УДК 519.72 DOI: 10.33111/mise.98.21

Чугаєва О. В.,
асистентка кафедри комп'ютерної математики
та інформаційної безпеки, Київський національний
економічний університет імені Вадима Гетьмана

Chuhaveva O. V.,
Assistant of the Computer Mathematics
and Information Security Department,
Kyiv National Economic University named after Vadym Hetman

МАТЕМАТИЧНИЙ ІНСТРУМЕНТАРІЙ І МЕТОДИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ ДЛЯ ЗАСТОСУВАННЯ В КРИПТОАНАЛІЗІ

MATHEMATICAL TOOLS AND METHODS OF COMPUTERMATHEMATICS IN APPLICATION CRYPTOGRAPFIC ANALYSIS

Анотація. *В роботі розглядаються відомі чисельні методи розв'язання алгебраїчних рівнянь, а також викладається новий чисельний метод розв'язання алгебраїчних рівнянь, який базується на розкладі многочлена на множники. Цей метод дозволяє виділяти групи кратних та близьких до кратних коренів рівняння. В окремих випадках цей метод призводить до добре відомих методів: методу січних, методу Ньютона, методу Ліна, методу інтерполяції. Зауважимо, що даний метод можна застосовувати лише для розв'язування алгебраїчних рівнянь. Для порівняння наводяться деякі відомі методи розв'язання довільних рівнянь з однією невідомою. Для усіх методів наведено приклади програм, складені на мові С++ в пакеті комп'ютерної математики MAPLE.*

Криптоаналіз — це наука про методи отримання вихідного значення зашифрованої інформації без доступу до секретного ключа. Головна мета криптоанализа — знаходження ключа. Вперше термін «криптоаналіз» ввів американський криптограф Вільям Ф. Фрідман в 1920 році. Під терміном «криптоаналіз» також розуміють спроби знайти вразливість в криптографічному алгоритмі або протоколі.

Спочатку методи криптоаналізу ґрунтувалися на лінгвістичних закономірностях тексту й реалізовувалися з використанням тільки олівця і паперу. Але з часом їм на зміну приходять математичні методи, для реалізації яких використовуються спеціалізовані криптоаналітичні комп'ютери. І якщо в недалекому минулому криптоаналітиками були переважно лінгвісти, то зараз — це «чисті» математики.

Оволодіння методами криптоаналізу неможливе без знань теорії ймовірностей та математичної статистики, лінійної алгебри, комбінаторики, теорії графів.

Брюс Шнаєр виділяє 4 основних і 3 додаткових методи криптоаналізу, припускаючи знання криптоаналітиків алгоритму шифру. До основних методів криптоаналізу відносять: атаку на основі шіфротекста, атаку на основі відкритих текстів та відповідних шифротекстів, атаку на основі підібраного відкритого тексту (можливість вибрати текст для шифрування), атаку на основі адаптивно підібраного відкритого тексту. До додаткових методів криптоаналізу відносять: атаку на основі пі-

дібраного шіфротекста, атаку на основі підібраного ключа, бандитський криптоаналіз.
Ключові слова: *чисельні методи розв'язання алгебраїчних рівнянь, метод січних, метод Ньютона, метод Ліна, метод інтерполяції, криптоаналіз.*

Abstract. *In paper, there are separate numbers of methods for algebraic equationss, and also a new numerical method for combining algebraic ones, which is a basis for multipliers, is introduced. This method allows to select groups of multiple and related multiple roots of the equation. In same cases this bads to well known methods: to the method of sieve, to the method of Newton, to the method of Lin, to the method of interpolation. Note that this method can be fixed for the development of algebraic derivations. For a similar purpose, it is necessary to direct the activities of a method of development of a common field of activity to a single one. For all methods examples of the program, written on C ++ in the packages of computer mathematics MAPLE.*

Cryptanalysis is the science of how to get the original value of encrypted information without access to a secret key. The main purpose of cryptanalysis is to find the key. For the first time the term "cryptanalysis" was introduced by the American cryptographer William F. Friedman in 1920. The term "cryptanalysis" also means attempts to find a vulnerability in a cryptographic algorithm or protocol.

Initially, the methods of cryptanalysis were based on the linguistic regularities of the text and implemented using only pencil and paper. But over time, they are replaced by mathematical methods, which are implemented by specialized cryptanalytical computers. And if in the recent past crypto-analysts were mostly linguists, now they are pure mathematics. Mastering the methods of cryptanalysis is impossible without knowledge of probability theory and mathematical statistics, linear algebra, combinatorics, graph theory.

Bruce Schneier identifies 4 basic and 3 additional methods of cryptanalysis, assuming knowledge of cryptanalysts of the cipher algorithm. The main methods of cryptanalysis include: ciphertext attack, plaintext attack and related ciphertext attack, selected plaintext attack (ability to select text for encryption), adaptive plaintext attack. Additional methods of cryptanalysis include: a ciphertext-based attack, a key-based attack, bandit cryptanalysis
Key words: *numerical methods for algebraic derivations, classical methods, Newton method, Lin method, interpolation method, cryptanalysis.*

Актуальність проблеми. Криптоаналіз — це наука про методи отримання вихідного значення зашифрованої інформації без доступу до секретного ключа. Головна мета криптоанализа — знаходження ключа. Вперше термін «криптоаналіз» ввів американський криптограф Вільям Ф. Фрідман в 1920 році. Під терміном «криптоаналіз» також розуміють спроби знайти вразливість у криптографічному алгоритмі або протоколі. Спочатку методи криптоаналізу ґрунтувалися на лінгвістичних закономірностях тексту й реалізовувалися з використанням тільки олівця і паперу. Але з часом їм на зміну приходять математичні методи, для реалізації яких використовуються спеціалізовані криптоаналітичні комп'ютери.

Метою роботи є дослідження математичного інструментарію і методів комп'ютерної математики з метою їх застосування в

криптоаналізі, а також запропоновано новий чисельний метод розв'язання алгебраїчних рівнянь, який базується на розкладі многочлена на множники.

Викладення основного матеріалу дослідження. Розглянемо актуальні універсальні методи розв'язання рівняння $f(x)=0$, які ілюструються програмами, що реалізують ці методи. Метод поділу відрізка навпіл (метод дихотомії) — цей метод є одним з найпростіших і найнадійніших методів розв'язання рівняння $f(x)=0$, де функція $y=f(x)$ неперервна на відрізку[*a; b*]. Наведемо алгоритм Коші.

1. Відрізок [*a*; *b*] обирається таким чином, щоб $f(a)f(b)\le 0$. Якщо $f(a)f(b)>0$, то необхідно змінити значення *a* та *b* так, щоб відрізок містив корінь рівняння.

2. Знаходимо середину відрізка $c=\frac{a+b}{2}$ та обчислюємо значення $f(c)$ Якщо $f(c)\neq 0$, то якщо $f(a)f(b)>0,\quad a=c$, у протилежному випадку $b=c$.

3. Поділ відрізка відбувається доти, доки не буде виконана нерівність $|b-a|<\varepsilon$, де ε — задана точність обчислень.

Можна заздалегідь визначити кількість ітерацій, необхідних для досягнення заданої точності

$$n>\frac{\ln\left(\frac{b-a}{\varepsilon}\right)}{\ln 2}.$$

Метод дихотомії є стійким до помилок округлення, але він відрізняється повільною збіжністю та зазвичай використовується для відшукання початкового наближення для інших швидших методів знаходження коренів рівняння.

Наведемо приклад програми, що реалізує метод половинного поділу.

Приклад. Розв'язати рівняння

$$\sin x=0.$$

Програма 1

Метод половинного поділу відрізка, що містить корінь

```
prog1 := proc(a, b)
 local a1, b1, c;
 a1 := a;
 b1 := b;
 if evalf(sin(a)·sin(b)) > 0 then print('sin(a)·sin(b) > 0');
 else
 while evalf(|b1 − a1|) ≥ 0.000002 do
 c := (a1 + b1)/2;
 if evalf(sin(c)·sin(b1)) > 0 then b1 := c end if;
 if evalf(sin(a1)·sin(c)) > 0 or evalf(sin(c)) = 0 then a1 := c end
     if
 end do;
 print( evalf(a1), evalf(b1), evalf( (a1 + b1)/2 ),
     evalf( sin( (a1 + b1)/2 ) ) );
 end if;
 end proc ;
```

```
prog1 := proc(a, b)
    local a1, b1, c;
    a1 := a;
    b1 := b;
    if 0 < evalf(sin(a)*sin(b)) then
        print('0 < sin(a)*sin(b)')
    else
        while 0.000002<=evalf(abs(b1 − a1)) do
            c := 1/2*a1 + 1/2*b1;
            if 0 < evalf(sin(c)*sin(b1)) then b1 := c end if;
            if 0 < evalf(sin(a1)*sin(c)) or evalf(sin(c)) = 0 then
                a1 := c
            end if
        end do;
        print(evalf(a1), evalf(b1), evalf(1/2*a1 + 1/2*b1),
        evalf(sin(1/2*a1 + 1/2*b1)))
    end if
end proc
```

```
>prog1(−1, 2)
```

$$-9.53674316410^{-7}, 4.76837158210^{-7}, -2.38418579110^{-7},$$
$$-2.38418579110^{-7}$$

```
>prog1(3, 4)
```

$$3.141592026, 3.141593933, 3.141592979, -3.25410206810^{-7}$$

```
>prog1(0.5, 0.6)
```

$$0 < \sin(0.5)\ \sin(0.6)$$

Іноді знаходження відрізка [*a; b*], що містить один простий корінь рівняння $f(x) = 0$, є само по собі складною задачею. На основі попередньої програми складена програма 2 для відшукання кількох коренів, розташованих на відрізку [*a; b*] для неперервної функції $y = f(x)$. Для цього послідовно розглядаються відрізки $[a + kh, a + kh + h]$, де h — крок дискретизації. Якщо на кінцях відрізка функція приймає значення різних знаків, то цей відрізок приймається за відрізок [*a; b*], і на ньому обчислюється значення кореня рівняння.

Приклад. Знайти дійсні корені рівняння

$$x^4 - 4x^3 - 19x^2 + 106x - 120 = 0$$

на відрізку [–10; 10], з кроком дискретизації $h = 0{,}1$.

Програма 2
Відшукання коренів рівняння на відрізку [*a*; *b*]

```
fny := proc(y)
 local z;
 z := y^4 - 4·y^3 - 19·y^2 + 106·y - 120;
> end proc : z

    prog2 := proc(a, b, h)
     local a1, b1, c, a2, b2;
     a2 := a;
     b2 := b;
     h2 := h;
     for i from 0 by 1 while evalf( a2 + i·h2) ≤ evalf (b2) do
     if evalf (fny(a2 + i·h2)·fny(a2 + i·h2 + h2)) ≤ 0 then
     a1 := a2 + i·h2;
     b1 := a2 + i·h2 + h2;

     while evalf (|b1 - a1|) ≥ 0.000002 do
     c := (a1 + b1)/2;
     if evalf (fny(c)·fny(b1)) > 0 then b1 := c end if;
     if evalf (fny(a1)·fny(c)) > 0 or evalf (fny(c)) = 0 then a1 := c
         end if;
    end do;
    print( evalf (a1), evalf (b1), evalf ( (a1 + b1)/2 ),
        evalf ( fny( (a1 + b1)/2 ) ) );
     end if;
     end do;
     end proc;
>
```

Warning, `h2` is implicitly declared local to procedure `prog2`

Warning, `i` is implicitly declared local to procedure `prog2`

```
prog2 := proc(a, b, h)
    local a1, b1, c, a2, b2, h2, i;
    a2 := a;
    b2 := b;
    h2 := h;
    for i from 0 while evalf(a2 + i*h2) <= evalf(b2) do
        if evalf(fny(a2 + i*h2)*fny(a2 + i*h2 + h2)) <= 0
        then
            a1 := a2 + i*h2;
            b1 := a2 + i*h2 + h2;
            while 0.000002 <= evalf(abs(b1 - a1)) do
                c := 1/2*a1 + 1/2*b1;
                if 0 < evalf(fny(c)*fny(b1)) then b1 := c end if;
                if 0 < evalf(fny(a1)*fny(c)) or evalf(fny(c))
                 = 0 then
                    a1 := c
                end if
            end do;
            print(evalf(a1), evalf(b1), evalf(1/2*a1 + 1/2*b1),
            evalf(fny(1/2*a1 + 1/2*b1)))
        end if
    end do
end proc
```

```
> prog2(-10, 10, 0.1)
    [-5.000001526, -5.0, -5.000000763, 0.000384]
    [-5.0, -4.999998474, -4.999999237, -0.000384]
    [1.999998474, 2.0, 1.999999237, -0.000010]
    [2.0, 2.000001526, 2.000000763, 0.000010]
    [2.999998474, 3.0, 2.999999237, 0.000006]
    [3.0, 3.000001526, 3.000000763, -0.000006]
    [3.999998474, 4.0, 3.999999237, -0.000013]
    [4.0, 4.000001526, 4.000000763, 0.000013]
```

При використанні програми припускаємо, що $a < b$ та $h > 0$. Функція, що розглядається може не існувати в окремих точках або на деяких інтервалах.

Для розв'язування складніших рівнянь, що містять ірраціональні вирази, складена програма 3 для дослідження функцій. На відрізку $[a; b]$ з кроком $h > 0$ обчислюються значення функції $f(a + kh)$.

Якщо $|f(a + kh)| \leq \varepsilon$, то на друк виводиться знак 0. Величина ε обирається досить малою. Якщо $|f(a + kh)| > \varepsilon$, то на друк виводиться знак +.Якщо $f(a + kh) < -\varepsilon$, то виводимо знак —. Значення k=0, 1, 2,… змінюються та друкуються лише ті значення $x = a + kh$, при яких знаки функцій $f(a + kh)$ та $f(a + kh + h)$ є різними. Також друкуються знаки функцій $(a + kh)$ та $f(a + kh + h)$.

Приклад. Знайти дійсні корені рівняння

$$x^4 - 4x^3 - 19x^2 + 106x - 120 = 0 .$$

Програма 3
Дослідження рівнянь методом інтервалів

```
> fny := proc(y) z := y^4 - 4*y^3 - 19*y^2 + 106*y - 120; end proc : z
```

Warning, `z` is implicitly declared local to procedure `fny`

```
> a1 := -10; b1 := 10; h1 := 0.1;
```

$$a1 := -10$$

$$b1 := 10$$

$$h1 := 0.1$$

a1 — ліва границя;
b1 — права границя;
h1 — крок;

```
x := a1; m := 2;
for i from 0 by 1 while evalf(x) <= evalf(b1) do
 x := a1 + i*h1 :
 y := evalf(fny(x)) :
 if evalf(|y|) <= 0.0000001 then l := 0; end if;
 if evalf(y - h1) > 0.0000001 then l := 1; end if;
 if evalf(y + h1) < -0.0000001 then l := -1; end if;
 if evalf(l) <> evalf(m) then print(m, "---------->", l, "x= ", x)
    end if;
 m := l :
end do:
```

$$x := -10$$

$$m := 2$$

2, "---------->", 1, "x= ", −10

1, "---------->", 0, "x= ", −5.0

0, "---------->", −1, "x= ", −4.9

−1, "---------->", 0, "x= ", 2.0

0, "---------->", 1, "x= ", 2.1

1, "---------->", 0, "x= ", 3.0

0, "---------->", −1, "x= ", 3.1

−1, "---------->", 0, "x= ", 4.0

0, "---------->", 1, "x= ", 4.1

Наведені результати вказують на те, що при $-10 < x < -5, f(x) > 0$, при $-5 < x < 2, f(x) < 0$, при $2 < x < 3, f(x) > 0$, при $3 < x < 4, f(x) < 0$, при $44 < x < 10, f(x) > 0.$

У програмі 4 для дослідження рівняння $f(x) = 0$ повторюється ідея програми 3, але при цьому уточнюється інтервал $[a; b]$, на кінцях якого функція $f(x)$ приймає різні значення.

Приклад. Дослідити рівняння

$$\sqrt{x-3-2\sqrt{x-4}} + \sqrt{x-4\sqrt{x-4}} - 1 = 0.$$

Програма 4
Дослідження рівнянь методом інтервалів

```
fny :=proc(x)
z := sqrt(x - 3 - 2*sqrt(x - 4)) + sqrt(x - 4*sqrt(x - 4)) - 1;
end proc : z
>
```

Warning, `z` is implicitly declared local to procedure `fny`*z*

```
d1 := 4.5;
d2 := 8.5;
h := 0.1;

eps := 0.000005;
eps2 := 0.0002;
m := 2;

>
```

$$d1 := 4.5$$

$d2 := 8.5$

$h := 0.1$

$eps := 5.\ 10^{-6}$

$eps2 := 0.0002$

$m := 2$

$d1$ — ліва границя;

$d2$ — права границя;

h — крок;

for i **from** 0 **by** 1 **while** $evalf(d1 + i \cdot h) \le evalf(d2)$ **do**

if $evalf(|fny(d1 + i \cdot h)|) \le evalf(eps)$ **then** $l := 0$; **end if**;

if $evalf(fny(d1 + i \cdot h)) > evalf(eps)$ **then** $l := 1$; **end if**;

if $evalf(fny(d1 + i \cdot h)) < evalf(-eps)$ **then** $l := -1$; **end if**;

if $evalf(l) \neq evalf(m)$ **then**

$a := d1 + i \cdot h - h$;

$b := d1 + i \cdot h$;

while $evalf(|b - a|) \ge evalf(eps2)$ **do**

$c := \dfrac{(a + b)}{2}$;

if $evalf(fny(c) \cdot fny(b)) > 0$ **then** $b := c$; **end if**;

if $evalf(fny(a) \cdot fny(c)) > 0$ **or** $evalf(fny(c)) = 0$ **then** $a := c$; **end if**;

end do;

$print\left(m, \text{"----------->"}, l, \text{"x= "}, \dfrac{a + b}{2}\right)$;

end if;

$m := l$:

end do:

```
>
```

```
2, "----------->", 1, "x= ", 4.450000000
1, "----------->", 0, "x= ", 4.99990234
0, "----------->", 1, "x= ", 8.00009765
```

При цьому знаходимо, що $f(x) = 0$ при $5 \le x \le 8$.

Очевидно, що наведені програми 3 та 4 можна застосовувати для розв'язання нерівностей.

Метод січних (метод хорд). Основна ідея розв'язання рівняння $f(x) = 0$, де функція $y = f(x)$ — достатньо гладка, полягає в тому, що функція $y = f(x)$ замінюється в околі кореня простішою функцією $y = g(x)$. Потім розв'язується рівняння $f(x) = 0$. Якщо

функція $y = g(x)$ — лінійна, т.т. рівняння $f(x) = 0$ заміняється рівнянням

$$g(x) = ax + b = 0,$$

і ми отримуємо метод січних.

В околі кореня рівняння $f(x) = 0$ обираємо довільно точки A й B та обчислюємо значення функції в цих точках: $y_1 = f(A), y_2 = f(B)$. Через точки $(A; y_1)$ і $(B; y_2)$ на площині (x, y) проводимо пряму

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - A}{B - A}.$$

При $y = 0$ знаходимо наближене значення кореня

$$x_1 = \frac{Ay_2 - By_1}{y_2 - y_1} = \frac{Af(B) - Bf(A)}{f(B) - f(A)}. \qquad (1)$$

Для перевірки правильності обчислення кореня знаходимо значення $y_3 = f(x_1)$.

Нехай ε — задане число, що задає точність обчислення кореня. Якщо $|f(x_1)| \geq \varepsilon$, то повторюємо обчислення, поклавши $A = x_1$ та зберігаючи значення B.

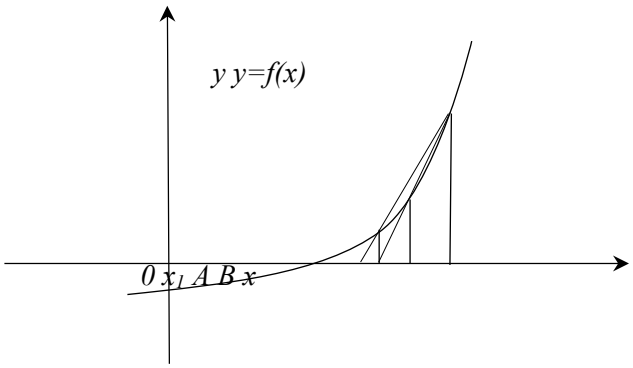

Рис. 1

Якщо $|f(x_1)| < \varepsilon$, то роздруковуємо значення кореня рівняння x_1.

Приклад. Розв'язати методом січних рівняння

$$sinx = 0$$

Програма 5

Метод січних

```
>eps := 0.0000002, f := 0.1;
```

$$eps := 2.\ 10^{-7}$$

$$f := 0.1$$

```
>fny := proc(a) z := evalf(sin(a)); end proc : z
```

Warning, `z` is implicitly declared local to procedure `fny`

$$z$$

```
>a := 0.2; b := 0.1;
```

$$a := 0.2$$

$$b := 0.1$$

```
>
```

```
while f ≥ eps do
x1 := (a·fny(b) − b·fny(a))/(fny(b) − fny(a));
f := abs(fny(x1));
a := x1;
end do:
print("x1=", evalf(x1), "f(x)=", evalf(fny(x1)));
```

"x1=", -2.78166716610^{-9}, "f(x)=", -2.78166716610^{-9}

Метод Ньютона-Рафсона (метод Ньютона) полягає в заміні функції $y = f(x)$ рівнянням дотичної до функції в околі кореня

$$y - f(A) = f'(A)(x - A)\ ,$$

де A — точка, що розташована близько до кореня рівняння $f(x) = 0$. Похідна наближено обчислюється за формулою

$$f'(A) = \frac{f(A + \Delta) - f(A - \Delta)}{2\Delta}.$$

При $y = 0$ знаходимо уточнене значення кореня

$$x = A - \frac{f(A)}{f'(A)}. \qquad (2)$$

Якщо $|f(x)| < \varepsilon$, то роздруковуємо значення кореня. Якщо $|f(x)| \geq \varepsilon$, то повторюємо обчислення, поклавши $A = x$.

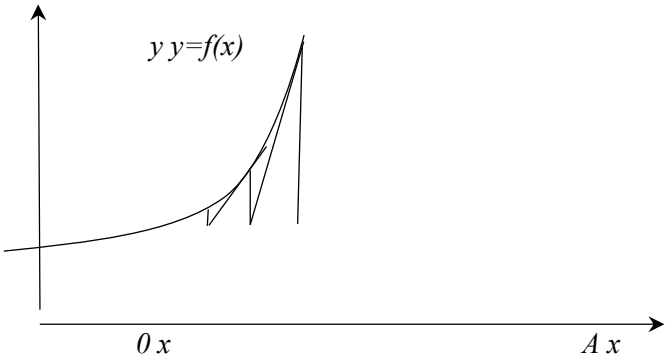

Рис. 2

Приклад. Розв’язати методом Ньютона рівняння

$$sinx = 0$$

Програма 6
Метод Ньютона

```
>eps := 0.0000002, f := 1;
                    eps := 2. 10^-7
                    f := 1
>fny := proc(a) z := evalf(sin(a)); end proc : z
Warning, `z` is implicitly declared local to procedure `fny`
                    z
>a := 1;
                    a := 1
while f ≥ eps do
d := (fny(a + 0.001) − fny(a − 0.001)) · 500;
x1 := a − fny(a)/d;
f := abs(fny(x1));
a := x1;
end do:
print("x1=", evalf(x1), "f(x)=", evalf(fny(x1)));
>
        "x1=", 1.622 10^-11, "f(x)=", 1.62200000010^-11
```

Обчислювальна проблема кратних коренів. Відомо, що корені рівняння

$$f(z) \equiv z^n + a_1 z^{n-1} + \ldots + a_n = 0 \qquad (3)$$

є неперервними функціями коефіцієнтів рівняння. Якщо корінь z рівняння (3) простий, то він є голоморфною функцією коефіцієнтів. Зазвичай коефіцієнти рівняння (3) відомі лише наближено з деякою точністю, тобто фактично розв'язується рівняння (3),

$$\text{де} \quad a_k = a_{k0} + \delta_k, \qquad (k = 1,\ldots,\ldots n) \qquad (4)$$

й відомі значення a_{k0} та оцінки ε_k для δ_k

$$| \delta_k | < \varepsilon_k, \qquad (k = 1,\ldots,\ldots n).$$

Якщо при деяких значеннях $a_k \ (k = 1,\ldots,\ldots n)$, що задовольняють умови

$$| a_k - a_{k0} | < \varepsilon_k, \qquad (k = 1,\ldots,\ldots n) \qquad (5)$$

рівняння (3) має кратні корені, то будемо казати, що рівняння

$$f_0(z) \equiv z^n + a_{10} z^{n-1} + \ldots + a_{n0} = 0 \qquad (6)$$

має близькі до кратних корені.

Якщо рівняння (6) має кратні або близькі до кратних корені, то виникають обчислювальні труднощі при відшуканні коренів.

Нехай рівняння (6) має простий корінь z_0.

З рівняння (3) знаходимо наближений вираз для кореня z близького до z_0

$$z - z_0 = -\frac{1}{f'(z_0)} \sum_{k=1}^{n} z_0^{k-1} (a_k - a_{k0}) + \ldots, \qquad (7)$$

де … позначають нескінченно малі другого порядку відносно $\delta_k = a_k - a_{k0}, \quad (k = 1,\ldots,\ldots n)$.

З формули (7) слідує наближена нерівність

$$| z - z_0 | \le \frac{1}{|f'(z_0)|} \sum_{k=1}^{n} |z_0^{k-1}| |\delta_k|,$$

з якої випливає, що малі обчислювальні похибки δ_k при відшуканні коефіцієнтів a_k $(k=1,\ldots,\ldots n)$ призводять до малих обчислювальних похибок при відшуканні простого кореня рівняння (3).

Якщо z_0 — кратний корінь рівняння (5) або близький до кратного, то величина $|f'(z_0)|$ є малою та малі похибки при відшуканні коефіцієнтів a_k $(k=1,\ldots,\ldots n)$ призводять до порівняно великим похибкам при відшуканні коренів рівняння (3).

Розглянемо алгебраїчне рівняння

$$z^n = \alpha, \tag{8}$$

де α — нескінченно мала величина. При $\alpha \to 0$ рівняння (8) має корінь кратності n $z=0$. З рівняння (8) знаходимо корені

$$z = \sqrt[n]{\alpha},$$

які є не диференційованими функціями α та мають критичну точку $\alpha=0$. Наприклад, рівняння $z^6=0,$ має кратний корінь $z=0$, а рівняння $z^6=0{,}000001$ має корені

$$z_k = 0{,}1\left(\cos\frac{\pi k}{3} + i\sin\frac{\pi k}{3}\right) \qquad (k=0,1,\ldots,5).$$

Таким чином, похибка в шостому знаку після коми в коефіцієнті α обумовлює похибку в першому знаку в коренях z_k, оскільки $|z_k|=0{,}1$. Окрім цього самі чисельні методи, орієнтовані на відшукання окремого кореня призводять до великих обчислювальних похибок при знаходженні близьких коренів.

Зазначимо ще на одну можливу причину виникнення обчислювальних похибок.

При відшуканні коренів многочлена $f_n(z)$ після відшукання кореня z_1 зазвичай ступень многочлена $f_n(z)$ знижується на одиницю. Для цього знаходимо многочлен ступеня n-1

$$f_{n-1}(z) = \frac{f_n(z)}{z - z_1},$$

а потім шукаємо корінь z_2 многочлена $f_{n-1}(z)$ й т.д. Якщо корінь z_1 знайдено з великою обчислювальною похибкою, то й многочлен $f_{n-1}(z)$ також знаходиться з великою похибкою.

Щоб уникнути накопичення великих похибок можна не знижувати ступень многочлена $f_n(z)$, а кожного разу шукати корені вихідного рівняння $f_n(z)=0$. Для цього потрібно запам'ятовувати знайдені раніше корені й порівнювати зі знову знайденим коренем. Другий спосіб, який пропонується в даній роботі, полягає у відшуканні рівняння q-ого ступеня, яке має усі близькі один до одного корені вихідного рівняння. При цьому обчислювальна похибка не зростає і можна понизити ступень вихідного многочлена на q одиниць.

Розв'язання квадратного рівняння з комплексними коефіцієнтами. Якщо алгебраїчне рівняння другого порядку

$$a_0 z^2 + a_1 z + a_2 = 0 \quad (a_0 \neq 0) \tag{9}$$

має дійсні коефіцієнти $a_0, a_1, a_2 = 0$, то розв'язки рівняння знаходяться за відомими формулами

$$z_{1,2} = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0 a_2}}{2a_0}. \tag{10}$$

Якщо дискримінант рівняння $D = a_1^2 - 4a_0 a_2 \geq 0,$ то розв'язки (9) дійсні, якщо $D < 0,$ то розв'язки рівняння (9) — комплексні.

Приклад. Розв'язати рівняння

$$z^2 + 2z + 5 = 0.$$

За формулою (10) знаходимо $z_{1,2} = -1 \pm 2i.$

Для використання формули (10) в загальному випадку, коли коефіцієнти рівняння є комплексними числами, необхідно вказати спосіб добування квадратного кореня з комплексного числа $a + ib.$ Нехай

$$\sqrt{a + ib} = x + iy. \tag{11}$$

Піднесемо цю рівність до квадрата, отримаємо вираз

$$a + ib = x^2 - y^2 + 2ixy,$$

з якого випливає система рівнянь

$$x^2 - y^2 = a, \qquad 2xy = b.$$

Підносимо ці рівняння до квадрата та додаємо. Отримаємо такий вираз

$$x^4 + 2x^2y^2 + y^4 = a^2 + b^2 \quad \Rightarrow \quad x^2 + y^2 = \sqrt{a^2 + b^2}.$$

Остаточно отримаємо вирази для x та y

$$x = \pm\sqrt{\frac{\sqrt{a^2 + b^2} + a}{2}}, \qquad y = \pm signb\sqrt{\frac{\sqrt{a^2 + b^2} - a}{2}}. \qquad (12)$$

Отримані формули є основою програми для розв'язання квадратного рівняння з комплексними коефіцієнтами

$$\big(A(0) + iB(0)\big)z^2 + \big(A(1) + iB(1)\big)z + \big(A(2) + iB(2)\big) = 0.$$

Програма 7
Розв'язання квадратного рівняння з комплексними коефіцієнтами

```
>a := Array(0..2, [1, 1, -5]); b := Array(0..2, [0, 0, -5· I]);
                  a := Array(0..2, {0 = 1, 1 = 1, 2 = -5})
                      b := Array(0..2, {2 = -5 I})
c := 2·sqrt(a[0]^2 + b[0]^2) :
a1 := a[1]^2 - b[1]^2 - 4·a[0]·a[2] + 4·b[0]·b[2] :
b1 := 2·a[1]·b[1] - 4·b[0]·a[2] - 4·a[0]·b[2] :
p := sqrt(a1^2 + b1^2) :
x1 := sqrt((p + a1)/2) :
if evalf(b1) < 0 then y1 := -sqrt((p - a1)/2) : else y1
    := sqrt((p - a1)/2) : end if:
c11 := -a[1] + x1 : d11 := -b[1] + y1 :
c12 := -a[1] - x1 : d12 := -b[1] - y1 :
x1 := (c11·a[0] - d11·b[0])/c : y1 := (c11·b[0] + d11·a[0])/c :
x2 := (c12·a[0] - d12·b[0])/c : y2 := (c12·b[0] + d12·a[0])/c :
print("x1= ", evalf(x1), "+j*", evalf(y1));
print("x2= ", evalf(x2), "+j*", evalf(y2));
>
```

```
"x1= ", 1.350781059, "+j*", 1.350781059 I
"x2= ", -2.350781059, "+j*", -1.350781059 I
```

Приклад. Розв'язати рівняння

$$z^2 + z - 5 - 5i = 0.$$

Задавши степінь рівняння (2) та значення коефіцієнтів 1, 1,-5-5i, отримаємо відповідь

$$x_1 = 2 + i; \qquad x_2 = -3 - i.$$

Приклад. Розв'язати рівняння

$$(-5 - 5i)z^2 + z + 1 = 0.$$

Задавши степінь рівняння (2) та значення коефіцієнтів -5-5*i*, 1, 1, отримаємо відповідь

$$x_1 = 0{,}4 - 0{,}2i; \qquad x_2 = -0{,}3 + 0{,}1i.$$

Наведемо також програму для добування кореня квадратного з комплексного числа.

Програма 8
Добування кореня квадратного з комплексного числа $z = a + ib$

```
>restart;
a –дійсна частина
b — уявна частина
>a := 2 : b := 1 :
      p := |a| + |b| :
      a1 := a/p :
      b1 := b/p :
      l := sqrt(p) :
      q := sqrt(a1^2 + b1^2) :
      x := sqrt((q + a1)/2) · l :
      y := sqrt((q - a1)/2) · l :
      if evalf(x · y · b1) < 0 then y := -y end if:
>   print("sqr(z)=+-(", evalf(x), "+j*", evalf(y), ")");
                "sqr(z)=+-(", 1.455346691, "+j*", 0.3435607492, ")"
```

Розв'язання алгебраїчного рівняння

$$z^N = a + ib$$

представимо у вигляді

$$z = \sqrt[N]{a + ib} = \sqrt[n]{r}\left(\cos\frac{\varphi + 2\pi k}{N} + i\sin\frac{\varphi + 2\pi k}{N}\right), \qquad (k = 0, 1, ..., N-1)$$

$$r=\sqrt{a^2+b^2},\qquad a=r\cos\varphi,\qquad b=r\sin\varphi. \tag{13}$$

Наведемо програму для добування кореня N-го ступеня з комплексного числа $z=a+ib$.

Програма 9
Добування кореня *N*-го ступеня
з комплексного числа $z=a+ib$

```
>restart;
```

a –дійсна частина
b –уявна частина
N –показник ступеня

```
>a := 0 : b := 1 : n := 3 :
  if evalf(a) = 0 then
    if evalf(b) > 0 then f := π/2 : else f := 3·π/2 : end if:
  else
    if evalf(a) > 0 then f := arctan(b/a) : else f := π + arctan(b/a)
      end if:
  end if:
  r := exp((0.5/n)·ln(a^2 + b^2)) :
  for i from 0 to evalf(n − 1) do
  arg := (f + 2·π·i)/n :
  print("z^(1/n)", evalf(r*cos(arg)), "+j·", evalf(r*sin(arg))) ;
  end do:
>
```

```
"z^(1/n)", 0.8660254040, "+j*", 0.500000000(
"z^(1/n)", −0.8660254040, "+j*", 0.500000000(
"z^(1/n)", 0., "+j*", −1.
```

Метод Ньютона є достатньо ефективним для розв'язання алгебраїчних рівнянь N-го ступеня

$$f(z)\equiv\sum_{k=0}^{N}\bigl(a(k)+ib(k)\bigr)z^{N-k}=0, \tag{14}$$

де $a(k),\ b(k)\quad (k=0,\ldots,N)$ — дійсні числа.

Опишемо обчислювальний алгоритм:

1. Задати N і коефіцієнти $a(k),\ b(k)\quad (k = 0,\ldots, N)$. Якщо коефіцієнти рівняння дійсні, то можна вводити лише дійсні частини коефіцієнтів $a(k), (k = 0,\ldots, N)$.

2. Обчислити радіус R круга $|z| \le R,$ що містить усі крені многочлена (14) за формулою

$$R = 1 + \frac{1}{|a(0) + ib(0)|} \sum_{k=1}^{N} |a(k) + ib(k)|. \qquad (15)$$

3. Випадковим чином обираємо початкову точку $z_0 = x_0 + iy_0,$ яка рівномірно розподілена в прямокутнику $|x_0| \le R, |y_0| \le R.$

4. Утворюємо послідовність комплексних чисел

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)} \qquad (n = 0, 1, 2,\ldots). \qquad (16)$$

5. Якщо послідовність $z_n \quad (n = 0, 1, 2,\ldots)$ збігається до деякого значення z_∞, то z_∞ є коренем рівняння $f(z) = 0$ і многочлен $f(z)$ ділиться без остачі на $z - z_\infty$. Оскільки значення многочлена $f(z_n)$ обчислюється по схемі Горнера, то одразу обчислюється значення многочлена

$$f_1(z) \equiv \frac{f(z)}{z - z_\infty}$$

ступеня N-1. Потім шукаємо корінь многочлена $f_1(z)$ і т.д.

6. Якщо послідовність z_n не збігається, то обираємо нове початкове наближення та всі обчислення повторюються.

Метод інтерполяції полягає в заміні загального рівняння $f(z) = 0$ простішим рівнянням $g(z) = 0$, де $g(z)$ — інтерполяційний многочлен для функції $f(z)$. Зазвичай степінь многочлена $g(z)$ менша степені N многочлена $f(z)$.

Нехай задано точки комплексної площини $z_1, z_2,\ldots, z_q$. Якщо виконуються рівності

$$f(z_k) = g(z_k) \qquad (k = 1,\ldots, q;\ q \le N),$$

то ми можемо знайти многочлен $g(z)$ степені q-1 за допомогою формули Лагранжа

$$g(z)=\sum_{k=1}^{q} f(z_k)\frac{(z-z_1)\ldots(z-z_{k-1})(z-z_{k+1})\ldots(z-z_q)}{(z_k-z_1)\ldots(z_k-z_{k-1})(z_k-z_{k+1})\ldots(z_k-z_q)}. \quad (17)$$

Але використання цієї формули у випадку, коли точки $z_1, z_2, \ldots, z_q$ розташовані близько одна до іншої, може привести до великих обчислювальних похибок, оскільки знаменники дробів у формулі (17) прямують до нуля. Для многочленів можна використовувати інший простіший та ефективніший метод інтерполяції.

Поділимо многочлен $f(z)$ на многочлен

$$d(z)=(z-z_1)(z-z_2)\ldots(z-z_q). \quad (18)$$

При цьому дістанемо рівність

$$\frac{f(z)}{d(z)}=\varphi(z)+\frac{g(z)}{d(z)}, \quad (19)$$

де $\varphi(z)$ – многочлен степені (N-q), $g(z)$ — остача від ділення — многочлен степені (q-1).

Маємо рівність

$$f(z)=\varphi(z)d(z)+g(z), \quad (20)$$

яка в силу представлення (17) приводить до системи рівностей (16). Отже, остача $g(z)$ від ділення многочлена $f(z)$ на $d(z)$ є інтерполяційним многочленом для $f(z)$ з вузлами інтерполяції $z_1, z_2, \ldots, z_q$.

Щоб знайти простий корінь $z=z_0$ рівняння $f(z)=0$, можна поділити многочлен $f(z)$ на дільник $d_n(z)=(z-\alpha_n)(z-\beta)$, де числа α_n, β — досить близькі до z_0. Прирівнюючи остачу від ділення до нуля, знаходимо уточнене значення кореня α_{n+1} та потім ділимо $f(z)$ на дільник

$$d_{n+1}(z)=(z-\alpha_{n+1})(z-\beta).$$

При цьому метод інтерполяції співпадає з методом хорд, якщо коефіцієнти многочлена $f(z)$ дійсні й числа z_0, α_n, β — дійсні.

Аналогічно, виконуючи ділення многочлена $f(z)$ на дільник $d_n(z) = (z - \alpha_n)^2$ та прирівнюючи до нуля лінійну остачу від ділення, приходимо до методу Ньютона.

Дійсно, з рівності

$$f(z) = \varphi(z)(z - \alpha_n)^2 + g(z), \qquad g(z) = bz + c \tag{21}$$

знаходимо

$$f(\alpha_n) = g(\alpha_n), \qquad f'(\alpha_n) = g'(\alpha_n).$$

При цьому знаходимо лінійну остачу

$$g(z) \equiv f'(\alpha_n)z + (f(\alpha_n) - f(\alpha_n)\alpha_n)$$

та уточнене значення кореня α_{n+1} з рівняння $g(\alpha_{n+1}) = 0$

$$\alpha_{n+1} = \alpha_n - \frac{f(\alpha_n)}{f'(\alpha_{n+1})}. \tag{22}$$

Отже, метод лінійної інтерполяції в частинному випадку приводить до методу Ньютона.

Метод Ліна або метод передостанньої остачі полягає в такому. Многочлен $f(z)$ степені *n* ділиться на довільно обраний многочлен $d_0(z)$ степені $q\,(q < n)$. Передостання остача від ділення є многочлена степені *q*. Після зведення, тобто ділення многочлена на коефіцієнт при старшому степені *z*, цю остачу позначим через $d_1(z)$. Потім ділимо многочлен $f(z)$ на многочлен $d_1(z)$ та передостанню остачу після зведення позначимо через $d_2(z)$ і т.д. При цьому утворюється послідовність дільників $d_k(z)$ $(k = 0, 1, 2, \ldots)$ степені *q*. Якщо послідовність $d_k(z)$ $(k = 0, 1, 2, \ldots)$ збігається до деякого многочлена $d(z)$, то $d(z)$ є дільником многочлена $f(z)$.

Схему метода Ліна можна представити у вигляді

$$\begin{aligned} f(z) &= \varphi_1(z)d_0(z) + b_1 d_1(z), \\ f(z) &= \varphi_2(z)d_1(z) + b_2 d_2(z), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ f(z) &= \varphi_k(z)d_{k-1}(z) + b_k d_k(z), \quad b_k = const \end{aligned} \tag{23}$$

де $d_k(z)$ — зведені многочлени степені *q*.

Якщо послідовність $d_k(z)$ збігається при $k \to \infty$ до многочлена $d(z)$, то отримаємо розклад $f(z)$ на множники

$$f(z) = (\varphi(z) + b)d(z), \tag{24}$$

де $\varphi(z) = \lim\limits_{k\to\infty} \varphi_k(z)$, $\quad b = \lim\limits_{k\to\infty} b_k$, $\quad d(z) = \lim\limits_{k\to\infty} d_k(z)$.

Збіжність методу Ліна не доведена, але в деяких роботах цей метод рекомендується для відшукання квадратичних множників, які відповідають комплексно-спряженим кореням. Ці рекомендації виявилися дещо неточними.

Метод Ліна можна викласти таким чином. А саме, розглянемо довільно зведений многочлен $d_0(z)$ степені q та поділимо вихідний многочлен $f(z)$ на многочлен $zd_0(z)$ степені q+1. Остача від ділення в загальному випадку є многочленом степені q та співпадає, це випливає з формул (23), з многочленом $b_1 d_1(z)$. Після зведення остачі отримаємо $d_1(z)$ многочлен. Ділимо многочлен $f(z)$ на многочлен $zd_1(z)$ та отримаємо в остачі многочлен $b_2 d_2(z)$ і т.д.

В обчислювальному відношенні обидва способи реалізації методу Ліна є рівносильними. І.Ф. Греджук запропонував у методі Ліна ділення многочлена $f(z)$ не на $zd_k(z)$, а на многочлен $(z-\alpha)d_k(z)$. При цьому приходимо до такої обчислювальної схеми

$$\begin{aligned} &f(z) = \varphi_1(z)(z-\alpha)d_0(z) + b_1 d_1(z), \\ &f(z) = \varphi_2(z)(z-\alpha)d_1(z) + b_2 d_2(z), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &f(z) = \varphi_k(z)(z-\alpha)d_{k-1}(z) + b_k d_k(z). \end{aligned} \tag{25}$$

Якщо послідовність зведених многочленів $d_k(z)$ $(k = 0, 1, 2, \ldots)$ збігається, то знаходимо розклад на множники

$$f(z) = (\varphi(z)(z-\alpha) + b)d(z), \tag{26}$$

де $\varphi(z) = \lim\limits_{k\to\infty} \varphi_k(z)$, $\quad b = \lim\limits_{k\to\infty} b_k$, $\quad d(z) = \lim\limits_{k\to\infty} d_k(z)$.

Збіжність послідовності $d_k(z)$ $(k = 0, 1, 2, \ldots)$ залежить від вибору числа α. Одразу виникає питання про оптимальний вибір числа α.

Розкладемо многочлен $d_k(z)$ степені на лінійні комплексні множники

$$d_k(z) = (z-\beta_{k_1})\ldots(z-\beta_{k_q}) \tag{27}$$

Нехай виконана рівність

$$f(z) = \varphi_{k+1}(z)(z - \alpha_k)(z - \beta_{k_1})\ldots(z - \beta_{k_q}) + b_{k+1}d_{k+1}(z). \qquad (28)$$

Підставляючи в цю рівність значення аргументу $z = \alpha_k$, $z = \beta_{k_1}, \ldots, z = \beta_{k_2}$, отримаємо рівності

$$f(\alpha_k) = b_{k+1}d_{k+1}(\alpha_k),$$
$$f(\beta_{k_1}) = b_{k+1}d_{k+1}(\beta_{k_1}),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$f(\beta_{k_q}) = b_{k+1}d_{k+1}(\beta_{k_q}), \quad (k = 0, 1, 2, \ldots).$$

З яких випливає, що многочлен $b_{k+1}d_{k+1}(z)$ є інтерполяційним многочленом для $f(z)$ з вузлами інтерполяції $\alpha_k, \beta_{k_1}, \ldots, \beta_{k_q}$.

Припустимо, що многочлен $f(z)$ має групу близьких один до одного коренів $z_1, z_2, \ldots, z_q$ достатньо віддалених від інших коренів. Шукаємо зведений многочлен $d_{k+1}(z)$, корені якого близькі до коренів $z_1, z_2, \ldots, z_q$. Для цього задаємо вузли інтерполяції $\alpha_k, \beta_{k_1}, \ldots, \beta_{k_q}$ достатньо близькі до точок $z_1, z_2, \ldots, z_q$ й з формули (27) знаходимо інтерполяційний многочлен $d_{k+1}(z)$. Очевидно, що інтерполяція буде доброю, якщо точка α_k буде достатньо близькою до точок $z_1, z_2, \ldots, z_q$.

При звичайній трактовці методу Ліна як правило обираємо $\alpha_k = 0$, що іноді дає добрі результати в тому випадку, коли точки $z_1, z_2, \ldots, z_q$ достатньо близькі до точки. Якщо точки $z_1, z_2, \ldots, z_q$ віддалені від точки $z = 0$, то метод Ліна призводить до розбіжних наближень. Оскільки корені $z_1, z_2, \ldots, z_q$ многочлена $f(z)$ невідомі, а числа $\beta_{k_1}, \ldots, \beta_{k_q}$ за припущенням близькі до чисел $z_1, \ldots, z_q$, то величину α_k обираємо за формулою

$$\alpha_k = \frac{1}{q}\left(\beta_{k_1} + \ldots + \beta_{k_q}\right). \qquad (29)$$

Цей висновок забезпечує близькість a_k до чисел $z_1, \ldots, z_q$. Значення a_k можна легко знайти для заданого дільника

$$d_k(z) = z^q + c_{k_1} z^{q-1} + \ldots + c_{k_q},$$

не розв'язуючи рівняння $d_k(z) = 0$. А саме, справедливою є рівність

$$\alpha_k = -\frac{1}{q} c_{k_1} \qquad (k = 0, 1, 2, \ldots). \tag{30}$$

Із запропонованого узагальнення методу Ліна випливає, що його доцільно застосовувати для одночасного відшукання кратних q або близьких один до одного коренів многочлена *f(z)*. Очевидно, що при відшуканні близьких один до одного комплексних коренів, многочлени $d_k(z)$ будуть мати комплексні коефіцієнти.

Зауважимо, що багато відомих методів відшукання коренів алгебраїчного рівняння є частинними випадками методу Ліна.

Щоб знайти простий корінь $z = z_1$ рівняння *f(z)* = 0, можна поділити многочлен *f(z)* на дільник

$$d_k(z) = (z - \alpha)(z - \beta_k),$$

де α, β_k — числа досить близькі до кореня z_1. Прирівнюючи остачу від ділення $f(z)$ на $d_k(z)$ до нуля, отримаємо уточнене значення кореня β_{k+1}. У викладеному варіанті метод Ліна співпадає з методом лінійної інтерполяції або з методом хорд.

Аналогічно, при діленні многочлена *f(z)* на дільник $d_k(z) = (z - \beta_k)^2$ та прирівнюванні лінійної остачі до нуля, приходимо до методу Ньютона, оскільки будуть справедливими рівності

$$\begin{aligned} &f(z) = \varphi_{k+1}(z)(z - \beta_k)^2 + R_{k+1}(z), \\ &R_k(z) \equiv b_k z + c_k \qquad (k = 0, 1, 2, \ldots). \end{aligned} \tag{31}$$

Підставляючи $z = \beta_k$, знаходимо

$$R_{k+1}(\beta_k) \equiv f(\beta_k), \qquad R'_{k+1}(\beta_k) \equiv f'(\beta_k),$$

звідки випливає явний вираз для остачі

$$R_{k+1}(z) \equiv f'(\beta_k) z + f(\beta_k) - \beta_k f'(\beta_k).$$

Нове уточнене значення β_{k+1} кореня z_1 знаходимо з рівняння $R_{k+1}(z)=0$ і воно має вигляд

$$\beta_{k+1}=\beta_k-\frac{f(\beta_k)}{f'(\beta_k)}. \qquad (32)$$

Очевидно, що формула (32) співпадає з формулою, визначеною за методом дотичних.

У випадку простого кореня рівняння $f(z)=0$ застосування узагальненого методу Ліна призводить до відомих раніше методам хорд або дотичних. Але необхідно зауважити, що при застосуванні методу хорд не виникає труднощів при значеннях α близьких до β_k, які виникають при використанні інтерполяційної формули Лагранжа.

Викладемо алгоритм узагальненого методу Ліна.

1. Нехай відомо, що в околі точки $z=z_0$ рівняння $f(z)=0$ має групу q близьких один до одного коренів $z_1,\ldots,z_q$. Нехай знайдено многочлен

$$d_k(z)=z^q+c_{k_1}z^{q-1}+\ldots+c_{k_q} \qquad (k=0,1,2,\ldots), \qquad (33)$$

корені якого, відповідно близькі до $z_1,\ldots,z_q$. При $k=0$ можна покласти

$$d_0(z)=(z-z_0)^q. \qquad (34)$$

2. Ділимо вихідний многочлен $f(z)$ на многочлен $(z-\alpha_k)d_k(z)$, де $\alpha_k=-c_{k_1}z^{-1}$. Остачу від ділення після зведення позначимо через $d_{k+1}(z)$. Маємо рівність

$$f(z)=\varphi_{k+1}(z)(z-\alpha_k)d_k(z)\ldots(z-\beta_{k_q})+b_{k+1}d_{k+1}(z). \qquad (35)$$

При цьому знаходимо новий многочлен $d_{k+1}(z)$ степені q.

3. Послідовні наближення продовжуємо до збіжності послідовності многочленів $d_k(z)$ $d_k(z)$ до деякого многочлена

$$d_k(z)=z^q+c_{k_1}z^{q-1}+\ldots+c_{k_q}, \qquad d(z)=\lim_{k\to\infty}d_k(z).$$

При цьому справедливим буде розклад вихідного многочлена на множники

$$f(z) = (\varphi(z)(z-\alpha)+b)d(z), \tag{36}$$

$$\varphi(z) = \lim_{k\to\infty} \varphi_k(z), \quad b = \lim_{k\to\infty} b_k, \quad \alpha = \lim_{k\to\infty} \alpha_k.$$

4. Якщо послідовність многочленів $d_k(z)$ $(k = 0, 1, 2,\ldots)$ розбігається, то збільшуємо на одиницю значення степені q.

5. Якщо послідовність многочленів $d_k(z)$ $(k = 0, 1, 2,\ldots)$ збігається, то на друк виводимо многочлен $d(z)$, а многочлен $f(z) = \varphi(z)(z-\alpha)$ знову розкладаємо на множники.

6. Якщо заздалегідь не відомі значення z_0 і q, то значення z_0 обираємо випадково, а значення qзнаходимо з умови збіжності многочленів $d_k(z)$ $(k = 0, 1, 2,\ldots)$.

Література

1. Лященко М. Я., Головань М. С. Чисельні методи: Підручник. — К.: Либідь, 1996. — 288 с.

2. Аладьев В.З. Программирование и разработка приложений в Maple / 2-издание, В. З. Аладьев, В.К. Бойко, Е. А. Ровба. — Гродно: ГрГУ; Таллинн: Международная Академия Ноосферы. Балт. отд., 2014. — 458 с.

3. Шнайер Брюс Криптоанализ // Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. — М.: Триумф, 2002. — 816 с.

References

1. Liashchenko M. Y., HolovanM. S. (1996) Chyselnimetody: Pidruchnyk. — K.: Lybid (inUkrainian).

2. Alad'yev V .Z. (2014) Programmirovaniye i razrabotka prilozheniy v Maple/2-izd, V. Z. Alad'yev, V.K. Boyko, Ye. A. Rovba. — Grodno: GrGU; Tallinn: Mezhdunarodnaya Akademiya Noosfery. Balt. otd. (in Russian).

3. Shnayer B. (2002) Kriptoanaliz//Prikladnaya kriptografiya. Protokolyi, algoritmyi, ishodnyie tekstyi na yazyike Si — M.: Triumf. (in Russian).

Статтю подано до редакції 25.09.2019 р.