = 0.001

**Итерация 0 (n = 0):**

x₀ = 2.000000

f(x₀) = 2.0² - 3 = 4 - 3 = 1.000000

f'(x₀) = 2 × 2.0 = 4.000000

x₁ = 2.0 - 1.0/4.0 = 2.0 - 0.25 = 1.750000

Δ = |1.750000 - 2.000000| = 0.250000 > 0.001 (условие не выполняется)

**Итерация 1 (n = 1):**

x₁ = 1.750000

f(x₁) = 1.75² - 3 = 3.0625 - 3 = 0.062500

f'(x₁) = 2 × 1.75 = 3.500000

x₂ = 1.75 - 0.0625/3.5 = 1.75 - 0.017857 = 1.732143

Δ = |1.732143 - 1.750000| = 0.017857 > 0.001 (условие не выполняется)

**Итерация 2 (n = 2):**

x₂ = 1.732143

f(x₂) = 1.732143² - 3 = 3.000318 - 3 = 0.000318

f'(x₂) = 2 × 1.732143 = 3.464286

x₃ = 1.732143 - 0.000318/3.464286 = 1.732143 - 0.000092 = 1.732051

Δ = |1.732051 - 1.732143| = 0.000092 < 0.001 (условие выполняется)

Корень найден на итерации 2

## ****Критерии останова****

### ****Основные критерии:****

1. **|xₙ₊₁ - xₙ| < ε** - приближения почти не меняются
2. **|f(xₙ₊₁)| < ε** - функция близка к нулю

### ****Аварийные критерии:****

1. **n ≥ Nmax** - превышено максимальное число итераций
2. **f'(xₙ) = 0** - производная обратилась в ноль
3. **|xₙ₊₁ - xₙ| > M** - расходимость (M - большое число)

**Найденный корень:** x ≈ 1.732051

**Точность:** |Δx| = 0.000092 < 0.001

|  Параметр | Метод половинного деления | Метод простых итераций | Метод Ньютона |
| --- | --- | --- | --- |
| **Скорость сходимости** | Линейная O(1/2ⁿ) | Линейная O(qⁿ) | Квадратичная O(ε²ⁿ) |
| **Класс сложности** | Логарифмический O(log(1/ε)) | Линейный O(log(1/ε)/log(1/q)) | Логарифмический O(log log(1/ε)) |
| **Сходимость** | Всегда сходится | При |φ'(x)| < 1 | При удачном x₀ |
| **Надежность** | Очень высокая | Средняя | Низкая |
| **Простота** | Очень простая | Простая | Средняя |
| **Требования** | f(a)×f(b) < 0 | Преобразование x = φ(x) | f'(x) ≠ 0 |

**Вариант выбирается по номеру списка**

**Варианты заданий**

1. x³ - 2x - 5 = 0
2. x³ + x - 1 = 0
3. x² - sin(x) - 2 = 0
4. eˣ - 3x = 0
5. cos(x) - x = 0
6. x - ln(x) - 2 = 0
7. sin(x) - x/2 = 0
8. x³ - 3x + 1 = 0
9. eˣ - 2x - 1 = 0
10. x² - cos(x) = 0
11. x - 2⁻ˣ = 0
12. x³ - 2 = 0
13. cos(x) - x/2 = 0
14. e⁻ˣ - x = 0
15. x² - ln(x+1) = 0
16. sin(x) + x - 1 = 0
17. x³ - x - 1 = 0
18. eˣ - x² - 2 = 0
19. cos(x) - 2x = 0
20. x - √(x+2) = 0
21. x³ + 2x - 5 = 0
22. eˣ - 4x = 0
23. sin(x) - 0.5x = 0
24. x² - eˣ + 2 = 0
25. cos(x) - x + 1 = 0
26. x - 3/x = 0
27. eˣ - 3x + 1 = 0
28. sin(x) - x² = 0
29. x³ - 4x + 2 = 0
30. cos(x) - 0.5x = 0