The Posit arithmetic is considered a promising alternative to classical approaches. Its conceptual features make it possible to enhance numerical accuracy, reduce hardware costs, and achieve a better balance between performance and energy efficiency. This makes the Posit system a relevant subject of investigation in such fields as deep learning, embedded systems, and high-performance computing.

However, the practical implementation of Posit arithmetic at the hardware level requires in-depth analysis. The viability and scalability of this new number representation system are determined by the efficiency of hardware architectures capable of providing the required levels of accuracy, performance, and robustness against computational errors.

In the main part of this article, prepared within the framework of obtaining the degree of Doctor of Philosophy, the fundamentals of Posit arithmetic, its structure, advantages, and challenges in comparison with IEEE 754 are described.This is followed by a detailed examination of hardware implementations of Posit arithmetic units. Applications in deep learning, accuracy and efficiency analyses, robustness studies, and error investigations will be considered. The review concludes with a summary of the key findings and implications for future research.

Keywords*: Posit Number System, IEEE floating point unit, embedded systems, hardware, deep neural network.*

УДК 519.725:629.7.052 **DOI: 10.18372/2073-4751.83.20525**

Гільгурт С.Я., д.т.н.,
orcid.org/0000-0003-1647-1790
Давиденко А.М., д.т.н.,
orcid.org/0000-0001-6466-1690

ОЦІНКА МОЖЛИВОСТЕЙ СТВОРЕННЯ ЗАВАДОСТІЙКИХ КОДІВ ШЛЯХОМ ПЕРЕБОРУ КОМБІНАЦІЙ КОДОВИХ СЛІВ[1]

Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України

e-mail: hilgurt@ukr.net
e-mail: davidenkoan@gmail.com

Вступ

Під час пересилання даних комунікаційними засобами, що піддаються впливу перешкод, вникає проблема запобігання втраті інформації. Наприклад, передача радіоканалом відеозображення з бортової камери БпЛА, які останнім часом все частіше використовуються не тільки для цивільних, але й для військових застосувань, ускладнюється внаслідок впливу різних факторів. Основним джерелом перешкод є засоби радіоелектронної боротьби, причому, не тільки ворожі, але також дружні [1, 2]. Заміна радіоканалу на оптоволоконну лінію зв'язку обмежує радіус дії, ускладнює маневреність та вносить ризик втрати зв'язку внаслідок пошкодження волоконного кабелю [3]. Тому радіозв'язок досі залишається основним рішенням для забезпечення керування та передачі відеосигналу. Тому проблема підвищення завадостійкості залишається гострою. Відомі суто радіотехнічні підходи щодо боротьби з перешкодами. Наприклад, розробляються спеціалізовані радіомодеми для БпЛА, які здатні динамічно змінювати частоту та знаходити "вікна" в спектрі ворожих глуш-

[1] Дослідження виконане за фінансової підтримки з боку Національного фонду досліджень України, проєкт 2025.6/0109 "Захищена завадостійка система передачі відеоінформації з безпілотного літального апарата", номер державної реєстрації 0125U003164.

ників, або застосовують ортогональне частотне мультиплексування [4]. Іншим дієвим напрямом протидії перешкодам є завадостійке кодування інформації, яке за рахунок введення надлишковості дозволяє не тільки виявляти факт спотворення інформації, що передається, але також виправляти певну кількість помилок.

Відомо багато завадостійких кодів з відносно обмеженими характеристиками. В даній роботі розглядається ідея пошуку найкращих кодів (здатних виправляти максимальну можливу кількість помилок для заданої розмірності) шляхом перебору можливих комбінацій кодових слів [5]. Здійснювати такий пошук передбачається з використанням високопродуктивних обчислювальних засобів, таких як обчислювальні кластери та розподілені суперкомп'ютерні грід-системи [6].

Метою даної роботи є оцінка основних видів обчислювальних ресурсів, потрібних для пошуку найкращих завадостійких кодів шляхом перебору можливих комбінацій кодових слів з використанням високопродуктивного комп'ютерного обладнання.

Для її досягнення необхідно виконати наступні завдання:

1) визначитися з термінами;

2) обґрунтувати основні теоретичні положення зрозумілим чином, щоб облегшити подальше складання алгоритмів розрахунків;

3) використовуючи низку евристик, скласти основний алгоритм та. по можливості, його спрощені варіанти;

4) оцінити обчислювальну складність отриманого алгоритму та його модифікацій;

5) знайти верхню оцінку потрібних обчислювальних ресурсів двох основних видів – обсягу пам'яті та машинного часу, як функцій від параметрів розмірності завадостійкого коду.

3. Основні терміни

Одним з підходів до боротьби з цим явищем є використання спеціальних кодів, здатних виправляти помилки – *коригуючими кодами*. В *блокових* кодах інформація кодується послідовностями бітів фіксованої довжини – *словами*. Існують нелінійні та лінійні коди. Останні більш прості у використанні, тому є більш розповсюдженими. *Лінійний код* довжини n і розмірністю k – це лінійний підпростір C розмірності k над бінарним полем GF(2) і позначається як (n, k)-код.

2. Теоретичне обґрунтування

Лінійний блоковий код однозначно задається лінійним базисом бітових векторів, які разом створюють так звану *породжувальну матрицю* $\boldsymbol{G}$. Будь-яке слово (двійковий вектор) $\bar{y}$ може бути подане (закодоване) відповідним кодом (двійковим вектором) $\bar{x}$ – у вигляді лінійної комбінації базисних векторів, що у матричній формі має наступний вигляд:

$$\bar{x} = \bar{y}\, G = (y_1, y_2, y_3, \ldots, y_k) \begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,n} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,n} \\ \cdots & \cdots & \cdots & \\ g_{k,1} & g_{k,2} & \cdots & g_{k,n} \end{bmatrix}. \qquad (1)$$

Оскільки елементи результуючого вектора $\bar{x}$ також є двійковими бітами, додавання у формулі (1) при перемноженні вектора на матрицю за відомими правилами здійснюється за модулем 2.

Властивість завадостійкості коригуючого лінійного блокового коду забезпечується шляхом внесення надмірності в кодові комбінації. Це означає, що до k бітів корисної інформації додається певна кількість надлишкових бітів, за рахунок чого довжина вихідних слів лінійного коду n завжди більша за розмірність k:

$$n > k. \qquad (2)$$

Якщо надлишкові біти розташовуються в кодовому слові після інформаційних, код називається *систематичним*. Для систематичних кодів породжувальна матриця $\boldsymbol{G}$ складається з двох підматриць

– одиничної (діагональної) матриці $[I_{k,k}]$ порядку k та прямокутної матриці $[A_{k,n-k}]$,

$$G = [I_{k,k}]\,[A_{k,n-k}] = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} a_{1,k+1} & a_{1,k+2} & \dots & a_{1,n} \\ a_{2,k+1} & a_{2,k+2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{k,k+1} & a_{k,k+2} & \dots & a_{k,n} \end{bmatrix}. \quad (3)$$

яка містить k рядків і $(n - k)$ стовпців:

Відомо, що перестановка будь-яких рядків та будь-яких стовпців не змінює властивостей коду. Тому нескладно довести, що будь-який коригуючий лінійний блоковий код є функціонально рівноцінним відповідному систематичному коду [7].

Вагою w кодового слова $\bar{x}=(x_1, x_2, x_3, \dots, x_n)$ називають кількість його ненульових елементів:

$$w = \sum_{i=1}^{n} x_i. \quad (4)$$

Відстань d в лінійних кодах – це відстань Геммінга між різними кодовими словами, яка визначається кількістю відмінних бітів. Тобто відстань d за Геммінгом дорівнює вазі побітової суми за модулем 2 двох кодових слів (векторів) $\bar{x}_1$ та $\bar{x}_2$, що порівнюються:

$$d = w(\bar{x}_1 \oplus \bar{x}_2). \quad (5)$$

Найважливішою характеристикою коригуючого коду є кількість помилок e, яку цей код здатний виправити. Ця величина напряму пов'язана з мінімальною кодовою відстанню d^* коду:

$$d \geq 2 \cdot e + 1, \quad (6)$$

де мінімальна кодова відстань d^* – це найменше значення кодової відстані між будь-якими двома прийнятними значеннями кодів $\bar{x}$. Отже практичний інтерес складають лише коди, що мають ***непарну мінімальну кодову відстань*** d^*. Інакше є можливим випадок, коли сотворений код виявиться рівновіддаленим від двох прийнятних значень кодів, і його первинне значення буде неможливо відновити.

Математична подібність операцій обчислення кодової відстані та лінійної комбінації базисних векторів (в обох випадках – додавання за модулем 2), а також той факт, що нульова комбінація бітів також є можливим значенням коригуючого коду (при нульовому вхідному векторі) мають наслідком наступну важливу властивість параметра d^*: він визначається як ***мінімальна вага серед всіх прийнятних ненульових кодових слів***:

$$d = \min_{x \neq 0} w(\bar{x}). \quad (7)$$

Нагадаємо, що згідно (1) прийнятні слова коду знаходяться як добуток вхідного інформаційного вектора (всіх його комбінацій) на породжувальну матрицю. Тому значення d^* напряму залежить від змісту породжувальної матриці, а у випадку систематичного коду – від матриці $[A_{k,n-k}]$.

Оскільки цінність коригуючого коду тим вища, чим більшу кількість помилок він здатний виправити, задачу пошуку найкращого коду можна сформулювати, як пошук такої породжувальної матриці (точніше – її складової $[A_{k,n-k}]$), для якої мінімальна відстань d^* коду приймає максимально можливе значення $d_{\max}$:

$$d_{\max} = \max_{A_{k,n-k}} d\ \left(A_{k,n-k}\right). \quad (8)$$

Тобто, для наданих значень k та n з усіх можливих варіантів створення матриці $[A_{k,n-k}]$ необхідно знайти таку, що відповідає умові (8). Якщо значення максимальної кодової відстані $d_{\max}$ для неї виявиться непарним, то найкращій код знайдено. У іншому випадку приходимо до висновку, що для наданих значень кількості корисних та надлишкових бітів найкращого коригуючого лінійного блокового коду не існує.

3. Складання алгоритму

Вирішення цієї задачі повним перебором потребує виконання такого числа операцій: $k \cdot (n-k) \cdot n \cdot 2^k \cdot 2^{k\cdot(n-k)}$. Для

її спрощення та прискорення можна запропонувати декілька евристик.

Е1. Немає сенсу розглядати всі варіанти для всіх можливих значень d^*. Є сенс виконувати пошук наступним чином: для заданих n і k спробувати побудувати породжувальну матрицю для послідовних значень d^*. Найбільше значення d^*, для якого вдасться побудувати правильну матрицю, вочевидь, і є максимальною кодовою відстанню $d_{\max}$.

Е2. Оскільки невдала спроба займає менше часу, є сенс починати з найбільшого теоретично можливого значення d^*, переходячи після невдалої спроби до наступного, меншого на одиницю його значення. Перше значення d^*, для якого вдасться побудувати правильну матрицю, є максимальною кодовою відстанню $d_{\max}$,

Е3. Оскільки перестановка будь-яких рядків та будь-яких стовпців не змінює властивостей коду, для більш зручного поводження з матрицями є сенс використовувати ранжування рядків за якимось принципом, наприклад, за зростанням умовного двійкового числа S_i, складеного зі значень елементів i-го рядка матриці $a_{i,m}$ наче з двійкових розрядів:

$$S_i = S(a_{i,m}) = \sum_{m=1}^{n-k} a_{i,m} \cdot 2^{n-k-m}. \quad (9)$$

Назвемо *гладкою* породжувальну матрицю $\boldsymbol{G}$, якщо для будь-яких двох її рядків, що входять до складу матриці $[A_{k,n-k}]$ з номерами i та j, причому $i < j$, виконується співвідношення:

$$S_i < S_j. \quad (10)$$

Е4. Очевидно, що гладких кодів істотно менше ніж довільних. Тому за рахунок пошуку лише гладких матриць можна значно зменшити кількість варіантів для розглядання. Доведено, що для довільного коду (n, k) з мінімальною кодовою відстанню не нижче 3 існує подібний до нього код з гладкою породжувальною матрицею.

Нижче наведений приклад гладкої породжувальної матриці $\boldsymbol{G}$ для коду (12,7) з мінімальною кодовою відстанню 4:

$$\begin{array}{l} [1\ 0\ 0\ 0\ 0\ 0\ 0 \quad 0\ 0\ 1\ 1\ 1|\ ] \\ [0\ 1\ 0\ 0\ 0\ 0\ 0 \quad 0\ 1\ 0\ 1\ 1|\ ] \\ [0\ 0\ 1\ 0\ 0\ 0\ 0 \quad 0\ 1\ 1\ 0\ 1|\ ] \end{array} \quad (11)$$

Зауважимо, що кількість одиниць в рядку породжувальної матриці не обов'язково дорівнює значенню мінімальної кодової відстані, Але завжди – не менша за неї.

Е5. Якщо припустити, що нам відомо перші (верхні) m рядків породжувальної матриці, то шляхом побітового додавання за модулем 2 усіх їх між собою, можливо знайти 2^m прийнятних кодових слів. Якщо хоча б для однієї з усіх 2^m комбінацій $\bar{x}$не виконується умова

$$w(\bar{x}) \geq d \quad - 1, \quad (12)$$

тобто хоча б в одному кодовому слові кількість одиниць менша за d^*, то наше припущення хибне.

З евристик Е1 – Е5 природним чином випливає наступна послідовність побудови гладкої породжувальної матриці лінійного блокового коду (точніше – її правої частини $[A_{k,n-k}]$). Назвемо її ***методом нарощування***.

Починаючи з першого рядка, будуємо матрицю $[A_{k,n-k}]$, відшукуючи послідовно кожний наступний рядок таким чином: на основі попереднього m-го генеруємо гіпотетичний наступний $(m+1)$-й рядок, збільшуючи його значення S_m на одиницю і перевіряючи, чи виконується при цьому умова (12). Знаходимо всі лінійні комбінації цього гіпотетичного рядку з вже відомими попередніми, перевіряючи для кожної з комбінацій виконання умови (12). В разі, якщо умова не виконується, збільшуємо S_{m+1} ще на одиницю, і так далі, поки не буде знайдено прийнятний $(m+1)$-й рядок. Якщо не вдалося побудувати всі необхідні k рядків, повертаємось на один рядок назад, збільшуємо його значення S_{m-1} на одиницю і знов намагаємось відшукати прийнятні значення кодових слів. І так – у зворотному напрямку до

першого рядку. Якщо всі можливі припущення перебрано, приходимо до висновку, що для наданих значень n і k надане значення d^* досягнути неможливо.

З наведеного методу випливає наступний алгоритм.

Крок 1. Сформуємо перший ($m = 1$) рядок $a_{1,n-k}$ питомої гладкої породжувальної матриці $[A_{k,n-k}]$. Внаслідок вимог (8) і (10), а також того факту, що в діагональній матриці одна одиниця вже присутня, такий рядок завжди є послідовністю з $d^* - 1$ одиниць, яким передує $n - k - d^* + 1$ нулів (див. приклад (11)).

Крок 2. Формуємо m-й рядок $a_{m,n-k}$ матриці $[A_{k,n-k}]$ таким чином, щоб виконувалися умови: $S(a_{m,n-k}) > S(a_{m-1,n-k})$ та $w(a_{m,n-k}) \geq d^* - 1$. Переходимо до Кроку 4.

Крок 3. Замінюємо знайдений раніше m-й рядок $a^*_{m,n-k}$ матриці $[A_{k,n-k}]$ таким новим рядком $a_{m,n-k}$, щоб виконувалися умови: $S(a_{m,n-k}) > S(a^*_{m,n-k})$ та $w(a_{m,n-k}) \geq d^* - 1$.

Крок 4. Знаходимо всі можливі лінійні комбінації зі знайдених перших m рядків матриці $[A_{k,n-k}]$ і для кожної з них перевіряємо – чи виконується умова (12). Якщо вона виконується для всіх комбінацій, переходимо до Кроку 5. Інакше вилучаємо поточний рядок $a_{m,n-k}$ і, якщо $S(a_{m,n-k}) < 2^{n-k} - 1$, переходимо до Кроку 3 (для пошуку нового рядка зі збільшеним значенням $S(a_{m,n-k})$). Якщо ж $S(a_{m,n-k}) = 2^{n-k} - 1$ (тобто увесь рядок складається з самих одиниць), переходимо до Кроку 6.

Крок 5. Якщо $m = k$, то процес побудови породжувальної матриці для наданого значення d^* є вдалим, і алгоритм завершує свою роботу. Інакше збільшуємо значення m на одиницю та переходимо до Кроку 2.

Крок 6. Зменшуємо значення m на одиницю. Якщо отримуємо нуль, алгоритм завершує роботу з висновком, що для наданого значення кодової відстані d^* побудувати правильну матрицю неможливо. Інакше переходимо до Кроку 3.

Обчислювальна складність даного алгоритму оцінюється як $O(2^{k(n-k-1)}/k!)$, тобто є дуже близькою до складності методу повного перебору, який, як було вказано вище, потребує виконання $k \cdot (n - k) \cdot n \cdot 2^k \cdot 2^{k \cdot (n-k)}$ операцій.

Е6. Метод можна спростити (і, як наслідок, прискорити), якщо замінити в розглянутому алгоритмі Крок 6 на наступний:

Крок 6*. Алгоритм завершує роботу з висновком, що для наданого значення кодової відстані d^* побудувати правильну матрицю неможливо.

Назвемо модифікований таким чином алгоритм спрощеним, а метод, який він реалізує – ***спрощеним методом нарощування***.

Суть спрощеного алгоритму полягає в тому, що породжувальна матриця будується лише за один прохід зверху до низу без додаткових спроб змінити вже знайдені прийнятні рядки. Тобто розглядається лише один правильний варіант гладкої породжувальної матриці – перший, "як склалося".

Звісно, спрощений алгоритм не гарантує, що перевірене ним значення кодової відстані d^* є найбільшим. Тобто отримане за його допомогою значення кодової відстані, строго кажучи, є не максимальною кодовою відстанню $d_{\max}$, а лише його нижньою оцінкою $d^{**}_{\max}$. Але, як свідчать експерименти, в багатьох випадках ця оцінка $d^{**}_{\max}$ збігається з величиною $d_{\max}$, а якщо ні, то ніколи не відрізняється більш, ніж на одиницю. Проте оцінка обчислювальної складності спрощеного методу має вже значно нижчий порядок – $O(2^n)$.

Е7. Ще більше прискорити обчислення можливо, якщо всі комбінації, перевірені на Кроці 4, запам'ятовувати і використовувати для наступних значень m. Це призведе до збільшення потрібної пам'яті, суттєвого – для великих значень k, але дозволить зменшити кількість потрібних операцій, хоча й не усуне ступеневий порядок оцінки обчислювальної складності $O(2^n)$.

Таблиця 1.
Пояснення щодо евристики Е7

1	1 рядок	2^0
2	2 рядок	2^1
3	1+2	
4	3 рядок	2^2
5	4+1	
6	4+2	
7	4+3	
8	4 рядок	2^3
9	8+1	
10	8+2	
11	8+3	
12	8+4	
13	8+5	
14	8+6	
15	8+7	
16	5 рядок	2^4
17	16+1	
18	16+2	
2^γ	.....	2^γ

Таблиця 1 ілюструє реалізацію цієї евристики. Наприклад, нам потрібно перевірити 4-й рядок матриці $[A_{k,n-k}]$, тобто $m = 4$ (8-й рядок у таблиці, в загальному випадку – 2^{m-1}). Виконуючи порозрядне додавання за модулем 2 змісту 4-го рядка послідовно зі всіма попередніми вже порахованими рядками допоміжної таблиці з 1-го по 7-й, розміщуємо отримані значення у рядках з номерами з 9-го (8+1) по 15-й (8+7) (в загальному випадку – з $(2^{m-1}+1)$-го по (2^m-1)-й рядок). Якщо в кожному з обчислених рядків число одиниць виявиться не меншим за питому кодову відстань, 4-й рядок вважається прийнятним та записується в питому матрицю $[A_{k,n-k}]$, а рядки з 9-го по 15-й допоміжної таблиці зберігаються для подальшого пошуку 5-го та наступних рядків матриці. В іншому випадку генерується наступне значення 4-го рядку матриці згідно Кроку 2 або Кроку 3, і перевірка повторюється.

Як видно з наведеного прикладу, розмір допоміжної таблиці складає $(2^k - 1)$ рядків. Але наступна евристика дозволяє скоротити це значення удвічі.

Е8. Оскільки під час пошуку останнього рядку матриці $[A_{k,n-k}]$ результати комбінування поточного рядка з усіма попередніми в подальшому не знадобляться, для даного значення $m = k$ заповнювати допоміжну таблицю не потрібно. Тоді розмір допоміжної таблиці складатиме $(2^{k-1} - 1)\ 2^{k-1}$ рядків.

4. Оцінка потрібних обчислювальних ресурсів

До основних видів обчислювальних ресурсів, витрати яких потрібно оцінити в першу чергу, відносяться обсяг пам'яті та машинний час.

4.1. Оцінка потрібної пам'яті

Оцінімо витрати на пам'ять для програми, яка функціонує згідно алгоритму, що використовує Е7 та Е8.

Якщо в програмі для зберігання рядків породжувальних матриць використовуються 32-бітні (4-байтні) змінні (така програма здатна обчислювати матриці для яких параметр $(n - k)$ може сягати значень до 32), об'єм пам'яті потрібної для зберігання допоміжної таблиці M_k можна оцінити знизу виразом, який залежить лише від параметру k:

$$M_k = M(k) > 2^2 \cdot 2^{k-1}[\text{байтів}] = 2^{k+1}\ [\text{б}]$$
$$= 2^{k-9}\ [\text{Кб}] = 2^{k-19}\ [\text{Мб}] = 2^{k-29}\ [\text{Гб}]. \qquad (13)$$

Оскільки зберігання інших змінних потребує суттєво менше пам'яті, їх загальним об'ємом можна знехтувати.

Отже, згідно (13) для пошуку породжувальної матриці, наприклад, для (64, 32)-коду програмі буде потрібно пам'яті трохи більше за 2^{32-29} Гб = 2^3 Гб = 8 Гб.

4.2. Оцінка потрібного часу обчислень

Оцінити обчислювальний час, потрібний для пошуку породжувальних мат-

риць не є тривіальною задачею. Крім численних евристик, використаних для скорочення цього часу, а також технік та прийомів, застосованих при оптимізації коду, на час обчислень також суттєво впливає використання кешу 3-го рівня центрального процесора, на якому виконуватиметься програма. Тому для приблизної оцінки цієї величини пропонується використовувати змішаний емпірично-теоретичній підхід.

Аналіз експериментальних даних свідчить, що чинник випадковості, який суттєво впливає на фактичний час обчислень, більш помітний при зміні або лише параметру k, або лише $n - k$. При одночасному збільшенні обох цих параметрів, тобто, при збільшенні параметра n, випадковий характер залежності менш впливовий. Тому пропонується побудувати апроксимаційну залежність як функцію від однієї цієї змінної – n. Оскільки оцінка обчислювальної складності використаного в програмі спрощеного методу має порядок $O(2^n)$, спробуємо апроксимувати зверху емпіричні дані за допомогою виразу:

$$T_n = T(n) < B \cdot 2^n , \qquad (14)$$

де T_n – час розрахунку коду довжиною n, B – константа, яку необхідно підібрати емпіричним шляхом.

Висновки

В роботі розглянуто алгоритм пошуку найкращого коригуючого лінійного блокового (n, k)-коду для заданих параметрів n і k шляхом перебору комбінацій, придатний для виконання на високопродуктивних обчислювальних системах. Запропоновано спрощений метод нарощування, який дозволяє збільшити наочність та суттєво підвищити швидкодію алгоритму, при втраті точності не гірше за одиницю значення максимальної кодової відстані. Наведено приблизні оцінки пам'яті та часу обчислень, потрібних для виконання алгоритму, які будуть корисними під час подальшої реалізації алгоритму на високопродуктивних обчислювальних засобах.

Література

1. Способи протидії FPV-дронам (з ворожого довідника). Статті компанії «FPV-дрони та аксесуари для аеророзвідки». *"FPV-дрони та аксесуари для аеророзвідки" - контакти, товари, послуги, ціни*. URL: https://drone.in.ua/a/506352-sposobi-protidiyi-fpv.html (дата звернення: 21.11.2025).

2. Гліб. Чому дрони не падають під РЕБом. *Повернись живим*. URL: https://savelife.in.ua/materials/blogs/homu-drony-ne-padayut-pid-rebom (дата звернення: 21.11.2025).

3. Focus: emergence of a new family of FPV drones. *Defense News security global military army equipment industry*. URL: https://armyrecognition.com/news/aerospace-news/2024/focus-why-jamming-isnt-a-solution-against-fpv-anymore-in-ukraines-offensive-initiative (date of access: 21.11.2025).

4. Drone defense system : пат. US10339818B2 (USA) : G08G5/00, H04W76/10. № 15/358,574 ; заявл. 22.11.2016 ; опубл. 02.07.2019. 37 с. URL: https://patentimages.storage.googleapis.com/96/9b/c4/04589f6c523d45/US10339818.pdf (дата звернення: 21.11.2025).

5. Експериментальна перевірка спрощеного алгоритму пошуку лінійних блокових кодів / А. Давиденко та ін. *Кібербезпека енергетики* : Матеріали наук.-практ. конф. Ін-ту проблем моделювання в енергетиці ім. Г.Є. Пухова. НАН України, м. Київ, 31 трав. 2023 р. Київ, 2023. С. 54–58.

6. Применение грид-системы при исследовании линейных блоковых кодов / С. Винничук та ін. *Системи обробки інформації* : Зб. наук. пр., м. Харків. Харків, 2013. С. 61–64.

7. Blahut R. Theory and practice of error control codes. Addison-Wesley Publishing Company, 1983. 500 p.

Гільгурт С.Я., Давиденко А.М.

ОЦІНКА МОЖЛИВОСТЕЙ СТВОРЕННЯ ЗАВАДОСТІЙКИХ КОДІВ ШЛЯХОМ ПЕРЕБОРУ КОМБІНАЦІЙ КОДОВИХ СЛІВ

Робота присвячена питанням пошуку найкращих кодів, здатних виправляти максимальну можливу кількість помилок для заданої розмірності. Здійснювати такий пошук пропонується шляхом перебору можливих комбінацій кодових слів з використанням високопродуктивного обчислювального обладнання. Обґрунтовано основні теоретичні положення. Запропоновано повний та спрощений алгоритми, що реалізують метод нарощування. При цьому спрощений алгоритм дозволяє збільшити наочність та суттєво підвищити швидкодію при втраті точності не гірше за одиницю значення максимальної кодової відстані. Наведено приблизні оцінки пам'яті та часу обчислень, потрібних для виконання алгоритму, які будуть корисними під час подальшої реалізації алгоритму на високопродуктивних обчислювальних засобах. Знайдено верхню оцінку потрібних обчислювальних ресурсів двох основних видів – обсягу пам'яті та машинного часу, як функцій від параметрів розмірності завадостійкого коду.

Ключові слова: *БпЛА, передача відеоданих, коригуючий код, перебір комбінацій, HPC, оцінювання, об'єм пам'яті, машинний час.*

Hilgurt S.Ya., Davydenko A.M.

EVALUATION OF THE POSSIBILITIES OF CREATING INTERFERENCE-RESISTANT CODES BY SELECTING CODE WORD COMBINATIONS

The work is devoted to the issues of finding the best codes capable of correcting the maximum possible number of errors for a given dimension. It is proposed to carry out such work by searching for possible combinations of code words using high-performance computing equipment. The main theoretical provisions are substantiated. Full and simplified algorithms are proposed that implement the method of building up, which allows improving the clarity and to significantly increase the speed of the algorithm, while losing accuracy not worse than the unit value of the maximum code distance. Approximate estimates of the memory and computation time required for the execution of the algorithm are given, which will be useful during the further implementation of the algorithm on high-performance computing devices. An upper estimate of the required computational resources of two main types is found – the amount of memory and machine time, as functions of the dimensionality parameters of the noise-resistant code.

Keywords: *UAV, video data transmission, error control code, combinations selection, HPC, estimation, memory size, wall time.*