УДК 004:005:51

РЕАЛІЗАЦІЯ ЧИСЕЛЬНИХ МЕТОДІВ ДЛЯ РОЗВ'ЯЗАННЯ НЕЛІНІЙНИХ РІВНЯНЬ У ПРОГРАМУВАННІ

А. І. Козьбан, І. В. Фриз

Анотація. У статті розглянуто застосування чисельних методів для розв'язання нелінійних рівнянь у сучасному програмуванні. На основі теоретичних основ та лабораторних матеріалів проаналізовано методи дихотомії, хорд, Ньютона та простої ітерації. Показано практичну реалізацію цих методів на мові C# з реальними прикладами коду. Отримані результати демонструють зручність, швидкість, точність програмної реалізації для задач інженерії, фізики та економіки.

Ключові слова: чисельні методи, нелінійні рівняння, метод Ньютона, метод хорд, точність обчислень.

Вступ. Нелінійні рівняння зустрічаються у багатьох наукових та інженерних задачах. Проблема їх розв'язання полягає в тому, що більшість таких рівнянь не мають аналітичних розв'язків або їх знаходження є надзвичайно складним [3]. Це вимагає застосування чисельних методів, які дають змогу отримувати точні наближені значення за допомогою ітераційних процесів. Різні методи обчислень допомагають швидко обробляти великі обсяги даних і знаходити корені з високою точністю, роблячи їх незамінними в сучасних розрахунках [1; 4].

Сьогодні програмування перетворює ці методи на потужний практичний інструмент. Замість ручних обчислень на папері ми можемо написати програму, яка за лічені секунди знайде потрібний корінь з будь-якою заданою похибкою. Особливо зручною для цього є мова C#, адже вона поєднує простоту синтаксису, високу швидкість виконання та можливість створення як простих консольних додатків, так і складних програм із графічним інтерфейсом [2; 5].

У реальному світі нелінійні рівняння описують найрізноманітніші процеси: рух тіл під дією сил, хімічні реакції, оптимізацію економічних показників чи розрахунок міцності конструкцій. Аналітичні методи часто виявляються безсилими, тому чисельні алгоритми стають єдиним надійним способом отримати результат. Програмна реалізація цих алгоритмів не тільки прискорює обчислення, але й дає змогу легко перевіряти різні початкові наближення та порівнювати ефективність методів.

Метою статті є дослідження застосування чисельних методів розв'язання нелінійних рівнянь у програмуванні на прикладі мови C#. Аналіз цих методів дасть змогу не тільки глибше зрозуміти теоретичні основи, а й отримати готовий робочий інструмент, який можна відразу застосовувати в дипломних проєктах чи реальних розробках.

Основна частина. Теоретичною основою розв'язання нелінійних рівнянь є знаходження кореня рівняння виду $f(x) = 0$, де $f(x)$ – неперервна функція. Аналітичний розв'язок часто неможливий, тому використовують чисельні методи. Спочатку необхідно відокремити корінь – знайти відрізок$[a, b]$, на якому функція змінює знак $(f(a) * f(b) < 0)$ і існує лише один корінь. Після цього застосовують методи уточнення, які дають змогу наблизити значення кореня з будь-якою заданою точністю [1; 4].

Усі розглянуті методи належать до ітераційних. Вони починають із початкового наближення і крок за кроком покращують його, доки похибка не стане меншою за задану величину ε (наприклад, 10^{-6}). Важливою перевагою програмної реалізації на C# є те, що комп'ютер може виконати тисячі ітерацій за частки секунди, автоматично перевіряти умови збіжності та виводити проміжні результати в зручному вигляді.

Найпростішим і найнадійнішим є метод дихотомії. Він гарантовано збігається, якщо функція неперервна і змінює знак на відрізку. Алгоритм полягає в тому, що на кожній ітерації відрізок ділять навпіл і обирають ту половину, де функція змінює знак.

Формула для нового наближення:

$$x_{k+1} = \frac{a_k + b_k}{2}. \quad (1)$$

Умови зупинки: $|b_k - a_k| < \varepsilon$ або $|f(x_{k+1})| < \varepsilon$.

Перевагою є те, що завжди все збігається, і водночас не потребує похідної. Проте є момент із повільною збіжністю (лише один біт точності за ітерацію). Реалізуємо метод дихотомії на C#. Наприклад, візьмемо $x^2 - 2 = 0$.

Лістинг програми:

```
using System;
class DichotomyMethod
{
   static double F(double x) => x * x - 2;
   public static void Main()
   {
      double a = 1.0, b = 2.0;
      double eps = 1e-6;
      int iter = 0;
      while (Math.Abs(b - a) > eps)
      {
         double c = (a + b) / 2;
         if (F(a) * F(c) < 0) b = c;
         else a = c;
         iter++;
         Console.WriteLine($"Ітерація {iter}: x = {c:F8}, f(x) = {F(c):F8}");
      }
      Console.WriteLine($"Корінь: { (a + b)/2 :F8} за {iter} ітерацій");
   }
}
```

Результат роботи програми наведений на рис. 1.

```
Ітерація 1: x = 1.50000000, f(x) = 0.25000000
Ітерація 2: x = 1.25000000, f(x) = -0.43750000
Ітерація 3: x = 1.37500000, f(x) = -0.10937500
Ітерація 4: x = 1.43750000, f(x) = 0.06640625
Ітерація 5: x = 1.40625000, f(x) = -0.02246094
Ітерація 6: x = 1.42187500, f(x) = 0.02172852
Ітерація 7: x = 1.41406250, f(x) = -0.00042725
Ітерація 8: x = 1.41796875, f(x) = 0.01063538
Ітерація 9: x = 1.41601563, f(x) = 0.00510025
Ітерація 10: x = 1.41503906, f(x) = 0.00233555
Ітерація 11: x = 1.41455078, f(x) = 0.00095391
Ітерація 12: x = 1.41430664, f(x) = 0.00026327
Ітерація 13: x = 1.41418457, f(x) = -0.00008200
Ітерація 14: x = 1.41424561, f(x) = 0.00009063
Ітерація 15: x = 1.41421509, f(x) = 0.00000431
Ітерація 16: x = 1.41419983, f(x) = -0.00003884
Ітерація 17: x = 1.41420746, f(x) = -0.00001726
Ітерація 18: x = 1.41421127, f(x) = -0.00000647
Ітерація 19: x = 1.41421318, f(x) = -0.00000108
Ітерація 20: x = 1.41421413, f(x) = 0.00000162
Корінь: 1.41421366 за 20 ітерацій
```

Рис. 1. Результат роботи методу дихотомії

Наступним є метод хорд. Його ще називають секантний метод. Метод використовує дві точки замість похідної. Нова точка лежить на прямій, що з'єднує попередні наближення. Загальною формулою є:

$$x_{k-1} = x_k - f(x_k) * \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}. \quad (2)$$

Цей метод є швидшим за метод дихотомії і також не потребує обчислення похідної. Але якщо функція сильно вигнута, то іноді може не збігатися. Реалізуємо метод хорд на C#. Приклад візьмемо той самий: $x^2 - 2 = 0$.

Лістинг основної частини програми:

```
{
    double a = 1.0, b = 2.0;
    double eps = 1e-6;
    int iter = 0;
    double c;
    do
    {
       c = b - F(b) * (b - a) / (F(b) - F(a));
       a = b;
       b = c;
       iter++;
       …
    } while (Math.Abs(F(c)) > eps);
    …
}
```

Результат роботи програми наведений на рис. 2.

```
Ітерація 1: x = 1.33333333, f(x) = -0.22222222
Ітерація 2: x = 1.40000000, f(x) = -0.04000000
Ітерація 3: x = 1.41463415, f(x) = 0.00118977
Ітерація 4: x = 1.41421144, f(x) = -0.00000601
Ітерація 5: x = 1.41421356, f(x) = 0.00000000
Корінь: 1.41421356 за 5 ітерацій
```

Рис. 2. Результат роботи методу хорд

Найшвидшим методом під час доброго початкового наближення є метод Ньютона. Використовуємо похідну для визначення напряму та кроку. Реалізується у формулі:

$$x_{k-1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{3}$$

Перевагою цього методу є дуже швидка квадратична збіжність. Але метод потребує похідної і може розходитися у разі поганого старту. Метод використовує дві точки замість похідної. Нова точка лежить на прямій, що з'єднує попередні наближення. Реалізуємо метод Ньютона на C#. Приклад візьмемо той самий: $x^2 - 2 = 0$.

Лістинг основної частини програми:

```
static double F(double x) => x * x - 2;
static double DF(double x) => 2 * x;   // похідна
public static void Main()
{
    double x = 1.0;
    double eps = 1e-6;
    int iter = 0;
    while (Math.Abs(F(x)) > eps)
    {
       x = x - F(x) / DF(x);
       iter++;
       …
    }
    …
}
```

Результат роботи програми наведений на рис. 3.

```
Ітерація 1: x = 1.50000000, f(x) = 0.25000000
Ітерація 2: x = 1.41666667, f(x) = 0.00694444
Ітерація 3: x = 1.41421569, f(x) = 0.00000601
Ітерація 4: x = 1.41421356, f(x) = 0.00000000
Корінь: 1.41421356 за 4 ітерацій
```

Рис. 3. Результат роботи методу Ньютона

Розглянемо останній метод простої ітерації. Він полягає у перетворенні вихідного рівняння $f(x) = 0$ до еквівалентного вигляду $x = \varphi(x)$. Після цього виконується ітераційний процес $x_{k+1} = \varphi(x_k)$, який продовжується, доки різниця між двома послідовними наближеннями не стане меншою за задану точність ε. Реалізуємо метод простої ітерації у C#. Лістинг основної частини програми:

```
{
   // Функція φ(x) для рівняння x² - 2 = 0
   static double Phi(double x) => (x + 2.0 / x) / 2.0;
   public static void Main()
   {
      double x = 1.0;          // початкове наближення (можна змінювати)
      double eps = 1e-6;       // задана точність
      int iter = 0;
      double x_old;
      do
      {
         x_old = x;
         x = Phi(x_old);
         iter++;
         …
      }
      while (Math.Abs(x - x_old) > eps);
```

Результат роботи програми наведений на рис. 4.

```
Ітерація  1: x = 1.5000000000   |x_new - x_old| = 5.00000000E-001
Ітерація  2: x = 1.4166666667   |x_new - x_old| = 8.33333333E-002
Ітерація  3: x = 1.4142156863   |x_new - x_old| = 2.45098039E-003
Ітерація  4: x = 1.4142135624   |x_new - x_old| = 2.12389982E-006
Ітерація  5: x = 1.4142135624   |x_new - x_old| = 1.59494640E-012

Корінь знайдено: 1.4142135624
Кількість ітерацій: 5
```

Рис. 4. Результат роботи методу простої ітерації

Порахуємо абсолютну похибку для кожного методу на основі рівняння $x^2 - 2 = 0$, порівняння відобразимо у табл. 1.

Таблиця 1

Результати оцінки похибку методів обчислень

Метод	Кількість ітерацій	Знайдений корінь	Абсолютна похибка
Дихотомії	20	1,41421366	$7{,}696 * 10^{-5}$
Хорд	5	1,41421356	$2{,}37 * 10^{-9}$
Ньютона	4	1,41421356	$2{,}37 * 10^{-9}$
Проста ітерація	5	1,4142135624	$6{,}16 * 10^{-8}$

Висновки. Аналіз результатів показує, що метод Ньютона найшвидший, але потребує похідної. Метод хорд – універсальний і надійний. Дихотомія – найповільніша, але гарантує результат навіть у разі поганого старту. Програмна реалізація на C# дає змогу за секунди протестувати всі методи і обрати найкращий для конкретної задачі.

У сфері інформаційних технологій такі алгоритми використовуються для оптимізації, машинного навчання, моделювання фізичних процесів та криптографії. Можливість швидко змінювати функцію $f(x)$ в коді робить C# ідеальною платформою для експериментів.

Abstract. The article examines the application of numerical methods for solving nonlinear equations in modern programming. Based on theoretical foundations and practical examples, the methods of dichotomy, chords, Newton, and simple iteration are analyzed. The practical implementation of these methods using the C# programming language is demonstrated with complete working code examples and comparison tables. The obtained results show the convenience, high speed, and accuracy of software implementation for real-world tasks in engineering, physics, economics, and information technology. The study confirms that C# is an ideal platform for rapid testing and selection of the most suitable numerical method depending on the specific problem.

Keywords: numerical methods, nonlinear equations, Newton method, chord method, computational accuracy.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Дичка І. А., Онай М. В., Гадиняк Р. А. Числові методи. Розв'язання задач лінійної алгебри та нелінійних рівнянь: лабораторний практикум. Київ: КПІ ім. Ігоря Сікорського, 2018. 95 с. URL: https://ela.kpi.ua/bitstreams/4d51b9f7-96eb-4d39-9886-8785c212968d/download

2. Тронь Д. В., Волонтир Л. О. Чисельні методи розв'язання нелінійних рівнянь на C#. URL: http://jait.donnu.edu.ua/article/view/17028/16921

3. Гончаров О. А., Васильєва Л. В., Юнда А. М. Чисельні методи розв'язання прикладних задач: навчальний посібник. Суми: Сумський державний університет, 2020. 142 с. URL: https://essuir.sumdu.edu.ua/server/api/core/bitstreams/0f8e860a-c101-4db8-9e54-c823abb78b76/content

4. Чисельні методи: навчальний посібник / Л. О. Волонтир, О. В. Зелінська, Н. А. Потапова, І. А. Чіков. Вінниця: ВНАУ, 2020. 322 с. URL: https://r.donnu.edu.ua/handle/123456789/1805

5. Задачин В. М., Конюшенко І. Г. Чисельні методи: навчальний посібник. Харків: ХНЕУ ім. С. Кузнеця, 2014. 180 с. URL: https://repository.hneu.edu.ua/handle/123456789/8310

УДК 004:005:51

ЗАСТОСУВАННЯ ЧИСЕЛЬНОГО ІНТЕГРУВАННЯ ЗА МЕТОДОМ СІМПСОНА ДЛЯ ОЦІНКИ СУМАРНИХ ОБСЯГІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНИХ ПОСЛУГ

Д. О. Круцюк, Ю. В. Поремський

Анотація. Робота присвячена глибокому теоретичному та практичному дослідженню одного з найпоширеніших методів чисельного інтегрування – формули Сімпсона. У статті детально розглянуто математичний апарат, що лежить в основі алгоритму, зокрема принцип заміни підінтегральної функції інтерполяційним багаточленом другого степеня. Наведено повне аналітичне виведення узагальненої формули для складеного квадратурного правила. Значну увагу приділено дослідженню залишкового члена та оцінці абсолютної похибки обчислень, що має фундаментальне значення для визначення меж застосовності методу. Як практичний приклад розглянуто задачу інтегрування складної трансцендентної функції, первісна якої не виражається в елементарних функціях. Результати дослідження підтверджують високий порядок точності методу Сімпсона, порівняно з лінійними методами наближеного інтегрування.

Ключові слова: чисельне інтегрування, визначений інтеграл, метод Сімпсона, квадратурна формула.

Вступ. У сучасній обчислювальній математиці задачі знаходження визначених інтегралів посідають одне з ключових місць. Потреба в обчисленні інтегралів виникає під час розв'язання широкого спектра інженерних, фізичних та інформаційних задач: від аналізу нелінійних динамічних систем до розрахунку ймовірностей у математичній статистиці та теорії інформації. Проте класичний апарат математичного аналізу, що спирається на формулу Ньютона–Лейбніца, має суттєві обмеження. Далеко не кожна підінтегральна функція має первісну, яку можна виразити через скінченну комбінацію елементарних функцій (класичними прикладами