ЛАБОРАТОРНА РОБОТА № 7
НЕЛІНІЙНА ОБРОБКА СИГНАЛІВ

Завдання:

1. Вивчити базові принципи нелінійної обробки сигналів.
2. Написати програму для розрахунку прямим та нелінійним методом (N – порядковий номер студента в журналі):
   - $\sqrt{x_1^2 + x_2^2}$ (діапазон чисел $x \in \left[N \times 10^6 \div (N+1)10^6 \right]$)
   - $\cos x$ ($x \in \left[2N\pi \div 2(N+1)\pi \right]$)
   - Поліном $y(x) = \sum_{i=3}^{N} i x^i$. Нелінійна обробка – за алгоритмом Горнера. $x \in \left[N \times 10^6 \div (N+1)10^6 \right]$
3. Забезпечити в програмі обчислення відносної похибки δ між результатом розрахунку за прямим та нелінійним методом.
4. Записати у файл **`nonlinear.dat`** в 4 стовпці:

Число	Розрахунок прямим методом	Розрахунок нелінійним алгоритмом	Відносна похибка

5. Порахувати та порівняти машинний час для розрахунку прямим та нелінійним методом (для помітних результатів потрібно розглянути >1 млн. розрахунків).

ТЕОРЕТИЧНІ ВІДОМОСТІ

Нелінійна обробка

Розглянемо приклади найпоширеніших алгоритмів нелінійної обробки сигналів.

1. Обчислення квадратного кореня

Однією з типових практичних задач ЦОС є багаторазове обчислення АЧХ фільтра з передаточною функцією $H(z) = \frac{1}{A(z)}$ за досить короткий часовий відрізок, що характерно для вокодерів з лінійним передбаченням. Для цього буде потрібно на великому безлічі частот $\{\omega_i\}$ обчислювати величину

$$\left| H\left(e^{j\omega T}\right) \right| = \frac{1}{\left| A\left(e^{j\omega T}\right) \right|} = \sqrt{x_1^2 + x_2^2}, \qquad (1)$$

де

$$z = e^{j\omega T}, \qquad x_1 = \mathrm{Re}\left\{ A\left(e^{j\omega T}\right) \right\}, \qquad x_2 = \mathrm{Im}\left\{ A\left(e^{j\omega T}\right) \right\}, \qquad (2)$$

Якщо апаратна реалізація операції розрахунку кореня квадратного в процесорах відсутня, то її втілюють програмно, що вимагає декількох десятків команд них циклів процесора. У завданнях реального часу це може виявитися непри-

пустимим, тому було запропоновано апроксимувати функцію $\sqrt{x_1^2 + x_2^2}$ поліномом невисокого порядку F, який є зручним для програмної реалізації на мові асемблера і відповідає необхідній точності.

Розв'язок мінімаксної задачі приводить до поліному другого порядку

$$\boxed{F = 0{,}828427\left(x_1 + x_2\right) - 0{,}3431452x_1x_2}, \tag{3}$$

який дає похибку $\delta \in \left[0;\, 0{,}171573\right]$, звідки $\varepsilon_{\max} = 1{,}636$ дБ, що задовольняє вимогам, що пред'являються до систем обробки мови.

2. Обчислення функції $\cos\omega$

Обчислення тригонометричних функцій неважко організувати через обчислення $\cos\omega$. для чого найчастіше в пам'яті процесора формують таблиці. до яких звертаються в ході обчислень. Коли створення таблиць ускладнене або недоцільно внаслідок недостатньої точності, необхідно організовувати спеціальну обчислювальну процедуру, яка відповідала б двом критеріям: необхідної точності і мінімуму команд. З цих позицій найкращою виявилася апроксимація у вигляді відрізка ряду за поліномами Чебишева

$$\cos\left(\lambda x\right) = \sum_{n=0}^{\infty} c_n\left(\lambda\right) T_{2n}\left(x\right), \tag{4}$$

де $x = \dfrac{\omega}{\lambda}$; $\lambda = \pi$, тобто $x \in \left(0;1\right)$. Дослідження, виконані по відношенню до компресії мови, показали, що при допустимому відхиленні частотної характеристики достатньо мати шах $\max\left\{n\right\} = 3$. У цьому випадку отримуємо

$$\boxed{\cos x = 0{,}998568 - 4{,}888184x^2 + 3{,}819208x^4 - 0{,}930944x^6}. \tag{5}$$

3. Обчислення поліномів

Алгебраїчні і тригонометричні поліноми, використовувані в цифровій обробці, можуть бути представлені в алгебраїчному вигляді простою заміною змінних. Тому алгоритм обчислення всякого полінома неважко звести до обчислення алгебраїчного полінома

$$y\left(x\right) = \sum_{i=0}^{N} a_i x^i. \tag{6}$$

Ясно, що обчислення полінома безпосередньо за формулою (6) вимагає N операцій додавання і $\dfrac{\left(N-1\right)N}{2}$ операцій множення, тобто всього необхідно виконати $\dfrac{\left(N+1\right)N}{2}$ арифметичних операцій, що при великих N, характерних для ЦОС, може стати критичним щодо забезпечення реального часу. Крім того, при використанні арифметики чисел з фіксованою комою у процесі обчислення можуть з'явитися дві небезпеки: одна – переповнення в суматорах, інша – втрата значущості внаслідок зведення у велику ступінь числа x, за абсо-

лютною величиною не перевищує 1.

Щоб уникнути перерахованих недоліків прямого обчислення полінома використовують ***алгоритм Горнера***

$$\boxed{y(x) = a_0 + x\left(a_1 + x\left(a_2 + \ldots + a_N\right)\ldots\right)}, \tag{7}$$

Який вимагає N добутків і N операцій додавання, тобто загальна кількість арифметичних операцій становить всього $2N$, що в $\frac{N+1}{4}$ рази менше, ніж у разі прямого обчислення. Даний алгоритм також не приводить до переповнення і втрати значущості.

Алгоритм Горнера неважко застосувати і для обчислення багатовимірних поліномів. Процедуру пояснимо на прикладі двовимірного полінома

$$y(x) = \sum_{i}^{\frac{N}{2}} \sum_{j}^{\frac{N}{2}} a_{ij} x_1^i x_2^j \,. \tag{8}$$

Розкриємо цей поліном, тобто запишемо його у вигляді сум:

$$\begin{aligned} y(x) = a_{00} + \\ + a_{10}x_1 + a_{01}x_2 + \\ + a_{20}x_1 + a_{11}x_1x_2 + a_{02}x_2 + \\ \ldots \\ + a_{N0}x_1^N + a_{N-1}x_1^{N-1}x_2 + \ldots + a_{0N}x_2^N \end{aligned} \tag{9}$$

звідки, взявши підсумовування по стовпцях, отримаємо більш зручну запис

$$y(x) = \left[A_0(x_1)\right]x_2^0 + \left[A_1(x_1)\right]x_2^1 + \left[A_2(x_1)\right]x_2^2 + \ldots + A_N(x_1)x_2^N \,. \tag{10}$$

Таким чином, можна записати

$$y(x) = \sum_{i=0}^{N} A_i(x_1)x_2^i \tag{11}$$

тобто вихідний двовимірний поліном можна розглядати як одновимірний, коефіцієнти якого

$$A_i(x_1) = \sum_{j=0}^{N-i} a_{j,i}x_1^j \tag{12}$$

також одновимірні поліноми. Зрозуміло, що обчислення одновимірних поліномів слід здійснювати за допомогою алгоритму Горнера.

4. Медіанні фільтри

Серед різноманітних адитивних перешкод в тимчасовій області часто зустрічаються імпульсні перешкоди, які діють на дуже коротких інтервалах часу і накладаються на відлік сигналу. Одним з ефективних методів боротьби з подібними перешкодами є застосування медіанному фільтрації, алгоритм якої полягає в наступному:

1. Вибирається непарна кількість до відліків прийнятого сигналу, починаючи з 1-го відліку.

2. Відлік розташовуються в порядку зростання за амплітудою.
3. Центральний відлік даної вибірки, званий медіаною, береться в якості представника цієї вибірки.
4. Вибираються нові до відліків прийнятого сигналу, починаючи з (i + 1)-го відліку, і виконуються дії 2–4.

Зазначимо, що алгоритм починається з $i = 0$.

Алгоритм медіанної фільтрації пояснюється на прикладі рис. 1, а результат відображено в табл. 1.

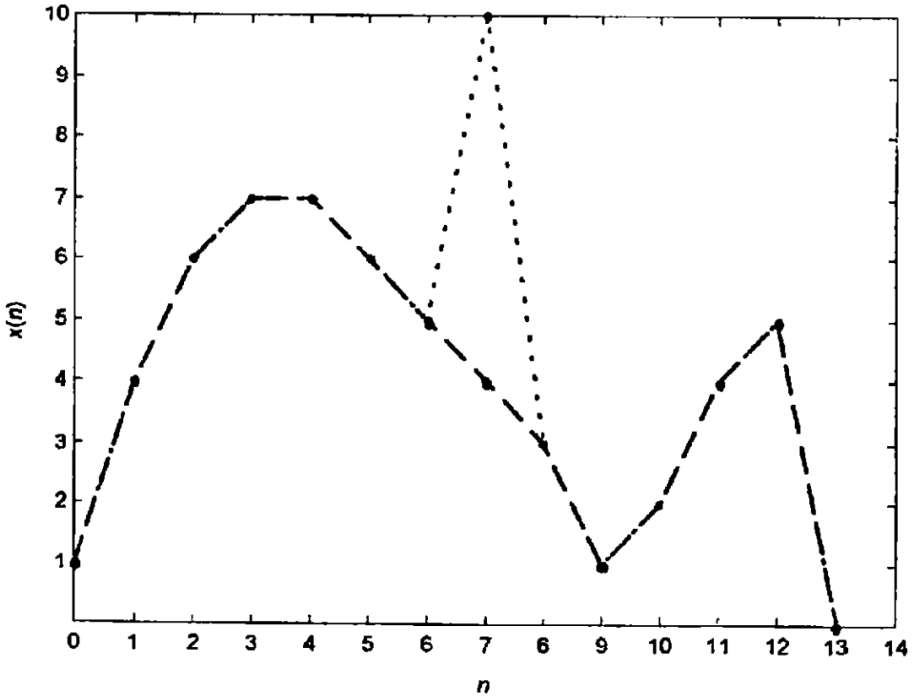

Рис. 1. Вихідна і прийнята послідовності

На рис. 1 точками відзначені значення відліків сигналу, що передається, який, як видно, є гладким. Припустимо, що під час передачі в каналі діяла імпульсна перешкода, в результаті чого сьомий відлік отримав значення 10, і у прийнятій послідовності утворився шумовий імпульс:

Значення відліку	1	4	6	7	7	6	5	10	3	1	2	4	5	0
Номер відліку	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Створимо вибірки з цього ряду по три послідовних відліку, переставимо їх у порядку зростання і візьмемо медіану, як показано в табл. 1. Результат фільтрації зображений на рис. 2, де точками відзначені відліки підсумкового сигналу. З рис. 2 і табл. 1 видно, що усунення імпульсної перешкоди зробило сигнал більш пласким в областях максимумів і мінімумів, розташованих праворуч поблизу відліку, на який вплинула імпульсна перешкода.

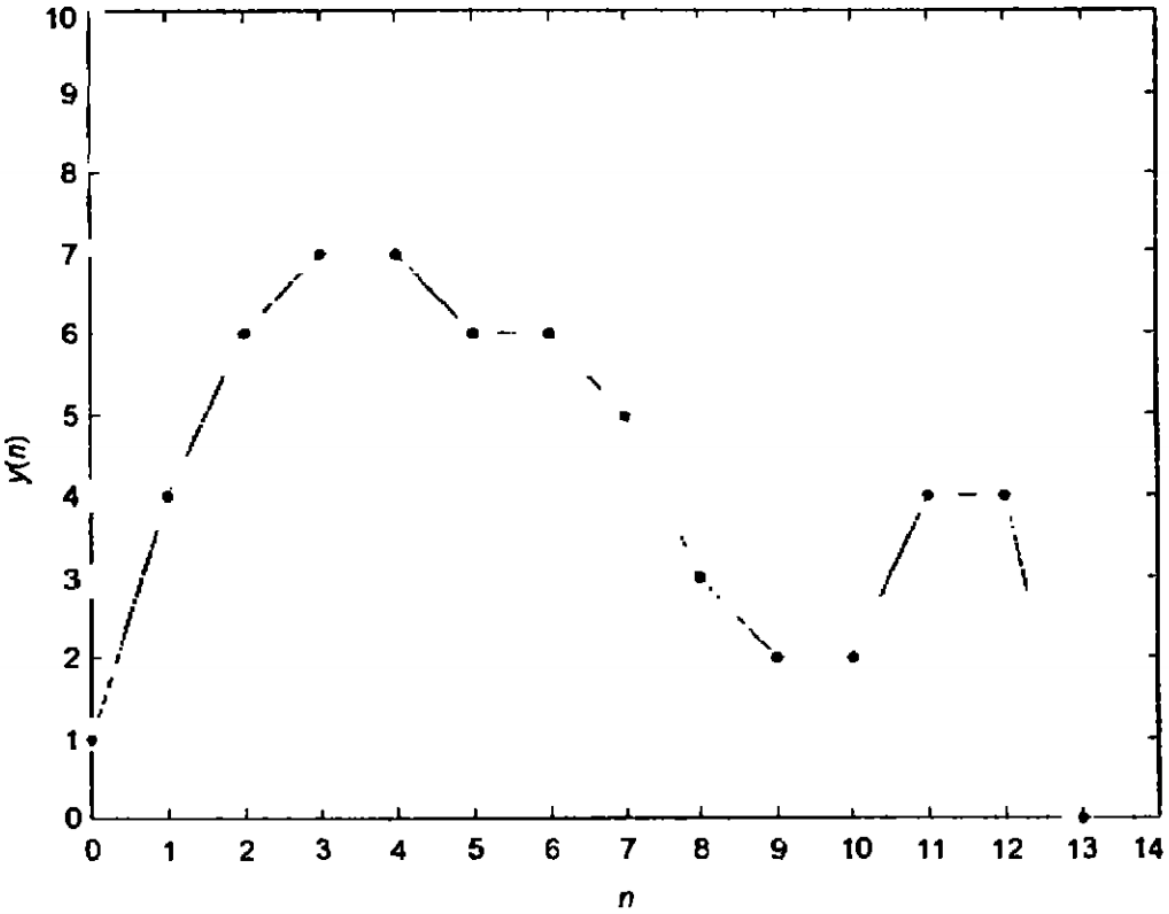

Рис. 2. Послідовність на виході медіанного фільтра

Таблиця 1.

Представлення медіанної фільтрації

Три послідовних відліки		Медіана	Номер відліку фільтрованого сигналу	Результат
Прийнято	Перестановка			
1 4 6	1 4 6	4	1	4
4 6 7	4 6 7	6	2	6
6 7 7	6 7 7	7	3	7
7 7 6	6 7 7	7	4	7
7 6 5	5 6 7	6	5	6
6 5 10	5 6 10	6	6	5
5 10 3	3 5 10	5	7	10
10 3 1	1 3 10	3	8	3
3 1 2	1 2 3	2	9	1
1 2 4	1 2 4	2	10	2
2 4 5	2 4 5	4	11	4
4 5 0	0 4 5	4	12	5
5 0 0	0 0 5	0	13	0

5. Векторне квантування

Нагадаємо, що відоме скалярний квантування, здійснюване аналогоцифровим перетворювачем, полягає в тому. що кожному відліку сигналу $x(t)$ ставиться у відповідність двійкове число $x(n)$; в канал зв'язку передається також двійкове число $y(n)$, одержуване в обчислювачі. Кількість таких чисел визначається частотою дискретизації / ,, а швидкість з їх передачі в каналі залежить ще й від розрядності Ь представлення чисел

$$c = bf_i \tag{13}$$

і при передачі по каналу, наприклад, мовного сигналу при стандартних $b = 12$, $f_Д = 8000$ Гц отримуємо $c = 96000$ біт/с, що перешкоджає використанню стандартних телефонних каналів. Можна вчинити інакше (рис. 3): визначити в кодованій послідовності L блоків по k відліків в кожному за умови, що ці відліки не сильно відрізняються один від одного. Ці k відліків можуть бути відображені із заданою помилкою (наближенням) споїмо представником y. Такі блоки називають *кластерами*, а представника i-го кластера y – *центроїдом*. Тоді кожен відлік $x(n)$, що належить i-му кластеру, замінюється відповідним центроїдом, і в канал зв'язку передається тільки номер кластера (центроїда). На передачі та прийомі необхідно мати множину з L центроїдів, яке називається кодовою книгою, а параметр L – розміром кодової книги.

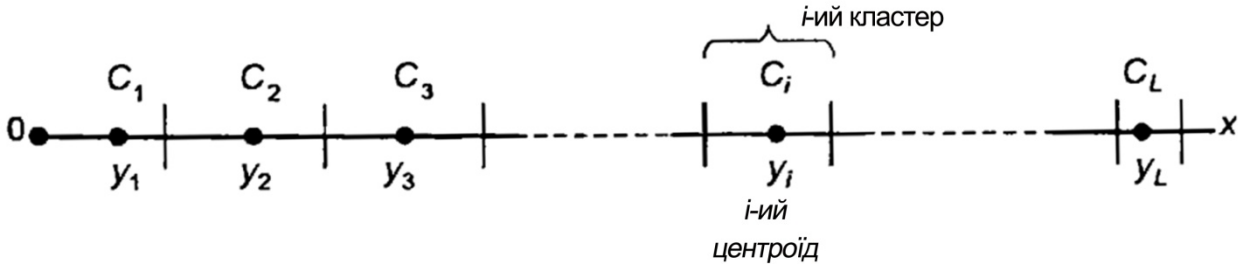

Рис. 3. Кластери та центроїди одновимірної послідовності

У загальному випадку вхідний послідовністю можуть бути k-вимірні вектори (на рис. 4 $k = 2$), що характерно для лінійного передбачення, якщо $k \geq 10$ і на кожному кадрі лінійного передбачення доводиться передавати десять параметрів (лінійних спектральних коренів), на які відводиться не більше 40 бітів.

Типові розміри кодових книг в мовних технологіях складають 256, 512, 1024 і 2048 центроїдів (кластерів). Якщо $L = 1024 = 2^{10}$; це означає, що кодова книга містить 1024 центроїда розмірністю $k = 10$ кожен. Отже, замість набору з десяти параметрів лінійного передбачення передається тільки номер кластера (а тому й номер центроїда), якому належить даний набір, на що буде потрібно всього 10 бітів, тобто забезпечується стиснення сигналу в

$$K_{СТ} = 40/10 = 4 \text{ рази.}$$

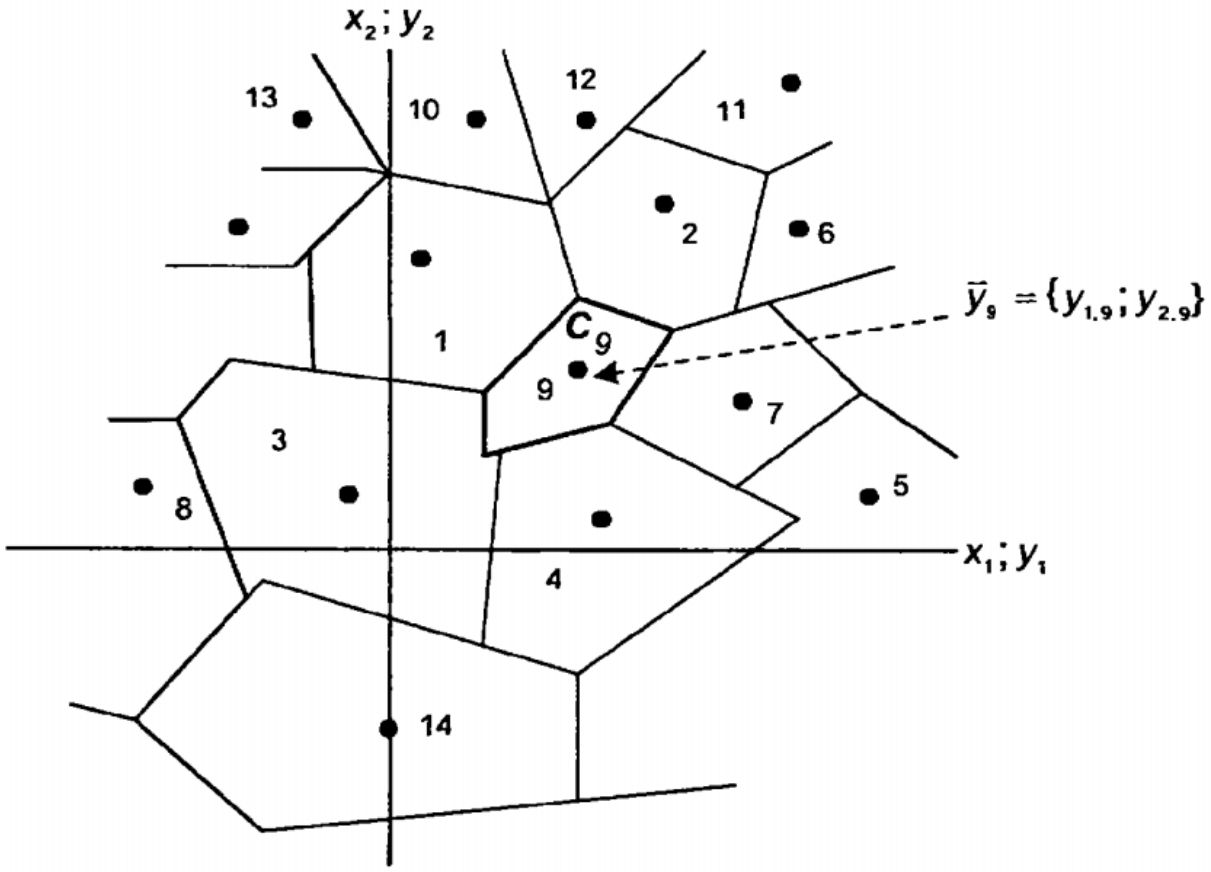

Рис. 4. Розбиття двовимірного масиву на кластери (точками позначені двовимірні центроїди)

На практиці $K_{\text{СТ}}$ може бути й істотно великим, що визначається співвідношеннями:

$$K_{\text{СТ}} = \frac{b}{R}, \qquad R = \frac{log_2 L}{k}, \tag{14}$$

де b – розрядність одного відліку k-мірного вектора: R – кількість бітів, що витрачаються на передачу одного відліку. Пошук відповідного центроїда у кодовій книзі простим перебором неефективний. Тому її задають у вигляді бінарного дерева з розщепленням кожного вузла на два (рис. 5). Центроїди $y_i = \{y_{j1}, y_{j2}, \ldots, y_{jk}\}$ у вузлах дерева нумеруються зверху вниз і зліва направо. Нерозщеплені вузлу (третього рівня пошуку в нашому випадку) присвоюється ознака S = 0. Перехід по лівій гілці з верхнього рівня на нижній відповідає запису нуля в регістр номера шуканого центроїда; інакше записується 1. Звідси отримуємо наступний алгоритм пошуку центроїда (рис. 6):

1. Формується поточний вхідний вектор $\bar{x} = \{x_1, x_2, \ldots, x_k\}$; встановлюється покажчик адреси центроїда $j = 1$, $S = M - 1$, регістр R_N номера шуканого центроїда $N = 0$.

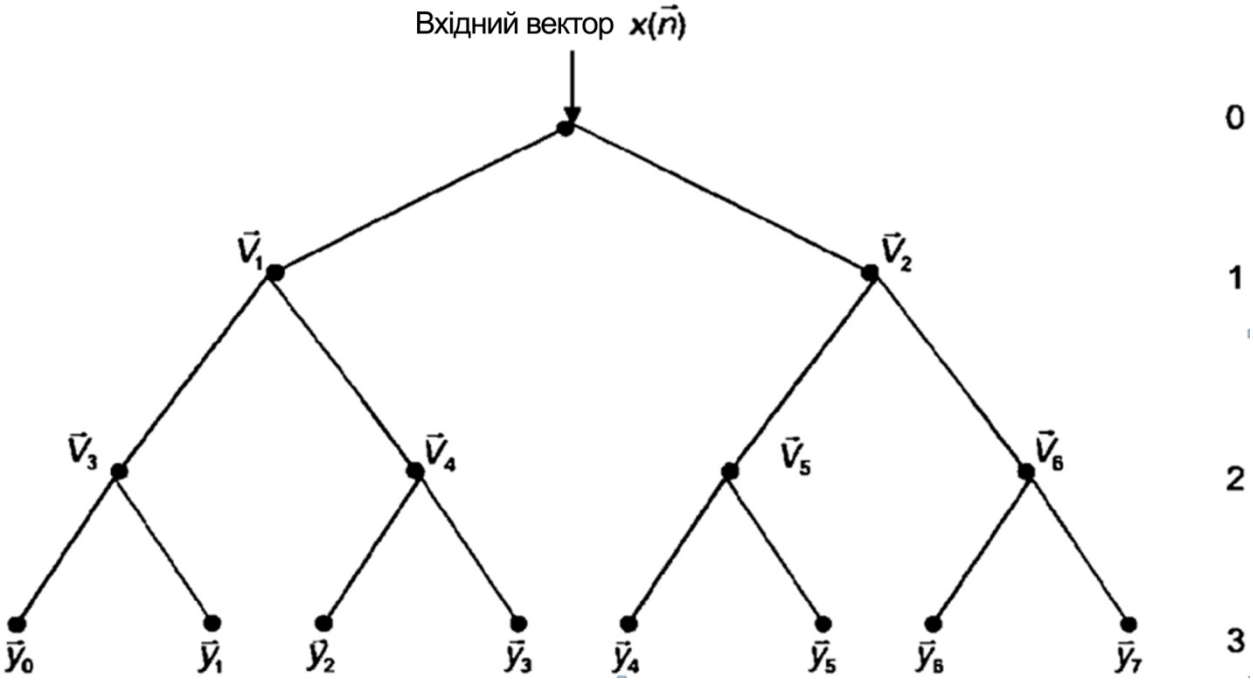

Рис. 5. Бінарне дерево (j = 1, …, 8))

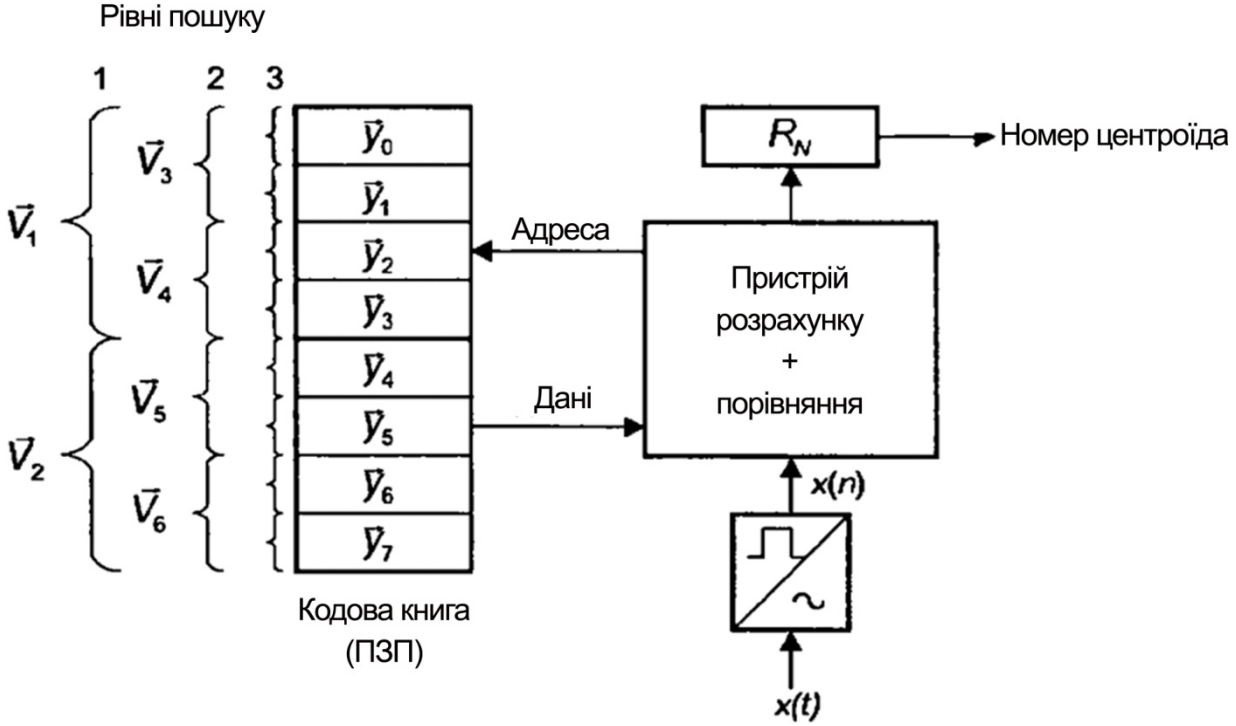

Рис. 6. Структурна схема векторного квантувальника

2. Визначаються відстані між $\bar{x}$ і центроїдами $\bar{v}_j$ і $\bar{v}_{j+1}$, наприклад по СКО:

$$d_1 = \frac{1}{k}\sum_{i=1}^{k}\left(x_i - v_{ji}\right)^2 \qquad \text{і} \qquad d_2 = \frac{1}{k}\sum_{i=1}^{k}\left(x_i - v_{(j+1)i}\right)^2 \tag{15}$$

3. Якщо $d_1 < d_2$ (рух піде по лівій гілці), адреси центроїдів дорівнюють $j = j + 2$, інакше $j = j + 3$.
4. Значення ознаки зменшується на 1: $S = S - 1$. Якщо $S \neq 0$, вміст R_N зсувається вліво на один біт; якщо $d_1 > d_2$ (рух по правій гілці), слід записати 1 в нульовий розряд регістра R_N, перейти до п. 2.

5. Якщо $S = 0$, $d_1 > d_2$, то необхідно записати одиницю в нульовий розряд і списати номер центроїда з регістра R_N.

Розглянуті приклади показують, що нелінійна обробка включає в себе алгоритми, що містять не тільки перераховані раніше операції і функції, а й більш складні процедури обробки великих масивів векторів, а також логічні операції.