Харківський національний університет імені В.Н. Каразіна
Факультет математики і інформатики
Кафедра прикладної математики

Кваліфікаційна робота

бакалавра

на тему ***«Чисельне розв'язання одного звичайного диференціального рівняння в кільці кополіномів»***

Виконав: студент групи МП41,
спеціальність
113 – Прикладна математика,
освітньо-професійна програма
«Прикладна математика»
Матюшенко І.М.

Науковий керівник: канд. фіз.-мат. наук,
доцент кафедри
прикладної математики
Півень О.Л.

Рецензент: доктор технічних наук,
професор кафедри
комп'ютерно-інтегрованих
технологій, автоматизації
та робототехніки Харківського
національного університету
радіоелектроніки
Ромашов Ю.В.

Харків — 2025 рік

Анотації

Матюшенко Іван Миколайович, Чисельне розв'язання одного звичайного диференціального рівняння в кільці кополіномів. У кільці кополіномів над полем розглянуто один клас звичайних диференціальних рівнянь, який узагальнює диференціальне рівняння Ріккаті. Встановлено умови існування, єдиності розв'язку, загальний вигляд розв'язків. Розв'язки шукаються у вигляді ряду за степенями кополіноміальної дельта-функції, коефіцієнти якого задовольняють рекурентне рівняння. Наведено програмну реалізацію пошуку результату дії розв'язку на заданий поліном.

Ключові слова: кополіном, поліном, дельта-функція, звичайне диференціальне рівняння, чисельне розв'язання

Matyushenko Ivan, Numerical solution of one ordinary differential equation in a ring of copolynomials. In the ring of copolynomials over a field, one class of ordinary differential equations that generalizes the Riccati differential equation is considered. Conditions for existence and uniqueness of solutions and their general form are established. Solutions are sought as power series in terms of the copolynomial delta-function, whose coefficients satisfy a recurrence relation. A software implementation for computing the action of the solution on a given polynomial is provided.

Keywords: copolynomial, polynomial, delta-function, ordinary differential equation, numerical solution

Зміст

Вступ

Останнім часом в роботах С.Л. Гефтера, О.Л. Півня, Т.Є. Стулової [2–6] було розвинено теорію кополіномів та різних типів диференціальних рівнянь у модулі кополіномів $K[x]'$ над деяким цілим комутативним кільцем K з одиницею. Кополіном — це лінійний функціонал, який задано на кільці поліномів $K[x]$. У роботах [2, 3, 6] кополіноми називались формальними узагальненими функціями і теорія кополіномів нагадує класичну теорію узагальнених функцій. Лінійні диференціальні рівняння (як звичайні так і з частинними похідними) в модулі кополіномів досліджувались в роботах [2, 3, 6]. В роботі [4] було введено операцію множення кополіномів, яка перетворила модуль кополіномів на комутативне кільце. Це дозволило розглянути певний клас нелінійних диференціальних рівнянь з поліноміальною нелінійністю в кільці формальних степеневих рядів з кополіноміальними коефіцієнтами та довести теорему існування та єдиності розв'язку відповідної задачі Коші. Єдиний розв'язок розглянутої задачі Коші шукався у вигляді формального ряду за степенями δ-функції.

Нехай F — поле, $a, b \in F$, $T \in F[x]'$ — заданий кополіном та $n > 1$ — натуральне число. Об'єктом дослідження даної кваліфікаційної роботи є нелінійне звичайне диференціальне рівняння

$$u^{(n-1)}(x) = au^n + bu + T. \tag{0.1}$$

У випадку $n = 2$ рівняння (1) перетворюється в рівняння Ріккаті [1], а у випадку $n = 2$ та $T = 0$ — у рівняння Бернуллі [1].

В оглядовому розділі 1 кваліфікаційної роботи наводяться необхідні ві-

домості з теорії кополіномів, а саме означення кополіному, його похідних, збіжності кополіномів, а також добуток кополіномів. Розділ 2 присвячено дослідженню розв'язуваності рівняння (1). Виявляється, що випадки $b = 0$ та $b \neq 0$ суттєво відрізняються. Розв'язки рівняння (1) шукаються у вигляді ряду

$$u(x) = \sum_{k=0}^{\infty} u_k \delta^{k+1}$$

за степенями кополіноміальної δ-функції. Рівняння (1) зводиться до рекурентного рівняння відносно коефіцієнтів u_k. У роботі встановлено, що при $b \neq 0$ таке рекурентне рівняння має єдиний розв'язок, і тим самим доведено, що і відповідне диференціальне рівняння (1) має єдиний розв'язок (див. п. 2.2.1). Випадок $b = 0$ приводить до неявного рекурентного рівняння. Тут ми вже обмежуємось випадком $F = \mathbb{R}$, а $a \neq 0$ та $T = 0$. Виявляється, що таке рівняння може мати кілька розв'язків (див. п. 2.2.2). В розділі 3 наведено програму реалізації, яка обчислює коефіцієнти u_k розв'язків і для заданого поліному $p \in F[x]$ знаходить результат дії функціонала a на p, який, на відміну від u, є скінченною сумою.

Розділ 1

Кополіноми та їх властивості

Тут ми наведемо деякі властивості кополіномів [4]

1.1. Кополіноми, їх похідні та ряди з кополіномів

Визначення 1.1 Нехай F — поле дійсних чисел, і нехай $F[x]$ — кільце многочленів з коефіцієнтами в F. [4] Кополіном над полем F — це F-лінійний функціонал, визначений на кільці $F[x]$, тобто гомоморфізм з модуля $F[x]$ в поле F.

Позначимо модуль кополіномів над F через $F[x]'$. Тобто, $T \in F[x]'$, якщо і тільки якщо $T : F[x] \to F$ і T має властивість F-лінійності [4]:

$$T(ap + bq) = aT(p) + bT(q) \tag{1.1}$$

для всіх $p, q \in F[x]$ і $a, b \in F$. Якщо $T \in F[x]'$ і $p \in F[x]$, то для значення T на p ми використовуємо позначення (T, p). Також пишемо кополіном $T \in F[x]'$ у вигляді $T(x)$, де x - аргумент многочленів $p(x) \in F[x]$, на які діє F-лінійне відображення T. [4]

Визначення 1.2 [4] Похідна T' від кополіному $T \in F[x]'$ визначається за формулою :

$$(T', p) = -(T, p') \tag{1.2}$$

де $p \in F[x]$.

Похідна n-го порядку від кополіному в узагальненій формулі [4]:

$$(T^{(n)}, p) = (-1)^n (T, p^{(n)}) \tag{1.3}$$

де $p^{(n)}$ — це похідна $p(x)$ n-го порядку.

Крім того,

$$\left(\frac{T^{(n)}}{n!}, p\right) = (-1)^n \left(T, \frac{p^{(n)}}{n!},\right), \quad p \in F[x],$$

кополіноми $\dfrac{T^{(n)}}{n!}$ визначені коректно для будь-якого $T \in F[x]'$ і $n \in \mathbb{N}$. [4]

Приклад 1.1 [4] Кополіноміальна δ-функція визначається за формулою:

$$(\delta, p) = p(0), \quad p \in F[x].$$

Визначення 1.3 [4].Послідовність $\{T_n\}_{n=0}^{\infty}$ кополіномів з $F[x]'$ збігається до кополіному $T \in F[x]'$, якщо для кожного многочлена $p \in F[x]$ існує число $n_0 \in \mathbb{N}$ таке, що

$$(T_n, p) = (T, p), \quad \text{для всіх } n = n_0, n_0 + 1, n_0 + 2, \ldots$$

Ряд $\sum\limits_{n=0}^{\infty} T_n$ збігається в $F[x]'$, якщо збігається послідовність його часткових сум.

Теорема 1.1. *[4]. Нехай $\{a_n\}_{n=0}^{\infty}$ — послідовність елементів з F, і $T \in F[x]'$. Тоді ряд*

$$\sum_{n=0}^{\infty} \frac{a_n T^{(n)}}{n!}$$

збігається в $F[x]'$.

Приклад 1.2 Знайдемо

$$\sum_{n=0}^{\infty} n^3 \left(\frac{\delta^{(n)}}{n!}, x^5 \right).$$

Маємо

$$\sum_{n=0}^{\infty} n^3 \left(\frac{\delta^{(n)}}{n!}, x^5 \right) = \sum_{n=0}^{5} n^3 (-1)^n \left(\delta, \frac{(x^5)^{(n)}}{n!} \right)$$

$$= \sum_{n=0}^{5} n^3 (-1)^n \frac{(x^5)^{(n)}}{n!}\bigg|_{x=0} = 5^3 \cdot (-1)^5 \cdot 1 = -125$$

Наступна лема виявляє можливість розкладання будь-якого кополіному в збіжний ряд за кополіномами

$$\left\{ \frac{\delta^{(n)}}{n!} \right\}_{n=0}^{\infty}$$

Лема 1.2. *[4] Нехай $T \in F[x]'$. Тоді*

$$T = \sum_{n=0}^{\infty} (-1)^n \frac{(T, x^n)}{n!} \delta^{(n)}. \tag{1.4}$$

Приклад 1.3 [4] Для кополінома δ знайдемо його похідну n-го порядку :

$$(\delta^{(n)}, p) = (-1)^n (\delta, p^{(n)}) = (-1)^n p^{(n)}(0), \quad n \in \mathbb{N}.$$

Приклад 1.4 [4] Необхідно знайти $(-1)^k \left(\frac{\delta^{(k)}}{k!}, x^m \right)$ для $k, m \in \mathbb{N}_0$.

Маємо

$$(-1)^k \left(\frac{\delta^{(k)}}{k!}, x^m \right) = 1 \left(\delta, \frac{(x^m)^{(k)}}{k!} \right) = \begin{cases} 0, k \neq m, \\ 1, k = m. \end{cases}$$

1.2. Перетворення Коші-Стілтьєса

Визначення 1.4 Нехай F — поле, і $T \in F[x]'$. Тоді перетворення Коші-Стілтьєса для T визначається як функція $C(T)(s)$, що є формальним рядом Лорана [4]:

$$C(T)(s) = \sum_{k=0}^{\infty} \frac{(T, x^k)}{s^{k+1}} \tag{1.5}$$

Цей формальний ряд Лорана є елементом кільця $\frac{1}{s}F[[1/s]]$, де $F[[t]]$ - це кільце формальних степеневих рядів за змінною t та коефіцієнтами з F. [4]

Приклад 1.5 [4]

Для кополінома δ маємо:

$$C(\delta)(s) = \frac{1}{s}.$$

Приклад 1.6 [4] Нехай $T \in \mathbb{Q}[x]', (T, x^n) = n$ при $n \in \mathbb{N}_0$, тоді:

$$C(T)(s) = \sum_{k=0}^{\infty} \frac{k}{s^{k+1}}.$$

В [4] доведено, що відображення $C : F[x]' \to \frac{1}{s}F\left[\left[\frac{1}{s}\right]\right]$ є ізоморфізмом векторних просторів.

Теорема 1.3. *[4] Нехай $T \in F[x]'$. Тоді виконується рівність*

$$C\left(T^{(n)}\right) = (C(T))^{(n)}, \quad n \in \mathbb{N}. \tag{1.6}$$

1.3. Множення кополіномів

Перетворення Коші–Стілтьєса та теорема 1.3 дозволяють ввести операцію множення на модулі кополіномів таким чином, щоб ця операція була узгоджена з диференціюванням.

Визначення 1.5 Нехай $T_1, T_2 \in F[x]'$. Визначимо добуток T_1 та T_2 за допомогою рівності [4]:

$$C(T_1 T_2) = C(T_1)C(T_2) \tag{1.7}$$

тобто, оскільки C - ізоморфізм, то

$$T_1 T_2 = C^{-1}(C(T_1)C(T_2)),$$

У наступній лемі дію добутку кополіномів на одночлени виражено через дію множників на одночлени.

Лема 1.4. *[4] Нехай $T_1, T_2 \in F[x]'$ та $n \in \mathbb{N}_0$. Тоді*

$$(T_1 T_2, x^n) = \begin{cases} \sum_{k=0}^{n-1} (T_1, x^k)(T_2, x^{n-1-k}), & n \in \mathbb{N}, \\ 0, & n = 0. \end{cases}$$

Приклад 1.7

Нехай $T \in \mathbb{Z}[x]'$, $(T, x^n) = n$ для всіх $n \in \mathbb{N}_0$. Знайдемо $T \cdot \delta$. Користуючись лемою (1.4), отримуємо:

$$(T \cdot \delta, x^n) = \begin{cases} 0, & \text{якщо } n = 0 \\ \sum_{k=0}^{n-1} (T, x^k)(\delta, x^{n-1-k}), & \text{якщо } n \in \mathbb{N} \end{cases}$$

$$= \begin{cases} 0, & n = 0 \\ \sum\limits_{k=0}^{n-1} k \cdot (\delta, x^{n-1-k}), & n \in \mathbb{N} \end{cases}$$

$$= \begin{cases} 0, & n = 0 \\ (n-1) \cdot 1, & n \in \mathbb{N} \end{cases}$$

Отже

$$(T \cdot \delta, x^n) = \begin{cases} 0, & n = 0 \\ n-1, & n \in \mathbb{N} \end{cases}$$

В [4] встановлено зв'язок між степенями δ-функції та її похідними

$$(-1)^n \frac{\delta^{(n)}}{n!} = \delta^{n+1}, \quad n = 0, 1, 2, \ldots \tag{1.8}$$

і наслідок з цієї формули

$$(\delta^n)' = -n\delta^{n+1}, \quad n \in \mathbb{N}. \tag{1.9}$$

З формули (1.9) і теореми 1.1 випливає збіжність ряду (див. [4])

$$\sum_{k=0}^{\infty} u_k \delta^{k+1} = \sum_{k=0}^{\infty} (-1)^k \frac{\delta^{(k)}}{k!} u_k, u_k \in F \tag{1.10}$$

в $F[x]'$. Тепер з леми 1.2 випливає таке розкладання довільного кополіному T в $F[x]'$ в ряд за степенями δ-функції:

$$T = \sum_{k=0}^{\infty} (T, x^k) \delta^{k+1}. \tag{1.11}$$

Крім того, якщо $p(x) = \sum_{j=0}^{m} a_j x^j \in F[x]$, то

$$(T, p) = \sum_{j=0}^{m} a_j (T, x^j) \tag{1.12}$$

Розділ 2
Диференціальні рівняння в кільці кополіномів

Нехай $F-$ поле, $a, b \in F$, $n \in \mathbb{N}$, $n > 1$, $T \in K[x]'$ У кільці $F[x]'$ розглянемо звичайне диференціальне рівняння

$$u^{(n-1)} = au^n + bu + T \tag{2.1}$$

За формулою (1.11)

$$T = \sum_{k=0}^{\infty} t_k \delta^{k+1}, \ t_k = (T, x^k) \tag{2.2}$$

2.1. Постановка задачі та зведення диференціального рівняння до рекурентного

Будемо шукати розв'язок рівняння (2.1) у вигляді (1.11):

$$u = \sum_{k=0}^{\infty} u_k \delta^{k+1}, \tag{2.3}$$

де $u_k = (u, x^k)$ - невідомий елемент поля F. За визначенням добутку кополіномів отримуємо:

$$\begin{aligned} u^n = \left(\sum_{k=0}^{\infty} u_k \delta^{k+1} \right)^n = \sum_{\alpha_1=0}^{\infty} u_{\alpha_1} \delta^{\alpha_1+1} ... \sum_{\alpha_n=0}^{\infty} u_{\alpha_n} \delta^{\alpha_n+1} = \\ = \sum_{\alpha_1,...\alpha_n=0}^{\infty} u_{\alpha_1}...u_{\alpha_n} \delta^{\alpha_1+...+\alpha_n+n} \sum_{k=0}^{\infty} \sum_{|\alpha|=k} u_{\alpha_1}...u_{\alpha_n} \delta^{k+n}. \end{aligned} \tag{2.4}$$

Тут $\alpha = (\alpha_1, \alpha_2, ...\alpha_{n-1}, \alpha_n)$ - мультиіндекс, тобто вектор з цілими невід'ємними координатами, $|\alpha| = \sum_{i=1}^{n} \alpha_i$, та $\sum_{|\alpha|=k} u_{\alpha_1} u_{\alpha_2} ... u_{\alpha_{n-1}} u_{\alpha_n}$ - це сума за усіма можливи мультиіндексами α, для яких $|\alpha| = k$. Тоді $au^n = a\sum_{k=0}^{\infty} \sum_{|\alpha|=k} u_{\alpha_1} ... u_{\alpha_n} \delta^{k+n}$. Знаходимо похідні $u(x)$ до порядку $n-1$ включно за формулами (1.8) та (1.9):

$$u'(x) = \sum_{k=0}^{\infty} u_k (\delta^{k+1})' = -\sum_{k=0}^{\infty} (k+1) u_k \delta^{k+2},$$

$$u''(x) = -\sum_{k=0}^{\infty} (k+1) u_k (\delta^{k+2})' = \sum_{k=0}^{\infty} (k+1)(k+2) u_k \delta^{k+3},$$

.....

$$u^{(n-1)} = (-1)^{n-1} \sum_{k=0}^{\infty} (k+1)(k+2)...(k+n-1) u_k \delta^{k+n} \tag{2.5}$$

Підставляючи (2.2), (2.3), (2.5), (2.4) у рівняння (2.1), отримуємо:

$$\begin{gathered} (-1)^{n-1} \sum_{k=0}^{\infty} (k+1)(k+2)...(k+n-1) u_k \delta^{k+n} = \\ = a \sum_{k=0}^{\infty} \sum_{|\alpha|=k} u_{\alpha_1} u_{\alpha_2} ... u_{\alpha_n} \delta^{k+n} + b \sum_{k=0}^{\infty} u_k \delta^{k+1} + \sum_{k=0}^{\infty} t_k \delta^{k+1} \end{gathered} \tag{2.6}$$

Прирівнюючи коефіцієнти при рівних степенях δ-функції, отримуємо початкову задачу для рекурентного рівняння:

$$bu_k + t_k + a \sum_{|\alpha|=k-n+1} u_{\alpha_1} ... u_{\alpha_n} = (-1)^{n-1} \frac{k!}{(k-n+1)!} u_{k-n+1}, k \in \mathbb{N}, k \geq n-1 \tag{2.7}$$

$$bu_k + t_k = 0, k = 0, ..., n-2 \tag{2.8}$$

2.2. Розв'язання рекурентного рівняння

Якщо $b = 0$, то рівняння (2.7) за аналогією з [7] будемо називати неявним. Тому окремо розглянемо випадки $b = 0$ та $b \neq 0$

2.2.1. Випадок $b \neq 0$

Якщо $b \neq 0$, то початкова задача (2.7), (2.8) переписується у вигляді

$$u_k = \frac{1}{b}((-1)^{n-1}\frac{k!}{(k-n+1)!}u_{k-n+1} - a\sum_{|\alpha|=k-n+1} u_{\alpha_1}...u_{\alpha_n} - t_k), k = n-1, ... \tag{2.9}$$

$$u_k = -\frac{t_k}{b},\ k = 0, 1, ..., n-2 \tag{2.10}$$

Початкова задача (2.9), (2.10) має єдиний розв'язок, оскільки розв'язок рекурентного рівняння (2.9) однозначно визначається початковими умовами (2.10). Отже, ми одержали наступний результат.

Теорема 2.1. *Нехай $b \neq 0$. Тоді для будь-якого кополіному $T \in F[x]'$ існує єдиний розв'язок диференціального рівняння (2.1). Цей розв'язок має вигляд (2.3), де коефіцієнти u_k є розв'язком початкової задачі (2.9), (2.10).*

Зокрема, якщо $T = 0$, то $u = 0$.

2.2.2. Випадок $b = 0, T = 0, a \neq 0,$ та $F = \mathbb{R}$

Розглянемо випадок, коли $b = 0$, $T = 0$, $a \neq 0$ та $F = \mathbb{R}$. Рівняння (2.7) матиме вигляд:

$$(-1)^{n-1}\frac{(k+n-1)!}{k!}u_k = a\sum_{|\alpha|=k} u_{\alpha_1}...u_{\alpha_n}, k = 0, 1, 2, ..., \tag{2.11}$$

а початкова умова (2.8) перетворюється в тотожність. Отже, будемо розглядати неявне рекурентне рівняння (2.11).

Підставимо $k = 0$ в рівняння (2.11). Враховуючи, що $|\alpha| = 0$ тоді і тільки тоді, коли $\alpha_i = 0, i = 1, ...n$ маємо:

$$(-1)^{n-1}(n-1)!u_0 = a \cdot u_0^n$$

Рівняння має такі розв'язки:

$$u_0 = 0 \text{ та } u_0 = \sqrt[n-1]{\frac{(-1)^{n-1}(n-1)!}{a}} \tag{2.12}$$

(у випадку коли або n - парне, або n - непарне та $a > 0$)

При $k \in \mathbb{N}$ перетворимо формулу (2.11) таким чином, щоб права частина рівняння не залежала від u_k. Для цього перенесемо всі доданки в правій частині рівняння (2.11), для яких $\alpha_i = k$, у ліву частину цього рівняння. Зауважимо, що якщо $\alpha_i = k$ для деякого $i = 1, ..., n$, то $\alpha_j = 0, j \neq i$, і тому всі такі доданки мають вигляд $u_0^{n-1}u_k$, а їх кількість дорівнює n:

$$\frac{(-1)^{n-1}(k+n-1)!}{k!}u_k = a\left(\sum_{|\alpha|=k} u_{\alpha_1}...u_{\alpha_n} \cdot u_0^{n-1}u_k\right), k = 1, 2, 3, ...$$

і

$$u_k\left(\frac{(-1)^{n-1}(k+n-1)!}{k!} - a \cdot n \cdot u_0^{n-1}\right) = a\sum_{|\alpha|=k} u_{\alpha_1}...u_{\alpha_n}$$

У такому випадку формулу (2.11) для $k \in \mathbb{N}$ можна зобразити у вигляді:

$$C_{n,k}(u_0)u_k = a\sum_{|\alpha|=k} u_{\alpha_1}...u_{\alpha_n}, k = 1, 2, 3, ... \tag{2.13}$$

де

$$C_{n,k}(u_0) = \frac{(-1)^{n-1}(k+n-1)!}{k!} - a \cdot n \cdot u_0^{n-1} \tag{2.14}$$

Лема 2.2. *Нехай послідовність* $\{u_k\}_{k=0}^{\infty}$ *задовольняє рівняння (2.11). Якщо* $u_0 = 0$, *то* $u_k = 0, \forall k \in \mathbb{N}$

Доведення:

Випишемо рівняння для коефіцієнта $C_{n,k}$, за умови, що $u_0 = 0$

$$C_{n,k}(u_0) = \frac{(-1)^{n-1}(k+n-1)!}{k!} \neq 0, k = 1, 2, 3, ...$$

Тоді з (2.13) випливає, що $u_1 = 0$, та $u_2 = 0$ Далі за індукцією:
Якщо $u_1 = u_2 = ... = u_{k-1} = 0$ для деякого u_k, то $u_k = 0$. Це випливає з рівняння (2.13), а саме з того, що $C_{n,k}(0) \neq 0$ і при цьому:

$$\sum_{|\alpha|=k} u_{\alpha_1}...u_{\alpha_n} = 0, k = 1, 2, 3$$

Таким чином, якщо $u_0 = 0$, то і $\forall k \in \mathbb{N} : u_k = 0$. Лема доведена.

Лема 2.3. *Нехай послідовність* $\{u_k\}_0^{\infty}$ *задовольняє рівнянню (2.11) і* $u_0 \neq 0$, *тоді коефіцієнт* $C_{n,k}(u_0) = 0$ *тоді і тільки тоді, коли* $k = 1$.

Доведення:

Якщо $u \neq 0$, то згідно з формулою (2.12) $u_0 = \sqrt[n-1]{\frac{(-1)^{n-1}(n-1)!}{a}}$. Випишемо формулу (2.14) та підставимо до неї значення u_0 за умови, що $k \in \mathbb{N}, n > 1, n \in \mathbb{N}$:

$$C_{n,k}(u_0) = \frac{(-1)^{n-1}(k+n-1)!}{k!} - a \cdot n \cdot u_0^{n-1} =$$

$$= (-1)^{n-1} \left(\frac{(k+n-1)!}{k!} - n! \right)$$

Підставимо $k = 1$:

$$C_{n,1}(u_0) = (-1)^{n-1}\left(\frac{(1+n-1)!}{1!} - n!\right) = (-1)^{n-1}(n! - n!) = 0$$

Тепер доведемо, що при будь-якому фіксованому $n > 1$, $n \in \mathbb{N}$ послідовність

$$\frac{(k+n-1)!}{k!} - n!$$

зростає при збільшенні k.

Порівняймо

$$\frac{(k+n-1)!}{k!} - n! \quad \text{та} \quad \frac{(k+n)!}{(k+1)!} - n!$$

Для цього знайдемо різницю цих виразів:

$$\frac{(k+n)!}{(k+1)!} - n! - \left(\frac{(k+n-1)!}{k!} - n!\right) = \frac{(k+n)!}{(k+1)!} - \frac{(k+n-1)!}{k!}$$

$$= \frac{(k+n)!}{(k+1)!} - \frac{(k+n)!}{(k+1)!}\cdot\frac{k+1}{k+n} = \frac{(k+n)!}{(k+1)!}\left(1 - \frac{k+1}{k+n}\right)$$

$$= \frac{(k+n)!}{(k+1)!}\cdot\frac{n-1}{k+n} = \frac{(k+n-1)!\cdot(n-1)}{(k+1)!}$$

Так як за умовою $n > 1$, то різниця буде менша за нуль при будь-якому значенні n та k. А це значить, що вираз

$$\frac{(k+n)!}{(k+1)!} - n! > \frac{(k+n-1)!}{k!} - n!$$

і $C_{n,k}(u_0)$ строго зростає.

Таким чином, так як $C_{n,k}(u_0)$ строго зростає, то єдине значення $C_{n,k}(u_0) = 0$ при $k = 1$. Лему доведено.

Теорема 2.4. *Рекурентне рівняння (2.11) має нетривіальні розв'язки тоді і тільки тоді, коли або n — парне, або n— непарне, та $a > 0$.*

Доведення:

При $u_0 = 0$ твердження теореми 2.4 випливає з леми 2.2. Якщо $u_0 \neq 0$, то згідно з формулою (2.12)

$$u_0 = \sqrt[n-1]{\frac{(-1)^{n-1} \cdot (n-1)!}{a}}.$$

За лемою 2.3 $C_{n,1}(u_0) = 0$ і рівняння (2.14) є справедливим при будь-якому $u_1 \in \mathbb{R}$. Також за лемою 2.3, $C_{n,k}(u_0) \in \mathbb{R}$, $C_{n,k}(u_0) \neq 0$, $k = 2, 3, 4, \ldots$, тому з (2.12) випливає, що

$$u_k = (C_{n,k}(u_0))^{-1} a \sum_{\substack{|\alpha|=k \\ \alpha_i \neq k,\ i=1,\ldots,n}} u_{\alpha_1} \cdots u_{\alpha_n} \in \mathbb{R}, \quad k = 2, 3, 4, \ldots$$

З цього співвідношення випливають всі твердження теореми. Теорема доведена.

Наслідок 2.5. *Диференціальне рівняння*

$$u^{(n-1)}(x) = au^n(x) \tag{2.15}$$

має нетривіальний розв'язок в $\mathbb{R}[x]'$ *тоді і тільки тоді, коли або* n *- парне, або* n *- непарне і* $a > 0$. *При цьому нетривіальні розв'язки рівняння(2.15) описуються формулою:*

$$u(x) = \sum_{k=0}^{\infty} u_k \delta^{k+1}$$

де

$$u_0 = \sqrt[n-1]{\frac{(-1)^{n-1}(n-1)!}{a}}, u_1 - \textit{довільне дійсне число} \tag{2.16}$$

та

$$u_k = (C_{n,k}(u_0))^{-1} \sum_{|\alpha|=k, \alpha_i \neq k, k=1,2,3,\ldots} u_{\alpha_1} \ldots u_{\alpha_n} \tag{2.17}$$

Зокрема, якщо $u_0, u_1 \in \mathbb{Q}$, то $u(x) \in \mathbb{Q}[x]'$.

Отже, якщо u_0, яке визначається за формулою (2.12), та u_1, яке ви-

значається як довільне число при $u_0 \neq 0$, є раціональними, тоді всі члени послідовності $\{u_k\}_{k=0}^{\infty}$, що задовольняють рекурентне рівняння (2.11), є раціональними. Наступна теорема в частковому випадку $n = 2$ дає опис загального розв'язку рекурентного рівняння (2.11).

Теорема 2.6. *Якщо $n = 2$ та $u_0 \neq 0$, то $u_0 = \dfrac{-1}{a}$ та при будь-якому $u_1 \in \mathbb{R}$ загальний розв'язок рівняння (2.11) має вигляд:*

$$u_k = (-a)^{k-1} u_1^k, \quad k = 0, 1, 2, ... \tag{2.18}$$

Доведення: Спочатку обчислюємо u_0 за формулою (2.10):

$$u_0 = \left(\frac{(-1)^{n-1}(n-1)!}{a} \right)^{\frac{1}{n-1}} = \left(\frac{(-0)!}{a} \right)^{1} = \frac{1}{-a}$$

Тепер запишемо рівняння (2.11) для випадку $n = 2$:

$$-(k+1)u_k = a \sum_{j=0}^{k} u_j u_{k-j}, \quad k = 0, 1, 2, ...$$

Тоді

$$u_k = -\frac{a}{(k+1)} \left(\sum_{j=1}^{k-1} u_j u_{k-j} + 2u_0 u_k \right), \quad k = 2, 3, 4, \ldots$$

Проведемо перетворення цього рівняння:

$$u_k + \frac{2au_0}{k+1} u_k = -\frac{a}{(k+1)} \sum_{j=1}^{k-1} u_j u_{k-j}, \quad k = 2, 3, 4, \ldots$$

$$u_k \left(\frac{k-1}{k+1} \right) = -\frac{a}{k+1} \sum_{j=1}^{k-1} u_j u_{k-j}, \quad k = 2, 3, 4, \ldots$$

$$u_k = \frac{-a}{k-1} \sum_{j=1}^{k-1} u_j u_{k-j}, \quad k = 2, 3, 4, ... \tag{2.19}$$

Елемент u_1 може бути обраний довільним чином згідно з лемою 2.3.

Тепер доведемо формулу (2.18) методом математичної індукції. Вона є очевидною при $k = 0$ та при $k = 1$. Тому в якості бази індукції візьмемо $k = 2$.

$$u_k = \frac{(k-1)}{k+1} u_k = -\frac{a}{k+1} \sum_{j=1}^{k-1} u_j u_{k-j}, \quad k = 2, 3, 4, \ldots$$

$$u_k = \frac{-a}{k-1} \sum_{j=1}^{k-1} u_j u_{k-j}, \quad k = 2, 3, 4, \ldots$$

Крок 1 (база індукції) Якщо $k = 2$, то згідно з формулою (2.19)

$$u_k = \frac{-a}{k-1} \sum_{j=1}^{k-1} u_j u_{k-j} = \frac{-a}{1} (u_1 u_1) = -a u_1^2$$

Тому формула (2.18) є справедливою при $k = 2$.

Крок 2 (індукційний перехід). Нехай для певного натурального $s \geq 2$ усі номери послідовності від 1 до $s - 1$ визначаються за формулою (2.19). Доведемо, що тоді елемент u_s визначається за формулою (2.19). З формули (2.19) маємо:

$$u_s = \frac{-a}{s-1} \sum_{j=1}^{s-1} u_j u_{s-j} = \frac{-a}{s-1} \sum_{j=1}^{s-1} (-a)^{j-1} u_1^j \cdot (-a)^{s-j-1} u_1^{s-j} =$$

$$= \frac{-a}{s-1} \sum_{j=1}^{s-1} (-a)^{j-1+s-j-1} u_1^s = \ldots$$

$$= \frac{-a}{s-1} \sum_{j=1}^{s-1} (-a)^{s-2} u_1^s = \frac{-a}{s-1} (s-1)(-a)^{s-2} u_1^s = -a(-a)^{s-2} u_1^s =$$

$$= (-a)^{s-1} u_1^s$$

Таким чином формула (2.18) справедлива для всіх s.

Теорема доведена.

Наслідок 2.7.

Нетривіальні розв'язки рівняння

$$u' = au^2 \tag{2.20}$$

описуються формулою:

$$u(x) = \sum_{k=0}^{\infty} u_k \delta^{k+1} \tag{2.21}$$

де u_1 - довільне дійсне число,

$$u_k = \begin{cases} \dfrac{-1}{a}, k = 0 \\ (-a)^{k-1} u_1^k, k \neq 0. \end{cases} \tag{2.22}$$

Зокрема, якщо $a, u_1 \in \mathbb{Q}$, то $u(x) \in Q[x]'$.

Розділ 3
Чисельне розв'язання диференціального рівняння

Використовуємо програму написану в середовищі програмування *Python*, [8] яка знаходить такі елементи послідовності $\{u_k\}_{k=0}^{\infty}$, які задовольняють рівняння (2.1).

3.1. Випадок $b \neq 0$

Нехай в диференціальному рівнянні (2.1):

$$a = 2, b = 1, F = \mathbb{Q}, T = \sum_{k=0}^{\infty} \delta^{k+1}, n = 2$$

За теоремою 2.1 існує єдиний розв'язок рівняння (2.1). Цей розв'язок має вигляд

$$u(x) = \sum_{k=0}^{\infty} u_k \delta^{k+1}$$

Коефіцієнти $\{u_k\}_{k=0}^{\infty}$ є розв'язками початкової задачі (2.9), (2.10). З (2.10) маємо:

$$u_0 = -\frac{t_0}{b}$$

З рекурентного рівняння (2.9) маємо:

$$u_1 = 0,\ u_2 = -1,\ u_3 = -2,\ u_4 = -1.$$

Якщо $p(x) = \sum_{j=0}^{4} x^j$, то за формулою (1.12) $(u, p(x)) = -5$ Дана програ-

```python
import math
from fractions import Fraction
import sys

def is_valid_number(value):
    try:
        Fraction(value)
        return True
    except ValueError:
        return False

def generate_partitions(k, n):
    result = []

    def find_partitions(k, n, prefix):
        if n == 1:
            result.append(prefix + [k])
            return
        for i in range(k + 1):
            find_partitions(k - i, n - 1, prefix + [i])

    find_partitions(k, n, [])
    return result

def main():
    try:
        amount = int(input("Enter the number of elements in the sequence (amount): "))

        n_input = input("Enter the number n: ")
        if not n_input.isdigit() or int(n_input) < 2:
            print("invalid data")
            sys.exit()
        n = int(n_input)

        t = []
        print(f"Enter {amount} elements for the array t:")
        for i in range(amount):
            val = input(f"t[{i}]: ")
            if not is_valid_number(val):
                print("invalid data")
                sys.exit()
            t.append(int(Fraction(val)))

        a_input = input("Enter the value of constant a: ")
        if not is_valid_number(a_input):
            print("invalid data")
            sys.exit()
        a = int(Fraction(a_input))

        b_input = input("Enter the value of constant b: ")
        if not is_valid_number(b_input):
            print("invalid data")
            sys.exit()
        b = int(Fraction(b_input))

    except Exception:
        print("invalid data")
        sys.exit()

    u = []
    for i in range(amount):
        if i < n - 1:
            fraction = Fraction(-t[i], b)
        else:
            k = i + 1 - n
            sign = (-1) ** (n - 1)
            numerator = math.factorial(k + n - 1)
            denominator = math.factorial(k)

            if k < len(u):
                u_k = u[k]
            else:
                u_k = Fraction(0, 1)

            Sum_1 = Fraction(sign * numerator, denominator) * u_k

            if k == 0:
                Sum_2 = Fraction(0, 1)
            else:
                partitions = generate_partitions(k, n)
                Sum_2 = Fraction(0, 1)
                for row in partitions:
                    product = Fraction(1, 1)
                    for idx in row:
                        if idx >= len(u):
                            product *= Fraction(0, 1)
                        else:
                            product *= u[idx]
                    Sum_2 += product
                Sum_2 *= a

            Sum_3 = Fraction(t[i], 1)

            fraction = (Sum_1 - Sum_2 - Sum_3) / b

        u.append(fraction)

    print("Array u:")
    for idx, val in enumerate(u):
        print(f"u[{idx}] = {val}")


    print(f"Enter {amount} coefficients for polynomial a_i:")
    a_i = []
    for i in range(amount):
        val = input(f"a_i[{i}]: ")
        if not is_valid_number(val):
            print("invalid data")
            sys.exit()
        a_i.append(Fraction(val))

    P_sum = sum(a_i[j] * u[j] for j in range(amount))
    print(f"P_sum = {P_sum}")

if __name__ == "__main__":
    main()
```

ма призначена для обчислення елементів послідовності u_k на основі вхідних параметрів та рекурсивної формули.

Основна функція `main`

Функція `main` є центральною частиною програми. Вона відповідає за взаємодію з користувачем, зчитування вхідних даних, їх валідацію, запуск обчислень та вивід результатів.

Послідовність дій у `main`:

- Зчитування поліному $p(x)$, спепень якого записується в змінну `amount`. Це вказує на кількість членів послідовності, яку необхідно обчислити.
- Зчитування параметра n, що використовується у формулі для обчислення елементів послідовності. Значення n має бути цілим числом, не меншим за 2. Якщо введені дані некоректні, програма завершується з повідомленням про помилку.
- Зчитування масиву t, який містить `amount` елементів, кожен з яких повинен бути раціональним числом. Для цього використовується перевірка коректності введених значень.
- Зчитування констант a і b, які також мають бути раціональними числами.

Якщо будь-яке зі значень введене некоректно, програма негайно завершує роботу, виводячи повідомлення `"invalid data"`.

Далі відбувається обчислення послідовності u. Для індексів менших за $n-1$, елементи u_i обчислюються за простою формулою:

$$u_i = -\frac{t_i}{b}.$$

Для елементів з індексом $i \geq n-1$, обчислення здійснюється за рекурсивною формулою, в якій враховуються множини можливих мультиінде-

ксів (розбиття числа $k = i + 1 - n$ на n частин), коефіцієнти, факторіали та знакові множники.

Обчислення включає:

- Обчислення знака за формулою $(-1)^{n-1}$.
- Обчислення факторіального виразу $\dfrac{(k+n-1)!}{k!}$.
- Виклик функції `generate_partitions` для отримання всіх можливих мультиіндексів, які задають варіанти розбиття числа k.
- Обчислення суми добутків елементів u для кожного мультиіндексу.
- Підсумовування обчислених значень із константою a та входом t_i.
- Остаточне обчислення значення u_i шляхом ділення отриманої суми на константу b.

Після завершення обчислень усі значення послідовності u виводяться в консоль.

Функція `generate_partitions(k, n)`

Ця функція відповідає за генерацію всіх можливих способів розбиття числа k на n додатних або нульових частин. Вона реалізована рекурсивно і повертає список, що складається зі списків мультиіндексів. Кожен мультиіндекс описує один можливий варіант розподілу k на n частин, що використовується в рекурсивній формулі для обчислення u_i.

Функція `is_valid_number(value)`

Ця допоміжна функція перевіряє, чи є рядок `value` коректним представленням раціонального числа. Для цього вона намагається перетворити рядок у об'єкт типу `Fraction`. Якщо перетворення проходить успішно, повертає `True`, інакше — `False`. Дана функція використовується у `main` для перевірки коректності вхідних даних.

Після успішного обчислення всі елементи послідовності u виводяться в консоль по черзі, у форматі:

$$u[0] = \ldots, \quad u[1] = \ldots, \quad \ldots$$

Далі, окрім елементів $u[i]$, программа зчитує коефіцієнти певного поліному p та виводить результат дії функціоналу u на вибраний поліном.

Таким чином, програма забезпечує коректне обчислення та вивід послідовності u з урахуванням заданих параметрів та вхідних даних, включаючи перевірку їх коректності і реалізацію рекурсивного алгоритму з комбінаторними обчисленнями.

3.2. Випадок $b = 0, T = 0, a \neq 0$ та $F = \mathbb{R}$

Нехай $n = 2$. Розглянемо диференціальне рівняння (2.20) в $\mathbb{R}[x]'$ при $a = 2$.

За наслідком 2.7 розв'язки рівняння (2.20) мають вигляд $u(x) = \sum_{k=0}^{\infty} u_k \delta^{k+1}$, де $u_0 = -\frac{1}{2}$, $u_1 \in \mathbb{R}$ – довільне число. $u_k = (-2)^{k-1} u_1, k = 2, 3, \ldots$ Тепер розглянемо диференціальне рівняння (2.15) $n = 3, a = 18, F = \mathbb{R}$. За наслідком 2.5, диференціальне рівняння (2.15) має нетривіальні розв'язки.

Загальний розв'язок цього рівняння має вигляд

$$u(x) = \sum_{k=0}^{\infty} u_k \delta^{k+1},$$

де u_k обчислюється за формулами (2.16) та (2.17).

За формулою (2.19)

$$u_0 = \frac{1}{3}$$

Далі обираємо $u_1 = 1$. Тоді за формулою (2.17)

$$u_2 = 3,$$

$$u_3 = 9.$$

Якщо поліном $p(x) = \sum_{k=0}^{3} x^k$, то за формулою (1.12) $(u, p) = 13\frac{1}{3}$

Код на мові програмування *Python* для виконання відповідних обчислень:

main — метод, в якому викликаються усі інші, тобто основна частина програми.

В ньому послідовно зчитуються значення n, a та u_1 з консолі, створюється масив раціональних чисел, обчислюється значення u_0 та розраховуються за допомогою методу `recurtion` усі інші елементи послідовності окрім u_1 і виводяться як елементи масиву u.

При обчисленні u_0, за допомогою методу `realcheck` перевіряється, чи є дійсним значення елементу u_0.

Окрім цього, тут використовується змінна `deep`, яка відповідає кількості елементів послідовності $\{u_k\}_{k=0}^{\infty}$, які необхідно вивести.

`recurtion` — метод, який обчислює значення елементу послідовності u_k.

За допомогою методу `binomial_coeff` визначаються розміри подвійного циклу, який потрібно пройти за згенерованим масивом мультиіндексів, знайденому за допомогою методу `generate_partition`, та визначається сума правих частин рекурентного рівняння для відповідних мультиіндексів. Після цього отримана сума ділиться на коефіцієнт з лівої частини рівняння (2.13), знайдений за допомогою методу `coefficient`, та помножається на a.

Цей метод відповідає рекурсивній формулі (2.17) і є виконанням її для вже визначених значень $u_i, i = 2, 3, 4, ...$

```python
import math
from fractions import Fraction
import sys

def is_valid_number(value):
    try:
        Fraction(value)
        return True
    except ValueError:
        return False

def realcheck(a, n):
    factorial = math.factorial(n - 1)
    value = Fraction(a, factorial)
    root = round(value ** (1 / (n - 1)))
    if root ** (n - 1) == value:
        return root
    return 0

def coefficient(n, deep):
    result = Fraction(deep + 1)
    for i in range(1, n - 1):
        result *= Fraction(deep + i + 1)
    result *= Fraction((-1) ** (n - 1))
    return result

def binomial_coeff(n, deep):
    if deep > n:
        return 0
    if deep == 0 or deep == n:
        return 1
    return math.comb(n, deep)

def generate_partitions(k, n):
    A = binomial_coeff(n + k - 1, k)
    result = []
    def find_partitions(k, n, prefix):
        if n == 1:
            result.append(prefix + [k])
            return
        for i in range(k + 1):
            find_partitions(k - i, n - 1, prefix + [i])
    find_partitions(k, n, [])
    return result

def recurtion(a, n, deep, u):
    result = Fraction(0)
    binom_size = binomial_coeff(n + deep - 1, deep)
    multiindex = generate_partitions(deep, n)
    polinom = [Fraction(1)] * len(multiindex)

    for i in range(len(multiindex)):
        for j in range(n):
            if multiindex[i][j] < len(u):
                polinom[i] *= u[multiindex[i][j]]
            else:
                polinom[i] *= Fraction(1)

    result = sum(polinom)
    result *= Fraction(a)
    denom = Fraction(coefficient(n, deep)) - Fraction(a) * n * u[0]**(n - 1)
    result /= denom
    return result

def main():

    n = int(input("Введіть значення n: "))
    amount = int(input("Введіть кількість елементів послідовності, які ви хочете обчислити: "))
    amount += 1

    a = Fraction(input("Введіть значення a: "))

    real_check_result = realcheck(a, n)
    if real_check_result != 0:
        real_check_result *= (-1) ** (n-1)
        u0 = Fraction(1, real_check_result)
    else:
        print("Введені дані не дозволяють розв'язати рівняння в раціональних дробах")
        sys.exit()

    u = [Fraction(0)] * amount
    u[0] = u0

    u[1] = Fraction(input("Введіть значення u[1]: "))

    for i in range(2, amount):
        u[i] = recurtion(a, n, i, u)

    for i in range(amount):
        value = u[i]
        if isinstance(value, Fraction):
            print(f"u[{i}] = {value.numerator}/{value.denominator}")
        else:
            print(f"u[{i}] = {value}")

    print(f"Введіть {amount} коефіцієнтів для полінома a_i, через пробіл або кому:")
    a_i_input = input(f"a_i: ")

    a_i = []
    try:
        a_i = [Fraction(x.strip()) for x in a_i_input.replace(',', ' ').split()]

        if len(a_i) != amount:
            print(f"Помилка: введено {len(a_i)} коефіцієнтів, а потрібно {amount}.")
            sys.exit()
    except ValueError:
        print("Невірний ввід коефіцієнтів, спробуйте ще раз.")
        sys.exit()

    P_sum = sum(a_i[i] * u[i] for i in range(amount))
    print(f"P_sum = {P_sum}")

if __name__ == "__main__":
    main()
```

`generate_partition` - метод, в якому за значеннями n, k визначається масив мультиіндексів.

Цей метод створює подвійний масив, рядки якого відповідають n номерам кожного з можливих мультиіндексів α з умови задачі, для поточного значення k, n. Для цього використовується прошитий метод `find.partition`, який ділить число k на n частин. У програмі наявний маркер максимального числа, таким чином, щоб наприкінці розбиття виділити n підмасивів, які відповідають мультиіндексам з нульовими значеннями всіх, окрім одного елементів. Саме ці мультиіндекси відповідають перенесеним поліномам.

`binomial_coeff` - метод, в якому визначається кількість розбиттів мультиіндексів.

Це загальний метод, який використовує прошитий метод `math.comb(n, deep)`, що генерує біноміальний коефіцієнт від відповідних значень. В самому методі використовуються певні граничні значення $n, deep$, щоб повертати коректні розбиття за граничних умов.

`coefficient` — метод, в якому обчислюються коефіцієнт у лівій частині рівняння (2.14).

`realcheck` — метод, в якому перевіряється, чи є число u_0 раціональним, чи ні. Якщо воно є раціональним, то метод повертає його, якщо ні — метод повертає нуль.

Висновки

Наведено огляд результатів по кополіномам. Для звичайного диференціального рівняння

$$u^{(n-1)}(x) = au^n + bu + T$$

в кільці кополіномів $F[x]^{'}$ при $b \neq 0$ встановлено умови існування та єдиності розв'язку, а у випадку $b = 0$, $T = 0$ та $a \neq 0$ — загальний вигляд нетривіальних розв'язків. Розв'язки зазначеного рівняння шукаються у вигляді ряду за степенями кополіноміальної δ-функції з коефіцієнтами, що задовольняють рекурентне рівняння. Наведено програмну реалізацію, яка розв'язує це рекурентне рівняння і надає результат дії розв'язку на заданий поліном. Програмну реалізацію виконано у середовищі Python.

Список використаних джерел

[1] Самойленко А.М., Перестюк М.О., Парасюк І.О. Диференціальні рівняння. Підручник. – К.: Либідь, 2003. – 600 с.

[2] Гефтер С.Л., Півень О. Л. Диференціальні оператори нескінченного порядку в модулі формальних узагальнених функцій та у кільці формальних степеневих рядів. *Український математичний журнал.* 2022. Т. 74(6). С. 784–799.

[3] Gefter S.L., Piven' A.L. *Linear Partial Differential Equations in Module of Formal Generalized Functions over Commutative Ring.* J. Math. Sci. 2021. V.257(5). P.579–596.

[4] Gefter S.L., Piven' A.L. *Some class of nonlinear partial differential equations in the ring of copolynomials over a commutative ring.* Front. Appl. Math. Stat. 2024. 10:1466569.

[5] Gefter S.L., Piven' A.L. *Partial Differential Equations in Module of Copolynomials over a Commutative Ring.* J.Math. Phys. Anal. Geom. 2025. V. 21(1). P.56–83.

[6] Gefter S.L., Stulova T.E., *Fundamental Solution of the Simplest Implicit Linear Differential Equation in a Vector Space.* J. Math. Sci. 2015. V. 207(2). P.166–175.

[7] M. V. Heneralov, A. L. Piven' *Implicit linear difference equation over residue class rings. Algebra and Discrete Mathematics.* . 2024. Vol. V.37 (1). P. 85–105.

[8] Sam Morley, *Applying Math with Python*, 2nd edition, *PACT PUBLISHING LTD*, 2020, 331 c.