Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _ інформаційних радіотехнологій та технічного захисту інформації_
(повна назва)

Кафедра __ медіаінженерії та інформаційних радіоелектронних систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

__Дослідження методів та алгоритмів завадостійкого кодування та їх застосування в твердотільних накопичувачах_____
(тема)

Виконав:
студент __2__ курсу, групи ____СТМм-22-2

_____Кременецький В.В._____
(прізвище, ініціали)

Спеціальність 171 Електроніка_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна____
Освітня програма Системи, технології і компютерні засоби мультимедіа_____
(повна назва освітньої програми)

Керівник _____доц. Ликов Ю.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ ____Володимир КАРТАШОВ_
(підпис)

2024 р.

Харківський національний університет радіоелектроніки

Факультет інформаційних радіотехнологій та технічного захисту інформації
Кафедра медіаінженерії та інформаційних радіоелектронних систем

Рівень вищої освіти другий (магістерський)

Спеціальність 171 Електроніка
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Системи, технології і компютерні засоби мультимедіа
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кременецькому В’ячеславу В’ячеславовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Дослідження методів та алгоритмів завадостійкого кодування та їх застосування в твердотільних накопичувачах
затверджена наказом по університету від "_20_"_____11_____ 2023 р. № ____1371 Ст_____

2. Термін подання студентом роботи до екзаменаційної комісії _____08.01.2024 р._____

3. Вихідні дані до роботи 1. Код Хеммінга. 2. Код БЧХ. 3. код Ріда-Соломона. 4. код LDCP. 5. довжина кодового слова та ймовірність помилки (7,4), (15,11), (31,26), (63,57), (255,187), (255, 247), (511,250) та (511,502)

4. Перелік питань, що потрібно опрацювати в роботі _____
Вступ
1. Аналіз методів завадостійкого кодування.
2. Принципи моделювання алгоритмів завадостійкого кодування та програмне забезпечення.
3. Експеримент з аналізу ефективності алгоритмів завадостійкого кодування.
Висновки
Перелік посилань
Додатки

5. Перелік графічного матеріалу із зазначенням обов’язкових креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Класифікація завадостійких кодів (1 аркуш А4).
2. Твердотільний накопичувач (1 аркуш А4).
3. Схема імітаційного моделювання (1 аркуш А4).
4. Схеми завадостійких кодів реалізованих в Simulink (1 аркуш А4).
5. Графіки коду з малою щільністю перевірок на парність (1 аркуш А4).
6. Висновки (1 аркуш А4).

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналітичний огляд літератури	01.09.23–13.09.23	
2	Алгоритми завадостійкого кодування	14.09.23–27.09.23	
3	Програмне забезпечення	28.09.23–11.10.23	
4	Експеримент з аналізу ефективності алгоритмів	12.10.23–25.10.23	
5	Обробка результатів	26.10.23–10.11.23	
6	Графічна частина роботи	11.11.23–25.11.23	
7	Перевірка керівником	26.12.23–02.01.24	
8	Перевірка на академічний плагіат	03.01.24	
9	Перевірка завідувачем кафедри, рецензування	04.01.24–15.01.24	

Дата видачі завдання 20.11.2023 р.

Студент _____ _____ В’ячеслав КРЕМЕНЕЦЬКИЙ
(підпис)

Керівник роботи _____ Юрій ЛИКОВ
(підпис)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 97 сторінок, 63 рисунків, 2 таблиці, 26 джерел.

ДОСЛІДЖЕННЯ, ЗАВАДОСТІЙКІСТЬ, ТВЕРДОТІЛЬНИЙ НАКОПИЧУВАЧ

Об’єктом дослідження даної дипломної роботи є завадостійке кодування.

Мета роботи – Аналіз та дослідження завадостійких кодів, що застосовують у твердотільних накопичувачах. Проведення експерименту з ефективності.

Метод дослідження – теоретичний аналіз, проведення практичних вимірювань, експериментальні дослідження.

Проведено теоретичний аналіз з завадостійкості кодування, та розглянуті твердотільні накопичувачі, а також алгоритми завадостійкого кодування що в них застосовуються. Для проведення експерименту з аналізу ефективності алгоритмів завадостійкого кодування було обрано систему комп’ютерної математики MATLAB.

В результаті проведення експерименту з аналізу ефективності алгоритмів завадостійкого кодування на прикладі кода Хеммінга з різною розмірністю було вставлено, що при збільшенні розмірності кода ріст кількості помилок з ростом ймовірність їх появи становиться більш плавним. Було виявлено, що найбільш завадостійкими є коди Ріда-Соломона та LDPC.

ABSTRACT

Explanatory note to the performance appraisal: 97 pages, 63 figures, 2 tables, 26 sources.

RESERCH, ERROR CORRECTION,SOLID-STATE DRIVE

THE OBJECT OF RESEARCH OF THIS GRADUATE WORK IS ERROR DETECTION AND CORRECTION.

OBJECTIVE – ANALYSIS AND RESEARCH OF ERROR CORRECTION CODES USED IN SOLID STATE DRIVES. CARRYING OUT THE EXPERIMENT ON EFFICIENCY.

Research method – theoretical analysis, practical measurements, experimental studies.

A theoretical analysis of coding immunity was carried out, and solid-state storage devices were considered, as well as the algorithms of interference-resistant coding that are used in them. The MATLAB computer mathematics system was chosen to conduct the experiment on the analysis of the effectiveness of noise-resistant coding algorithms.

As a result of conducting an experiment on the analysis of the effectiveness of noise-resistant coding algorithms on the example of a Hamming code with different dimensions, it was found that with an increase in the size of the code, the number of errors increases with the increase in the probability of their occurrence. It was found that Reed-Solomon and LDPC codes are the most resistant to interference.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

ДСК – двійкового симетричного каналу;

ЗК – Згорткові коди;

Код БЧХ – код Боуза - Чоудхурі - Хоквінгема;

ОЗК – Ортогональними згортковий код;

ПЗ – програмного забезпечення ;

СЗК – Самоортогональні згортковий код;

ECC – error-correcting code memory;

LDPC-code – Low-density parity-check code;

NAND – not and;

QSBC – Quadruple Swing-By Code;

RG – register;

SSD – solid-state drive.

ВСТУП

Твердотільний накопичувач повинен підтримувати функцію виявлення помилок, щоб підвищити надійність даних. Оскільки генерація флеш-пам'яті NAND просувається до більш вузького визначення правила процесу, технологія виявлення даних і виправлення помилок (ECC) стала більш важливою для SSD. ECC, які використовуються для флеш-пам'яті NAND, - це код Хеммінга, код Ріда-Соломона, код БЧХ (код Боуза – Чоудхурі – Хоквінгема), LDPC (перевірка парності з низькою щільністю). Код Хеммінга часто використовувався в ранньому поколінні флеш-пам'яті NAND. Розрахунок коду Хеммінга простий і часто реалізується програмним забезпеченням. Код Ріда-Соломона обробляє кілька бітів як один символ, а виправлення помилок виконується в одиницях символів. Корекція помилок, що виконується в одиницях символів, підходить для виправлення помилок пакетного біта. Код БЧХ має гнучкість щодо довжини блоку і можливості виправлення помилок. Їх споживана потужність мала по порівнянню з можливістю виправлення помилок. Код БЧХ - найпопулярніший ECC в наші дні. Можливість корекції помилок LDPC надзвичайно висока. Але необхідні міркування для споживання енергії і часу обробки. Щоб вирішити компроміс, Toshiba розробила оригінальну методику виявлення і корекції помилок QSBC ™ і використовувала її в продуктах SSD.

Основне завдання дипломної роботи – дослідження алгоритмів завадостійкого кодування, Їх порівняння та аналіз ефективності.

Актуальність даного проекту – є пошук рішень для оптимізації різних типів алгоритмів під різні задачі. Вивчення слабких та сильних сторін кожного з алгоритмів. Та практичне застосування результатів.

В роботі проаналізовано що таке завадостійке кодування, які існують його види, було розглянуте ,що таке твердотільні накопичувачі, а також алгоритми завадостійкого кодування що в них застосовуються. Розглянуто основні види програмного забезпечення (ПЗ) ,яке дозволяє реалізувати

завадостійке кодування, описано декілька систем комп’ютерної математики, також було розглянуто схему імітаційного моделювання та її складові частини. Проведено аналіз ефективності різних алгоритмів завадостійкого кодування.

1 АНАЛІЗ МЕТОДІВ ЗАВАДОСТІЙКОГО КОДУВАННЯ

1.1 Методи завадостійкого кодування

На сьогоднішній день відомо багато різних класів завадостійких кодів (рис.1.1) [1], що відрізняються один від одного структурою, функціональним призначенням, надмірністю, енергетичною ефективністю, алгоритмами кодування і декодування, способом передачі кодових символів і т.д [1].

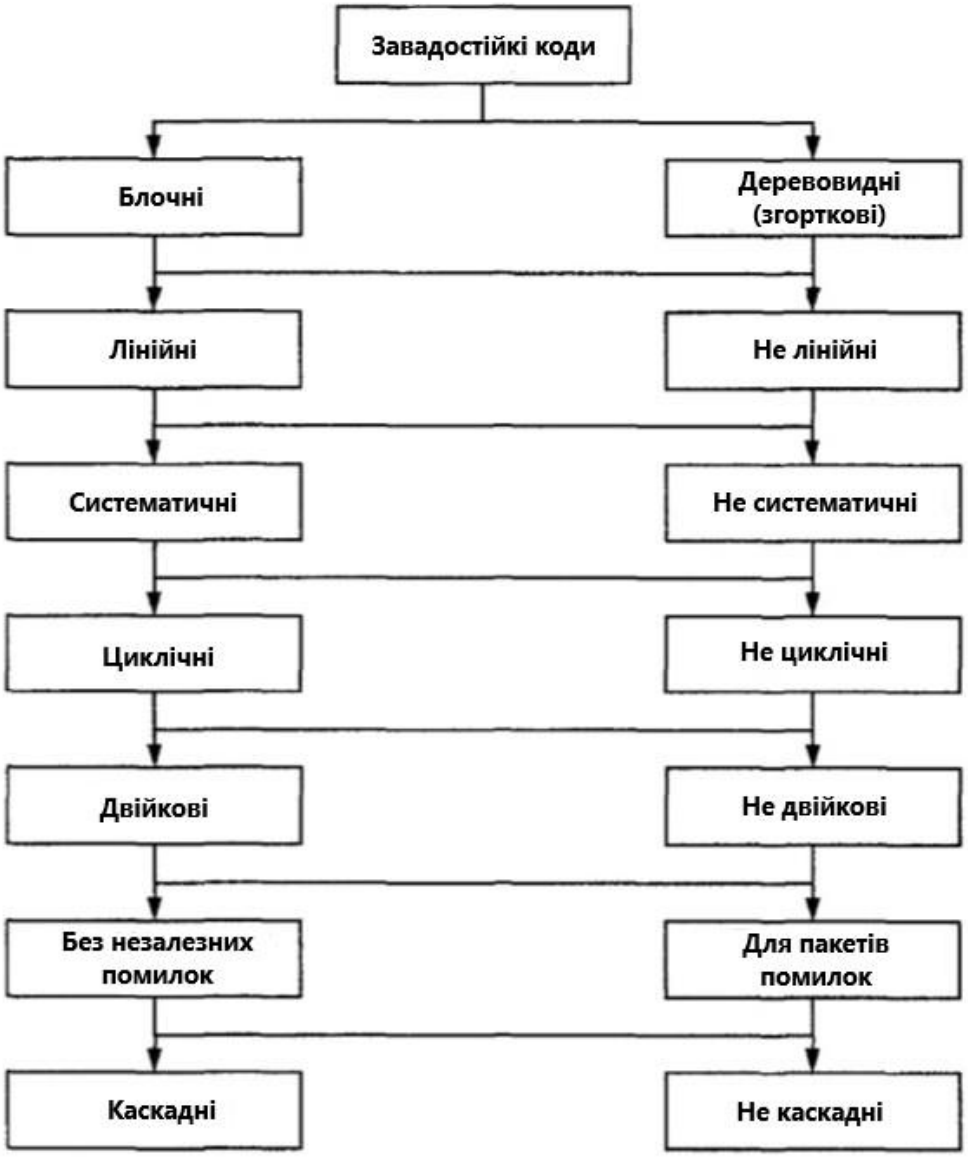

Рисунок 1.1 – Класифікація завадостійких кодів

Нарівні з цими є деякі підходи класифікації завадостійких кодів.

При одному підході виділяють два принципово різних типи кодів. У перших безперервна послідовність інформаційних символів розбивається на відрізки або блоки, що містять по k символів. Такі коди називається блоковими. По-друге інформаційна послідовність піддається обробці без попереднього розбиття її на незалежні блоки. Такі коди називаються деревоподібними. Згорткові коди є окремим випадком і важливим підкласом деревовидних кодів. Надалі розглядаються тільки згорткові деревовидні коди, звані просто згортковими кодами. В іншому підході завадостійкі коди можна розбити на коди, що виправляють випадкові або незалежні помилки, і коди, що виправляють пакети помилок. В основному, на практиці, застосовуються коди, що виправляють випадкові помилки, оскільки для виправлення пакетів помилок часто виявляється легше використовувати коди для виправлення незалежних помилок разом з пристроями перемежения і відновлення [1].

Майже всі відомі блокові і деревовидні коди є деяким підкласом лінійних кодів. Лінійні коди задаються в векторному просторі, що сильно спрощує процедури кодування та декодування. Найбільш важлива властивість даних кодів: сума двох кодових комбінації також є кодовим словом. Через це майже всі схеми кодування, що застосовуються на практиці, засновані на лінійних кодах [1].

1.1.1 Характеристики завадостійкого кодування

Існує і використовується багато визначень коду і завадостійкому кодуванню. Наприклад, код є форма подання інформаційного повідомлення, яка не залежить від його фізичної сутності або код, представляється (виражається) сукупністю кодових символів, між якими штучно введені кореляційні зв'язки або код є кінцева безліч кодових послідовностей (кодових слів), побудованих за одним і тим же алгоритмом або правилом. Завадостійке кодування являє собою процес перетворення переданих інформаційних

символів за певним алгоритмом або за певними правилами, і в результаті чого формуються послідовності кодових символів, що відображають передані інформаційні повідомлення. Загалом завадостійке кодування грунтується на введенні надлишкової інформації в інформаційні повідомлення,що передаються. В результаті завадостійкого кодування формуються кодові послідовності (кодові слова, кодограми) , що відображають передані інформаційні повідомлення, які умовно поділяються на дозволені і заборонені і за якими потім за певними алгоритмами виявляються і коректуються помилкові інформаційні символи [2].

Основними характеристиками завадостійких кодів є:

1. Основа коду (q) - визначається числом (кількістю) різних одиничних елементів (символів), які використовуються при побудові коду і кодових послідовностей. Основа коду найчастіше позначається через q і при цьому: l = (n - k) r = (n - k)/ n = (l / n) *100%.

Якщо q = 2, тобто використовуються двійкові символи 1 і 0, то такі коди називаються двійковими і кодування проводиться в двійковому полі Галуа, тобто GF (q) = GF (2). Якщо q>2, наприклад, q=2^m , $m^3$2, о такі завадостійкі коди називаються недвійковий і кодування інформації проводиться в недвійковий полі Галуа, тобто GF(q^m) = GF(2^m).

2. Довжина кодової послідовності (n) - дорівнює кількості двійкових символів (біт) в даній кодовії послідовності. Довжину кодової послідовності прийнято позначати через n : мінімальна довжина кодової послідовності n = 2 двійковим символам (нижня межа); верхня межа n в принципі може досягати тисячі і десятки тисяч двійкових символів.

3. Кількість інформаційних символів (k) - це кількість двійкових символів ,що несуть корисну інформацію; кількість інформаційних символів в кодовії послідовності прийнято позначати через k і їх число може бути рівним k = 1, 2, .. і т.д. двійкових символів.

4. Швидкість передачі коду R=k/n - характеризує кількість надлишкових символів , що припадають на один інформаційний символ. Чим

більше R, тобто R → 1 (R завжди менше 1), тим ефективніше завадостійкий код, так як передається менше надлишкової інформаціїэ.

- Надлишковість завадостійкого коду ділиться на: абсолютну надлишковість $l = (n - k)$ двійкових символів; відносну надлишковість $r = \frac{n-k}{n} = \left(\frac{l}{n}\right) * 100\%$.

5. Вага кодової послідовності ($w_{\text{к.посл.}}$) – визначаться кількістю ненульових символів, що входять в кодову послідовність. Наприклад, нехай n = 10 двійкових символів, а дві кодові послідовності даного завадостійкого коду мають наступну структуру: $F_1(x) = 1001110010$ та $F_2(x) = 0010101100$, то вага $F_1(x)\ w_1 = 5, a\ F_2(x) -\ w_2 = 4$.

-Класифікація кодових послідовностей: якщо задані або відомі такі характеристики завадостійкого коду як n і k, то можна визначити: $K_{\text{заг}} = 2^n$ - загальна кількість кодових послідовностей; $K_{\text{доз}} = 2^k$ - кількість дозволених кодових послідовностей; $K_{\text{заб}} = 2^l$ - кількість заборонених кодових послідовностей. В принципі із загальної кількості $K_{\text{заг}} = 2^n$ кодових послідовностей деякі кодові послідовності матимуть однакову вагу, а їх загальне число буде визначати кількість помилок найменшію кратності, що викликають помилки, не визначаються цим кодом [2].

6. Кодова відстань (d) - дорівнює кількості позицій, якими відрізняються дві порівнювані кодові послідовності. d_0 – мінімальна кодова відстань (відстань Хеммінга). Якщо загальна кількість кодових комбінацій $К_{заг}$=2n, то загальне число кодових відстаней може бути більш ніж 2n/2. Для згорткових кодів d_0=J+1, де J – кількість ртогональних перевірок, необхідних для корекції її $t_{випр}$ помилок.

7. Кратність (кількість) виправляємих і виявлених помилок.

Якщо деякий завадостійкий код з параметрами q=2, n і k використовується для передачі інформації, то $К_р=2^k$ кодових послідовностей довжини n двійкових символів утворюють інформаційну частину кодового

простору або безлічі, а решта, $К_з=К_л=2^{n-k}$ - творюють помилкову частину кодової безлічі, що умовно можна уявити так (рис.1.2) [2].

Рисунок 1.2 – Підпростори кількості кодових послідовностей (n,k) - кода

Розподіл кодової безлічі на інформаційну і хибну частини дозволяє виявляти помилки шляхом порівняння прийнятої кодової послідовності з кодовими послідовностями обох частин; якщо кодова послідовність виявляється в хибній частині, то вважається, що прийнята кодова послідовність є помилковою, а в іншому випадку - істинною чи без помилок; або помилки просто не виявлені кодом [2].

Для того щоб виправити помилки, повну кодову безліч розбивають на 2^k інформаційних частин, взаємно однозначно відповідних інформаційних послідовностей, що умовно можна уявити так (рис.1.3) [2].

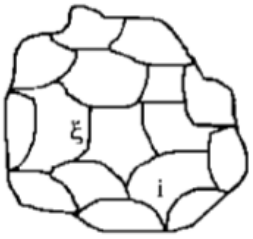

Рисунок 1.3 – Взаємно однозначні відповідні інформаційні послідовності

Прийняте кодове слово при порівнянні з кодовою безліччю може опинитися в одній з його частин: якщо воно буде в частині, що відповідає цьому слову, наприклад, ξ, о одержувачу інформації видається справжнє слово, а якщо воно потрапить в будь-яку іншу його частину, наприклад, і, о

слово вважається помилковим і його треба виправити, тобто скорегувати помилку [2].

Якщо ж в цій кодовій безлічі виділити три частини (груп) кодових послідовностей, наприклад, ξ (помилок немає), *i* (є кореговані помилки) та γ (помилки, які не коригуються) (рис.1.4) [2], то в цьому випадку можна виявляти кодові послідовності з помилками, які не коригуються.

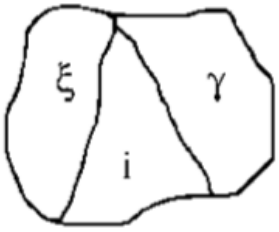

Рисунок 1.4 – Умовне уявлення підпросторів кодових послідовностей

Хеммінг довів, що коригувальні властивості завадостійкого коду визначаються його мінімальною кодовою відстанню. Хеммінга довів наступне: якщо дві кодові послідовності відрізняються один від одного в t (t≥1) позиціях (розрядах, символах), а від всіх інших кодових послідовностей цієї кодової безлічі будуть відрізнятися більш, ніж в t позиціях, то для корекції t помилок необхідно щоб мінімальна кодова відстань повинна бути $d_0 \geq 2*t+1(2)$.

Отже, для того щоб завадостійкий код виявив і виправив t помилок необхідно, щоб мінімальна кількість позицій, якими б відрізнялися будь-які дві кодові послідовності розглянутого завадостійкого коду становило б $d0 \geq 2 * t + 1$ (2) позицій.

Якщо виконується така умова, то завадостійкий код може виправити $t_{випр} \leq \frac{d_0 - 1(2)}{2}$ помилкових символів, або виявити $t_{вияв} \leq d_0 - 1$ помилкових двійкових символів; число 2 віднімається при парному значенні d_0.

Сказане останнім можна проілюструвати наступними рисунками (рис.1.5) [2].

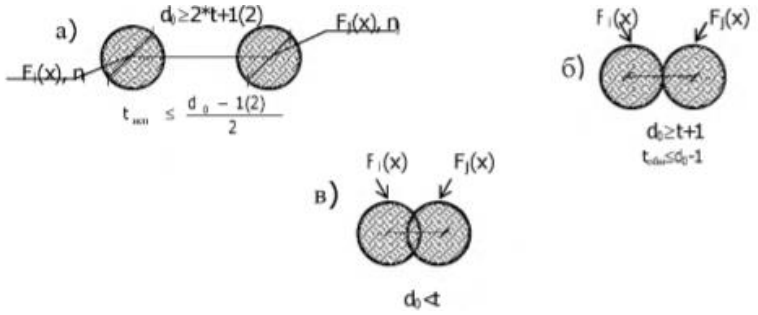

Рисунок 1.5 – Умовне розташування кодових послідовностей для різних значень d_0

Для першого випадку (рис.1.5 а) розташування кодових послідовностей випливає, що якщо кодова послідовність $F_i(x)$ відстоїть від кодової послідовності $F_j(x)$ на відстані d0 військових символів, то це означає, що в кодової послідовності $F_i(x)$ необхідно замінити d0 кодових символів на зворотні, щоб отримати кодову послідовність $F_j(x)$, яка заміняла собою б кодову послідовність $F_i(x)$ [2].

Якщо ж дві кодові послідовності будуть знаходитися один від одного на меншій відстані ніж d_0, тобто вони матимуть більше ніж d_0 співпадаючих (однакових) кодових символів, то декодер не виявить і не виправить помилки в прийнятій кодової послідовності $F_i(x)$; випадок (рис.1.5 в).

Якщо ж дві кодові комбінації будуть перебувати на відстані $d_0 \geq t+1$, то декодер може тільки виявити, що прийнята кодова послідовність є помилковою випадок (рис.1.5 б).

Для того, щоб завадостійкий код виправляв $t_{випр}$ помилок і виявляв $t_{вияв}$ помилок необхідно, щоб $d0 \geq t_{випр} + t_{вияв} + 1$.

Розглянувши прийняті умовні позначення основних характеристик завадостійкого коду і їх сутність, можна записати загальне позначення блокового завадостійкого коду через його основні характеристики, наприклад, так (n, k, d_0) = (15,7,5).

8. Імовірність помилкового декодування групового лінійного коду істотно залежить від типу коду, його параметрів і типу каналу зв'язку. Для

двійкового симетричного каналу зв'язку без пам'яті ймовірність визначається за формулою

$$P_{ОШ} = \sum_{i=t_{исп}+1}^{n} C_n^i P_k^i \, q^{n-i}. \quad (1.1)$$

1.1.2 Блочні коди

Блоковий код в інформатиці – тип канального кодування. Він збільшує надмірність повідомлення так, щоб в приймачі можна було розшифрувати його з мінімальною (теоретично нульовою) похибкою, за умови, що швидкість передачі інформації (кількість передавальної інформації в бітах в секунду) не перевищила б канальну продуктивність [6].

Головна характеристика блокового коду полягає в тому, що це – канальний код фіксованої довжини (на відміну від такої схеми кодування джерела даних, як код Хаффмана, і у відмінність таких методів канального кодування, як конволюційне кодування («згортальне» кодування)). Зазвичай, система блокового кодування отримує на вході k-знакове кодове слово w, і перетворює його в n-знакове кодове слово c(W). Це кодове слово і називається блоком [6].

Блокове кодування було головним типом кодування, використовуваного в ранніх системах мобільної комунікації.

Нехай інформація, яка кодується ділиться на фрагменти довжиною k біт, які перетворюються в кодові слова довжиною n біт. Тоді відповідний блоковий код зазвичай позначають (n,k). При цьому число $R = \frac{k}{n}$ називається швидкістю коду.

Якщо вихідні k біт код залишає незмінними, і додає n-k перевірочних, такий код називається систематичним, інакше несистематическим.

Задати блоковий код можна по-різному, в тому числі таблицею, де кожній сукупності k інформаційних біт зіставляється n біт кодового слова.

Однак, гарний код повинен задовольняти, як мінімум, наступними критеріями:

- здатність виправляти якомога більше число помилок;
- якомога менша надмірність;
- простота кодування і декодування.

Неважко побачити, що наведені вимоги суперечать один одному. Саме тому існує велика кількість кодів, кожен з яких придатний для свого кола завдань.

Практично всі коди,що використовуються є лінійними. Це пов'язано з тим, що нелінійні коди значно складніше досліджувати, і для них важко забезпечити задовільну легкість кодування і декодування.

1.1.3 Лінійні коди

Лінійним блоковим кодом називається такий завадостійкий код, у якого перевірочні символи формуються шляхом підсумовування по модулю два інформаційних символів, що стоять (розташованих) на певних позиціях (місцях), а сума двох кодових послідовностей і добуток кодової послідовності на елемент поля утворюють також кодові послідовності [3].

Основними властивостями виконавчих роздільних систематичних лінійних блокових кодів є:

-лінійність коду визначається спеціально обраною структурою коду. Лінійність коду істотно спрощує процедуру кодування і декодування, дозволяючи висловити кожну кодову послідовність у вигляді лінійної комбінації невеликого числа виділених кодових послідовностей, так званих базисних векторів;-сумма по модулю два двух кодовых последовательностей также является кодовой последовательностью;

-лінійний блоковий код завжди містить кодову послідовність, що складається цілком з нулів;

- якщо скласти по модулю два деяку кодову послідовність з усіма кодовими послідовностями, то знову вийде безліч всіх кодових послідовностей, розташованих, можливо, в іншому порядку;

- -вага кодової последовательності ($\omega_{кп}$) завжди повинна бути $\geq d_0$;

- вага перевірочної частини кодової послідовності ($\omega_{пер.кп}$) завжди повинна бути $\geq (d_0 - 1)$;

- вага суми по модулю два двох дозволених кодових послідовностей ($\omega_{\sum кп}$) завжди повинна бути $\geq (d_0-1)$,але дозволяється $\geq (d_0-2)$;

- груповий двійковий лінійний блоковий код повністю задається як породжуюча $G_{k.n}(x)$, так і перевірочна $H_{l,n}(x)$ матриці.

Найменшу складність побудови і реалізації мають роздільні систематичні виконавчі лінійні блокові коди. При використанні даних кодів для підвищення достовірності ПІ з виходу джерела інформації передається інформація надходить у вигляді інформаційних блоків, що містять k двійкових символів. Дані інформаційні блоки назвемо повідомленнями і позначимо через Q(x). Дані повідомлення є без надмірними і складають $K_p=2^k$ дозволених кодових повідомлень. У кодері з отриманих k інформаційних символів формуються за певними правилами l еревірочних символів і в канал зв'язку надходять кодові послідовності (позначимо їх через F (x)) довжиною n = (k + l) двійкових кодових символів Отже, загальна кількість n- розрядних (символьних) кодових послідовностей складе $K_{заг}=2^n$, з яких $K_{заб}=2^{n-k}=2^l$ є забороненими і від їх кількості залежить коригувальна здатність коду [3].

1.1.4 Циклічні коди

Циклічні коди є підкласом в класі лінійних блокових кодів, які відповідають певним вимогам. Були запропоновані Прейнджом. Свою назву ці коди отримали через те, що основною операцією побудови кодових послідовностей ($F_i(x)$) є цикл, а точніше циклічна перестановка двійкових

символів дозволених кодових послідовностей. На основі циклу забезпечується як формування кодових послідовностей, так і декодування [7].

Циклічним кодом називається такий груповий лінійний код, у якого в результаті циклічного зсуву кодових символів в дозволеній кодовії комбінації на 1,2,…,k-1 розрядів формуються дозволені кодові комбінації. Циклічний код може задаватися з використанням породжуючої і перевірочної матриць, породжуючого і перевірочного полиномов, а також коренів породжуючого полінома і залишків від ділення [7].

Основні властивості циклічних кодів:

- вага ($w_{к.п}$) дозволеної кодової последідовності $\geq d_0$;
- вага перевірочної частини $w_{пер.ч}$ дозволеної кодової последідовності $\geq d_0$-1;
- зсув кодових символів дозволеної кодової послідовності вліво або вправо на 1, 2,…, (k-1) символів знову призводить до дозволеної кодової послідовності. Якщо ж при циклічному зсуві завжди буде виходити нова кодова послідовність, то такий код буде називатися квазіциклічним;
- дозволена кодова послідовність без помилок $F_p'(x)$ при діленні на поліном P(x) дає нульовий залишок, тобто $F_p'(x)/P(x)=R(x)=0$ і R(x) не дорівнює 0 – при наявності помилок;
- сума по модулю два символів 2, 3, ..., (k-1) дозволених кодових послідовностей знову утворює дозволену кодову послідовність;
- Двочлен виду x^n +1 повинен ділитися на породжуючий поліном P (x) без залишку (мається на увазі звичайна операція ділення многочленів);
- результат ділення двочлена x^n+1 на утворюючий поліном P (x) дає поліном, який носить назву перевірочного полінома і позначається як h(x), тобто $h(x)=(x^n+1)/P(x)$;
- якщо всі операції над поліномами (кодовими послідовностями) проводяться в двійковому полі Галуа (GF (2)), тобто дії над коефіцієнтами поліномів здійснюється по модулю два, а множення поліномів проводиться

по модулю утворюючого полінома P (x), то застосування зазначених операцій не призводить до кодових послідовностей, довжина яких більше довжини заданого коду, тобто n.

Класифікація циклічних кодів:

- коди Хеммінга, як правило використовуються для корекції випадкових помилок;
- коди БЧХ – подальший розвиток кодів Хеммінга. Використовуються для корекції випадкових і пакетних помилок;
- коди Файр. Використовуються для корекції пакетних помилок;
- коди Ріда-Майлера. Використовуються для корекції помилок (випадкових і пакетних), а також для формування складних сигналів;
- коди Ріда-Соломона. Використовуються для корекції модульних помилок.

1.1.5 Згорткові коди

Згорткові коди (ЗК) мають великий науковий і практичний інтерес для сучасних систем і мереж телекомунікацій. Це визначається багатьма їх перевагами, а саме: високою швидкістю обробки інформації (десятки і сотні Мбіт / с), високою корегуючою здатністю як випадкових, так пакетних помилок, реалізацією ефективних кодеків і систем синхронізації розподільників інформації, ефективного використання в каналах зв'язку з фазовою невизначеністю та ін [5].

Сверточних код - це рекурентний код з періодичною напівнескінченною структурою символів кодової послідовності (рис.1.6) [6].

У загальному вигляді кодування інформації СК може бути представлене в таким чином.

$$T^{(i)} = \sum_{j=1}^{k_0} I^{(j)}(x) \cdot g^{(j)}(x), j = 1,2K\ k_0, i = j + 1, \qquad (1.2)$$

де I(x) – послідовність переданих інформаційних символів;

x – оператор затримки: (оператор затримки іноді позначають через латинську букву D);

g(x) – породжуючий або утворюючий поліном;

k_0 – блок інформаційних символів, що одночасно надходять на вхід кодуючого пристрою ($k_0 \geq 1$).

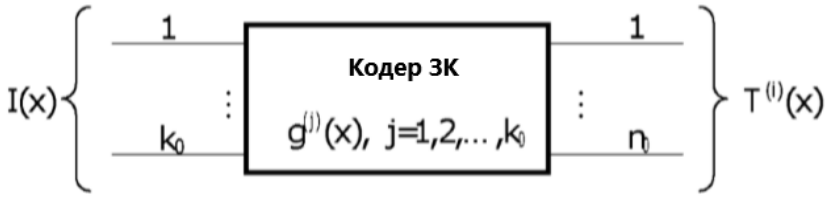

Рисунок 1.6 – Узагальнена структура кодера СК

До основних характеристик ЗК відносяться:

- $R=k_0/n_0=1/2$; 2/3; 3/4 і т.д. або $R= k_0/n_0=1/2$; 1/3; 1/4 і т.д.
- швидкість передачі коду, яка для ЗК записується у вигляді дробу;
- $J \geq 2$ кількість ортогональних перевірочних рівнянь;
- $d_0=J+1$ – мінімальна кодова відстань;
- $d_{віл}$ – вільна кодова відстань;
- $t_{випр} \leq J/2$ – кратність або кількість виправляє помилок;
- $t_{вияв} \leq d_0-l=J$ – кратність виявлених помилок;
- $n_A =(m+1)*n_0$ – довжина кодового обмеження або довжина кодової послідовності, відповідна кодуванню інформаційних блоків з k_0 символів протягом (m + 1) такту;
- $k_A=R*n_A$ – кількість інформаційних символів, що приходять на n_A кодових символів.

Згорткові коди, як і блокові лінійні коди, бувають:

- двійкові і недвійкові;
- алгебраїчні і неалгебраїчні;

- лінійні і нелінійні;
- систематичні і несистематичні;
- ортогональні і неортогональні.

Алгоритм формування кодових символів ЗК такий, що будь-якому вхідному інформаційному блоку з k0 двійкових символів і m (m-максимальний ступінь породжуючого полінома g (x)) попередніх інформаційних символів, що зберігаються в регістрі зсуву (RG) кодера, відповідає вихідний кодовий блок з n_0 двійкових символів. У зв'язку з тим, що в процесі формування n_0 кодових символів беруть участь m попередніх інформаційних символів (введених m тактами раніше), то такий алгоритм кодування називають кодуванням з пам'яттю [5].

У несистематичних ЗК в кодових блоках з n_0 двійкових символів немає в явному вигляді (неможливо виділити) інформаційних символів або блоків з k_0 двійкових символів. Кодування вхідної інформації здійснюється з пам'яттю і процес кодування може бути нескінченно тривалим.

Залежно від способу формування перевірочних рівнянь ЗК бувають ортогональними, самоортогональними и ортогоналізуємими.

Ортогональними ЗК (ОЗК) називають такі коди, у яких система з J ($J \leq 2$) перевірочних рівнянь ортогональна щодо декодуємих k_0 інформаційних символів і неортогональна щодо інформаційних символів, що входять в дані перевірочні рівняння [2].

Самоортогональні ЗК (СЗК) – коди, у яких декодований інформаційний символ входить одночасно в усі перевірочні рівняння, а всі інші символи, які беруть участь в декодуванні в даний момент часу, входять не більше, ніж в одне перевірочне рівняння, тобто ЗК формує, так звану, систему розділених перевірок.

Ортогоналізованими ЗК називаються такі коди, у яких під час декодування інформаційного або k_0 символів потрібно виконати додаткові лінійні перетворення над перевірочними символами для отримання додаткових, так званих, складових перевірок [5].

1.2 Твердотільні накопичувачі

SSD (або solid-state drive) – це немеханічний запам’ятовуючий пристрій на основі мікросхем пам’яті (як флеш або оперативна пам’ять). На сьогоднішній день SSD широко застосовуються в компактних пристроях: нетбуках, смартфонах, ноутбуках і комунікаторах (рис.1.7) та (рис.1.8) [11]. Також популярні гібридні жорсткі диски, що з’явилися через велику вартість твердотільних накопичувачів. Дані пристрої поєднують в собі твердотільний накопичувач малого об’єму, і звичні магнітні диски. Втім, не дивлячись на свою дорожнечу, порівняно зі звичайними жорсткими дисками, SSD все більше використовуються і в стаціонарних комп’ютерах. Нижче наведемо недоліки і переваги цих накопичувачів інформації [11].

Недоліки SSD:

- головний недолік solid-state drive – це те, що кількість циклів перезапису обмежено. Звичайна флеш пам’ять дозволяє переписувати дані приблизно десять тисяч разів. Дорогі види пам’яті – дозволяють перезаписувати дані більше ста тисяч разів. Схеми балансування навантаження – це те, що дозволяє боротися з нерівномірним зносом. Інформація про перезапис блоків зберігається на контролері і в разі необхідності блоки міняються місцями;

Рисунок 1.7 – Твердотільний накопичувач

- також існує проблема сумісності з актуальними і старішими версіями ОС Microsoft Windows. Вони не враховують специфіку SSD накопичувачів і зношують їх додатково. Наприклад використання ОС підкачки (файл підкачки) на SSD зменшує термін експлуатації накопичувача;
- гігабайт пам'яті HDD (звичайного механічного жорсткого диска) істотно нижче ціни гігабайту пам'яті SSD-накопичувача. До того ж ціна на накопичувачі SSD-зростає пропорційно, а на HHD росте повільніше, оскільки вона залежить від кількості пластин, що і викликає менше підвищення ціни;
- так як в SSD – накопичувачах застосовується команда TRIM, то відновлення видаленої інформації з допомогою recovery-утиліт стає неможливим;
- ціна.

Рисунок 1.8 – HDD та SSD накопичувачі

Переваги SSD:

- рухомі частини відсутні;
- швидкість читання\запис – висока, вона перевершує пропускну здатність інтерфейсу ЖД;
- низьке енергоспоживання;
- у зв'язку з тим, що відсутні рухомі частини, то і шум – відсутній;

- механічна стійкість – висока;
- діапазон робочих температур – широкий;
- стабільність часу зчитування файлів незалежно від їх розташування або фрагментації;
- габарити і вага – малі;
- потенціал, модернізаційний, як у технологій виробництва, так і у самих накопичувачів – великий;
- сприйнятливість до зовнішніх електромагнітних полів – набагато менша ніж у HHD [27].

1.3 Алгоритми, які застосовують у твердотільних накопичувачах

1.3.1 Код Хеммінга

Одними з найбільш простих є коди Хеммінга, представлені Хеммінга в 1950 р. До даних кодів відносяться лінійні блокові коди з параметрами (n, k) виду $2^m - 1, 2^m - m - 1$, де m=n-k – число перевірочних символів коду. Коди Хеммінга мають кодову відстань $d_{min} = 3$ і тому здатні виправляти тільки одну або виявити дві помилки [1].

Для завдання кодів Хеммінга зазвичай використовується перевірочна матриця H, що містить m рядків і $2^m - 1$ стовпців, причому стовпцями є всі можливі ненульові двійкові вектори довжини m. Для прикладу наведемо перевірочну матрицю (7, 4) коди Хеммінга

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \quad (1.3)$$

Породжуюча матриця даного коду формується з одиничної матриці розміром k × k і транспонованої підматриці, що складається з перших k рядків матриці H

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \quad (1.4)$$

Коди Хеммінга є одними з небагатьох кодів, для яких повністю відомо розподіл ваг кодових слів, яке визначається ваговою функцією

$$N(x) = \sum_{i=0}^{n} N_i x^i, \quad (1.5)$$

де N_i – число кодових слів ваги i. Для кодів Хеммінга з $d_{min} = 3$ s n $=2^m$-1 вагова функція має вигляд

$$N(x) = \frac{1}{n+1}\left[(1+x)^n + n(1+x)^{\frac{n-1}{2}}(1-x)^{\frac{n+1}{2}}\right]. \quad (1.6)$$

Знання спектру коду, може виявитися корисним при отриманні кордонів його ефективності.

Крім описаних кодів Хеммінга існують так звані скорочені і розширені коди Хеммінга. Скорочені коди Хеммінга виходять при виключенні будь-якої перевірки (тобто видалення одного рядка перевірочної матриці). Розширені коди Хеммінга виходять шляхом введення додаткової перевірки на парність всіх символів кодового слова. результаті їх мінімальна відстань збільшується до $d_{min} = 4$, що дозволяє цим кодами виправляти одну і виявляти дві або тільки виявляти три помилки. Перевірочна матриця H розширеного коду Хеммінга, заданого матрицею (1.7), має вигляд

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (1.7)$$

Для зручності роботи дану матрицю можна привести до канонічної (систематичної) форми за допомогою додавання до останнього рядка всіх інших рядків

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (1.8)$$

Спектр розширених кодів Хеммінга задається ваговою функцією

$$N(x) = \frac{1}{2n}\left[(1+x)^n + (1+x)^n + 2(n-1)(1-x)^{\frac{n}{2}}\right]. \quad (1.9)$$

Далі розглянемо характеристики кодів Хеммінга. На рис.1.9 і 1.10 [1] представлені експериментальні залежності ймовірності бітової помилки P_b від ймовірності помилки в каналі p і від ставлення сигнал-шум на біт E_b/N_0 для кодів Хеммінга з m= 3…7 при роботі в ДСК, декодуємих за допомогою декодера Меггіта. Між ймовірністю помилки p в каналі і ставленням сигнал-шум на біт E_b/N_0 існує наступна залежність

$$p = Q\left(\sqrt{2\frac{E_s}{N_0}}\right) = Q\left(\sqrt{2r\frac{E_b}{N_0}}\right), \quad (1.10)$$

де Q(x) – функція, визначена (2.3);

r - кодова швидкість коду (r=k/n).

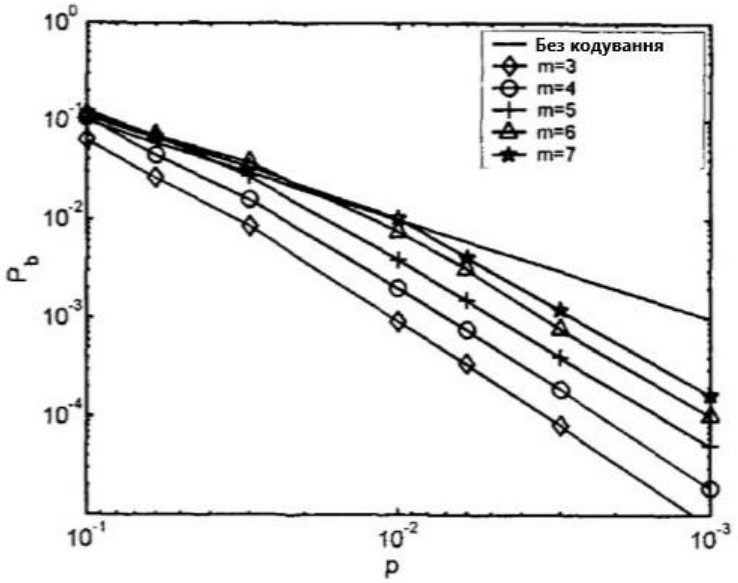

Рисунок 1.9 – Характеристики кодів Хеммінга в ДСК для різних значень m

Як видно, з представлених рисунків, коди Хеммінга мають дуже слабку корегуючу здатністю і окремо практично не використовуються. Однак застосування даних кодів в складі каскадних схем кодування дозволяє отримати дуже хороші результати [1].

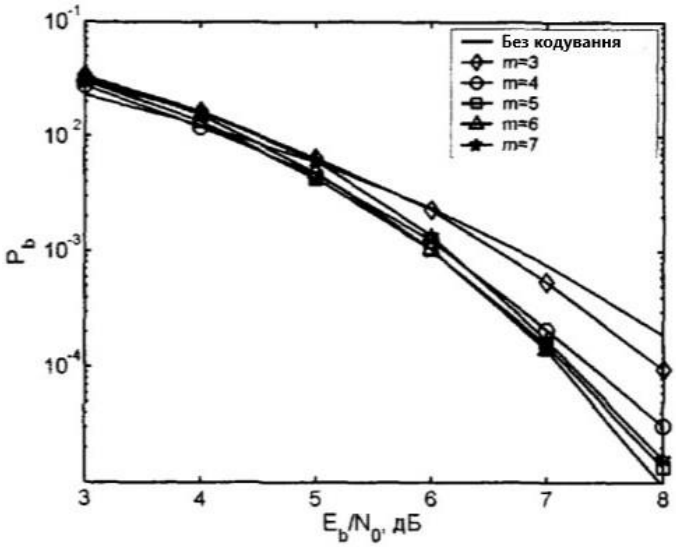

Рисунок 1.10 – Характеристики кодів Хеммінга в ДСК для різних значень m

1.3.2 Код Боуза - Чоудхурі – Хоквінгема

Коди Боуза – Чоудхурі – Хоквінгема представляють собою клас лінійних циклічних кодів, що виправляють кратні помилки, і є узагальненням раніше описаних кодів Хеммінга. Коди БЧХ зазвичай задаються через коріння породжуючого багаточлена g (x) ступеня n-k. Дані коди визначаються представленими нижче чином [1].

Примітивним кодом БЧХ, що виправляє t помилок, називається блоковим кодом завдовжки $n=q^m - 1$ над полем Галуа GF (q), для якого елементи α^{m0}, α^{m0+1}, …, $\alpha^{m0+2t-1}$ (для довільного m_0) є корінням породжуючого багаточлена g(x), де α – примітивний елемент поля $GF(q^m)$.

Грунтуючись на представленому визначенні породжуючого багаточлена коду БЧХ можна записати у вигляді:

$$g(x) = \text{НОК}\left(m_{m_0}(x), (m_{m_0+1}(x), \dots, (m_{m_0+2t-1}(x)\right), \qquad (1.9)$$

де $m_c(x)$ – мінімальна функція a^c у полі GF(q), причому в разі двійкових кодів БЧХ найменше спільне кратне шукається тільки для мінімальних функцій з непарними індексами.

Непрімітівние коди БЧХ визначаються аналогічно, але примітивний елемент α замінюється непрімітівним елементом β поля $GF(q^m)$, і довжина блоку стає рівною порядку β.

Для більшої кількості практично використовуваних значень n, k і t породжують поліноми вже отримані [1].

Можна отримати асимптотичні оцінки залежності ймовірності помилки в кодовому блоці P_B, q-ічном символі P_s і біті P_b при декодуванні кодів БЧХ по максимуму правдоподібності. Дані оцінки не вимагають знання спектра коду і мають вигляд:

$$P_B = \sum_{l=t+1}^{n} C_n^l P_q^l (1 - P_q)^{n-l}. \qquad (1.10)$$

$$P_s \leq \frac{1}{n}\sum_{l=t+1}^{n}(i+t)C_n^l P_q^l\,(1-P_q)^{n-l}. \qquad (1.11)$$

$$P_b \approx \frac{q/2}{q-1}P_s. \qquad (1.12)$$

При виведенні даних оцінок передбачалося, що кодовий блок декодируется невірно при наявності в ньому більш ніж t помилкових символів, а неправильно декодуване кодове слово відрізняється від переданого не більше ніж в (i+t) q-ічных символах. Також вважалося, що в неправильно декодованому символі приблизно половина кодових бітів буде помилковою.

Для деяких (n,k) кодів БЧХ з $r = k/n \approx 1/2$ оцінка ймовірності символьної помилки представлена на рис. 1.11 [1]. На рис. 1.12 [1] показані графіки залежності оцінки ймовірності помилки на символ P_s від відношення сигнал-шум на біт E_b/N_0 для кодових швидкостей r, близьких до 1/3 і 2/3. Як випливає з зіставлення представлених графіків, енергетична ефективність кодів БЧХ зі зменшенням кодової швидкості знижується [1].

Оскільки коди БЧХ відносяться до циклічних блокових кодів, що задається утворюючим поліномом, кодування інформаційної послідовності здійснюється відповідно до $C(x) = U(x)g(x)$. Декодування кодів БЧХ може виконуватися за допомогою алгебраїчних методів.

На рис. 1.13 [1] представлені експериментальні залежності ймовірності помилки на біт Pb від відношення сигнал-шум на біт Eb/N0 для кодів БЧХ з кодовою швидкістю r≈1/2 при роботі в ДСК. З зіставлення даного малюнка з рис. 1.11 (для двійкових кодів ймовірність помилки на біт Pb і на символ Ps совпадают) збігаються) видно, що аналітична залежність ймовірності помилки (1.12) цілком підходить для оцінювання зверху її реального значення [1].

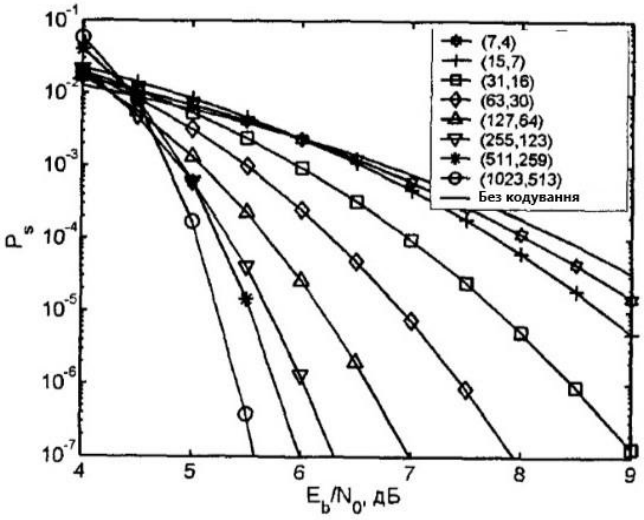

Рисунок 1.11 – Оцінка характеристик різних кодів БЧХ в ДСК для кодових швидкостей, близьких до 1/2

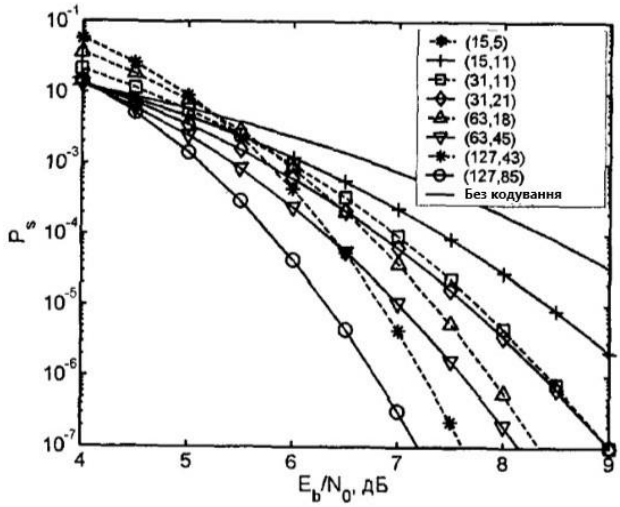

Рисунок 1.12 – Оцінка характеристик різних кодів БЧХ в ДСК для кодових швидкостей, близьких до 1/3 і 2/3

Як видно з представлених характеристики, коди БЧХ мають суттєво кращу ефективність в порівнянні з кодами Хеммінга, однак ці характеристики все ще дуже далекі від граничних значень . Крім того, складність декодування даних кодів при великому значенні n дуже велика. Тому коди БЧХ невеликої довжини, так само як і коди Хеммінга, в основному застосовуються в якості складових елементів більш ефективних каскадних кодів [1].

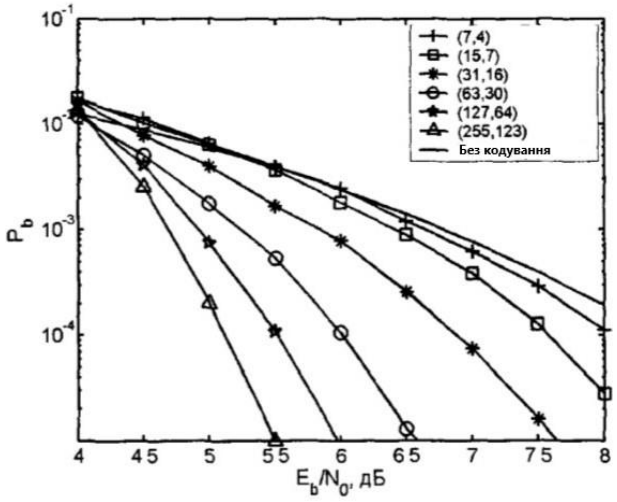

Рисунок 1.13 – Характеристики різних кодів БЧХ в ДСК для кодових швидкостей, близьких до ½

1.3.3 Код Ріда-Соломона

Важливий підклас кодів БЧХ становлять коди Ріда Соломона, для яких $m=m_0=1$. Ці недвійкові коди характеризуються такими параметрами:

- довжина блоку n=q-1, виражена q-ічних символів;
- кількість інформаційних символів k від 1 до n-1;
- кодова швидкість r=k/n.

Для задання кодів Ріда-Соломона використовується породжуючий багаточлен виду:

$$g(x) = (x - \alpha)(x - \alpha^2) \dots (x - \alpha^{2t}), \qquad (1.13)$$

де $t = \left[\frac{1}{2}(d_{min} - 1)\right]$.

Оскільки утворюючий поліном має ступінь 2t, можливе використання всього лише 2t перевірочних символів для виправлення пакетів з t помилок. Остання властивість дозволило знайти цим кодами досить широке застосування в каскадних методах кодування. Крім того, для декодування кодів Ріда-Соломона існують досить ефективні алгоритми декодування жорстких рішень, що дозволяє використовувати відносно довгі коди в

багатьох практичних додатках, особливо в системах з q-ічною модуляцією [1].

Так як коди Ріда-Соломона є однією з різновидів кодів БЧХ, для оцінки їх ефективності можна скористатися формулами (1.10)-(1.12). Для прикладу на рис. 1.14 [1] представлені залежності оцінки ймовірності символьної помилки P_s від ймовірності помилки P_q в q-ичном симетричному каналі qСК для різних значень q і кодових швидкостей r. Криві на рис.1.15 [1] відображають залежність оцінок P_s для тих же кодів від відносини сигнал-шум E_b/N_0 а інформаційний біт.

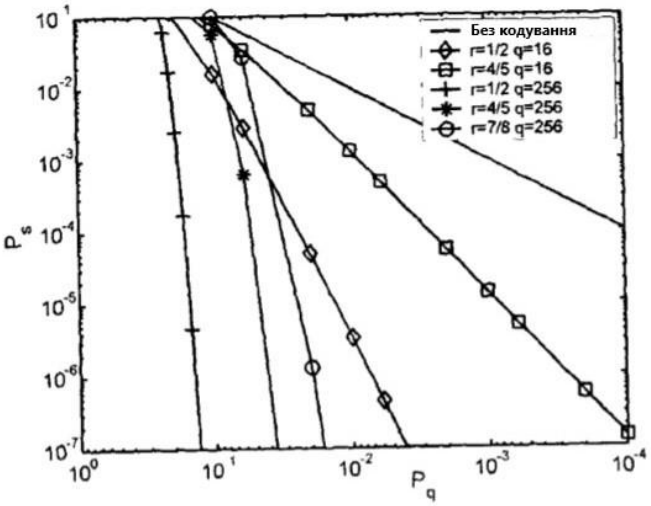

Рисунок 1.14 – Оцінка характеристик кодів Ріда-Соломона в qСК

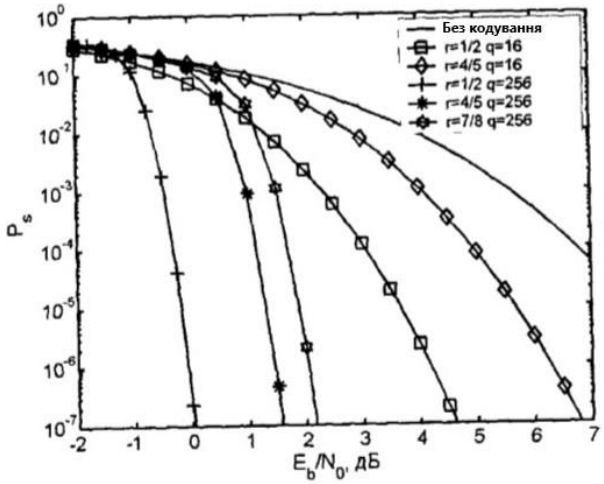

Рисунок 1.15 – Оцінка характеристик кодів Ріда-Соломона в qСК

1.3.4 Код з малою щільністю перевірок на парність

Код з малою щільністю перевірок на парність (LDPC-код від англ. Low-density parity-check code, LDPC-code, нізкоплотностний код) - використовується в передачі інформації, окремий випадок блочного лінійного коду з перевіркою парності. Особливістю є мала щільність значущих елементів перевірочної матриці, за рахунок чого досягається відносна простота реалізації засобів кодування.

Також називають кодом Галлагера, на ім'я автора першої роботи на тему LDPC-кодів.

У 1948 році Шеннон опублікував свою роботу з теорії передачі інформації. Одним з ключових результатів роботи вважається теорема про передачу інформації для каналу з шумами, яка говорить про можливість звести ймовірність помилки передачі по каналу до мінімуму при виборі достатнього великої довжини ключового слова - одиниці інформації, що передається по каналу.

Рисунок 1.16 – Спрощена схема передачі інформації по каналу з шумами

При передачі інформації її потік розбивається на блоки певної (найчастіше) довжини, які перетворюються кодером (кодуються) в блоки, званими ключовими словами. Ключові слова передаються по каналу, можливо з спотвореннями. На приймаючій стороні декодер перетворює ключові слова в потік інформації, виправляючи (по можливості) помилки передачі рис.1.16 [8].

Теорема Шеннона стверджує, що при певних умовах ймовірність помилки декодування (тобто неможливість декодером виправити помилку передачі) можна зменшити, вибравши велику довжину ключового слова.

Однак, дана теорема (і робота взагалі) не показує, як можна вибрати велику довжину, а точніше як ефективно організувати процес кодування і декодування інформації з великою довжиною ключових слів. Якщо припустити, що в кодері і декодері є якісь таблиці відповідності між вхідним блоком інформації та соответствующим кодовым словом, то такие таблицы відповідним кодовим словом, то такі таблиці будуть займати дуже багато місця. Для двійкового симетричного каналу без пам'яті (якщо говорити спрощено, то на вхід кодера надходить потік з нулів і одиниць) кількість різних блоків становить 2^n, де n – кількість біт (нулів або одиниць) які будуть перетворюватися в одне кодове слово. Для 8 біт це 256 блоків інформації, кожен з яких буде містити в собі відповідне кодове слово. Для 8 бит это 256 блоков информации, каждый из которых будет содержать в себе соответствующее кодовое слово. Причому кодове слово зазвичай більшої довжини, так як містить в собі додаткові біти для захисту від помилок передачі даних. Тому одним із способів кодування є використання перевірочної матриці, які дозволяють за одну математичну дію (множення рядка на матрицю) виконати декодування кодового слова. Аналогічним чином кожної перевірочної матриці відповідає породжуюча матриця, аналогічним способом однією операцією множення рядка на матрицю генеруючої кодове слово.

Таким чином, для порівняно коротких кодових слів кодери і декодери можуть просто містити в пам'яті всі можливі варіанти, або навіть реалізовувати їх у вигляді напівпровідникової схеми. Для більшого розміру кодового слова ефективніше зберігати породжуючу і перевірочну матрицю. Однак, при довжинах блоків в кілька тисяч біт зберігання матриць розміром, відповідно, в мегабіта, вже стає неефективним. Одним із способів вирішення цієї проблеми стає використання кодів з малою щільністю перевірок на парність, коли в матриці, яка перевіряє кількість одиниць порівняно мало, що дозволяє ефективніше організувати процес зберігання матриці або ж

безпосередньо реалізувати процес декодування за допомогою напівпровідникової схеми.

Першою роботою на цю тему стала робота Роберта Галлагера «Low-Density Parity-Check Codes» 1963 роки (основи якої були закладені в його докторської дисертації 1960 року),. В роботі вчений описав вимоги до таких кодів, описав можливі способи побудови і способи їх оцінки. Тому часто LDPC-коди називають кодами Галлагера. У науковій літературі коди також називають нізкоплотностнимі кодами або кодами з малою щільністю перевірок на парність.

Однак, через складність в реалізації кодерів і декодерів ці коди не використовувалися. Лише значно пізніше, з розвитком телекомунікаційних технологій, знову зріс інтерес до передачі інформації з мінімальними помилками. Незважаючи на складність реалізації в порівнянні з турбо-кодом, відсутність завад до використання (незахищеність патентами) зробило LDPC-коди привабливими для телекомунікаційної галузі, і фактично стали стандартом де-факто. У 2003 році LDPC-код, замість турбо-коду, став частиною стандарту DVB-S2 супутникової передачі даних для цифрового телебачення. Аналогічна заміна сталася і в стандарті DVB-T2 для цифрового наземного телевізійного мовлення.

LDPC-коди описуються нізкоплотностной перевірочною матрицею, що містить в здебільшого нулі і відносно малу кількість одиниць. За визначенням, якщо кожен рядок матриці містить рівно k <n і кожен стовпець рівно j <r одиниць, то код називають регулярним (в іншому випадку - нерегулярним). У загальному випадку кількість одиниць в матриці має порядок O (n), тобто росте лінійно зі збільшенням довжини кодового блоку (кількості стовпців перевірочної матриці) рис.1.17. [8].

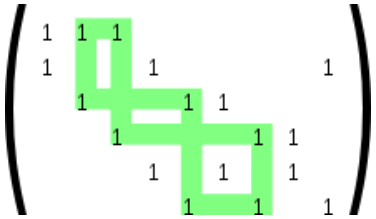

Рисунок 1.17 – Перевірочна матриця LDPC-коду (9, 2, 3) з мінімальним циклом довжини 8

Зазвичай розглядаються матриці великих розмірів. Наприклад, в роботі Галлагера в розділі експериментальних результатів використовуються «малі» кількості рядків n = 124, 252, 504 і 1008 рядків (число стовпців перевірочної матриці трохи більше). На практиці застосовуються матриці з великою кількістю елементів - від 10 до 100 тисяч рядків. Однак кількість одиниць в рядку або в стовпці залишається досить малим, зазвичай меншим 10. Помічено, що коди з тією ж кількістю елементів на рядок або стовпець, але з більшим розміром, мають кращі характеристики.

Важливою характеристикою матриці LDPC-коду є відсутність циклів певного розміру. Під циклом довжини 4 розуміють утворення в перевірочній матриці прямокутника, в кутах якого стоять одиниці. Відсутність циклу довжини 4 можна також визначити через скалярний добуток стовпців (або рядків) матриці. Якщо кожний попарний скалярний твір всіх стовпців (або рядків) матриці не більше 1, це говорить про відсутність циклу довжини 4. Цикли більшої довжини (6, 8, 10 і т.д.) можна визначити, якщо в перевірочній матриці побудувати граф, вершинами якого є одиниці, а ребра - всі можливі сполуки вершин, паралельні сторонам матриці (тобто вертикальні або горизонтальні лінії). Мінімальний цикл в цьому графі і буде мінімальним циклом в перевірочній матриці LDPC-коду. Зрозуміло, що цикл буде мати довжину як мінімум 4, а не 3, так як ребра графа повинні бути паралельні сторонам матриці. Взагалі, будь-який цикл в цьому графі буде мати парну

довжину, мінімальний розмір 4, а максимальний розмір зазвичай не грає ролі (хоча, зрозуміло, він не більше, ніж кількість вузлів в графі, тобто n × k).

Опис LDPC-коду можливий декількома способами:

- перевірочній матрицею;
- дводольним графом;
- іншим графічним способом;
- спеціальним способом.

Останній спосіб є умовним позначенням групи уявлень кодів, які побудовані за заданими правилами-алгоритмам, таким, що для повторного відтворення коду досить знати лише ініціалізуючі параметри алгоритму, і, зрозуміло, сам алгоритм побудови. Однак цей спосіб не є універсальним і не може описати всі можливі LDPC-коди.

Спосіб задання коду перевірочної матриці є загальноприйнятим для лінійних кодів, коли кожен рядок матриці є елементом деякої безлічі кодових слів. Якщо всі рядки лінійно-незалежні, рядки матриці можуть розглядатися як базис безлічі всіх кодових векторів коду. Однак використання даного способу створює складності для подання матриці в пам'яті кодера - необхідно зберігати всі рядки або стовпці матриці у вигляді набору двійкових векторів, через що розмір матриці стає рівний j × k біт рис.1.19. [8].

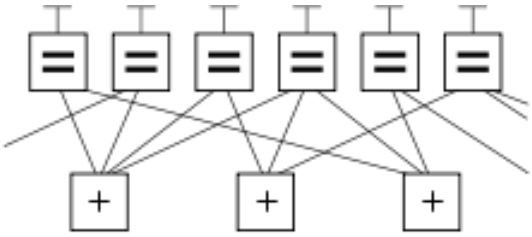

Рисунок 1.19 – Подання LDPC-коду у вигляді двудольного графа

Поширеним графічним способом є представлення коду у вигляді двудольного графа. Порівняємо всі k рядків матриці k нижнім вершинам

графа, а n стовпців - верхнім, і з'єднаємо верхні і нижні вершини графа, якщо на перетині відповідних рядків і стовпців стоять одиниці.

До інших графічних способів відносять перетворення двудольного графа, що відбуваються без фактичної зміни самого коду. Наприклад, можна всі верхні вершини графа представити у вигляді трикутників, а всі нижні - у вигляді квадратів, після чого розташувати ребра і вершини графа на двомірної поверхні в порядку, зручному для візуального розуміння структури коду рис.1.20. [8].

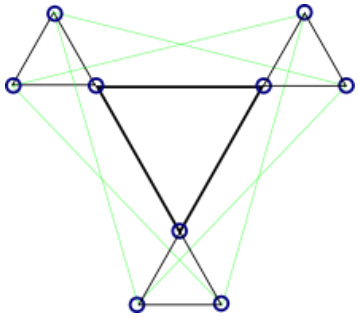

Рисунок 1.20 – Подання (9, 2, 3) LDPC-коду у вигляді графа спеціального виду.

Вводячи додаткові правила графічного відображення і побудови LDPC-коду, можна досягти, що в процесі побудови код отримає певні властивості. Наприклад, якщо використовувати граф, вершинами якого є тільки стовпці перевірочної матриці, а рядки зображуються многогранниками, побудованими на вершинах графа, то дотримання правила «два багатогранника не поділяють одне ребро» дозволяє позбутися від циклів довжини 4.

При використанні спеціальних процедур побудови коду можуть використовуватися і свої способи подання, зберігання і обробки (кодування і декодування).

В даний час використовуються два принципи побудови перевірочної матриці коду. Перший заснований на генерації початкової перевірочної матриці за допомогою псевдовипадкового генератора. Коди, отримані таким

способом, називають випадковими. Другий - використання спеціальних методів, заснованих, наприклад, на групах і кінцевих полях. Коди, отримані цими способами, називають структурованими. Кращі результати щодо виправлення помилок показують саме випадкові коди, однак структуровані коди дозволяють використовувати методи оптимізації процедур зберігання, кодування і декодування, а також отримувати коди з більш передбачуваними характеристиками.

У своїй роботі Галлагер вважав за краще за допомогою генератора псевдовипадкових чисел створити початкову перевірочну матрицю невеликого розміру з заданими характеристиками, а далі збільшити її розмір, дублюючи матрицю і використовуючи метод перемішування рядків і стовпців для позбавлення від циклів певної довжини.

У 2003 році Джеймсом МакГованом і Робертом Вільямсоном був запропонований спосіб видалення циклів з матриці LDPC-коду, в зв'язку з чим стало можливим на початку згенерувати матрицю із заданими характеристиками (n, j, k), а потім видалити з неї цикли. Так відбувається в схемі Озарова-Вайнера.

У 2007 році в журналі «IEEE Transactions on Information Threory» була опублікована стаття про використання кінцевих полів для побудови квазі-циклічних LDPC-кодів для каналів з адитивним білим гауссовим шумом і двійкових каналів зі стиранням.

1.3.5 Код розроблений фірмою Toshiba

Однією із самих основних вимог для будь-якого пристрою зберігання даних є здатність ідентифікувати та виправляти помилки даних. Більш щільні і більш економічні NAND-технології, такі як TLC NAND і більш дрібні літографії NAND, мають підвищену частоту помилок в біт. У традиційних реалізаціях виправлення помилок SSD використовувався BCH ECC для боротьби з помилками даних, але більш високі частоти помилок при бітах привели до розробки LDPC (перевірка коефіцієнта неповної щільності).

LDPC володіє двома рівнями виправлення помилок, жорстким рішенням і м'яким рішенням, що підвищує можливості корекції і зменшує затримку.

У режимі реального часу LDPC вимагає більше обчислювальної потужності, ніж BCH ECC, і компроміс полягає в тому, що схеми декодування більше і споживають більше енергії в результаті. LDPC має кілька переваг для попередніх підходів до виправлення помилок, але Toshiba дотримується іншого шляху за допомогою власної технології корекції помилок QSBC (Quadruple Swing-By Code) рис.1.21. [10].

Виробники контролерів Flash розробляють власні алгоритми LDPC, і кожен з них буде мати різні характеристики, тому пряме порівняння різних підходів не зовсім можливе. Багато деталей пропрієтарних кодів LDPC, наряду з QSBC, тісно охороняються інтелектуальною власністю.

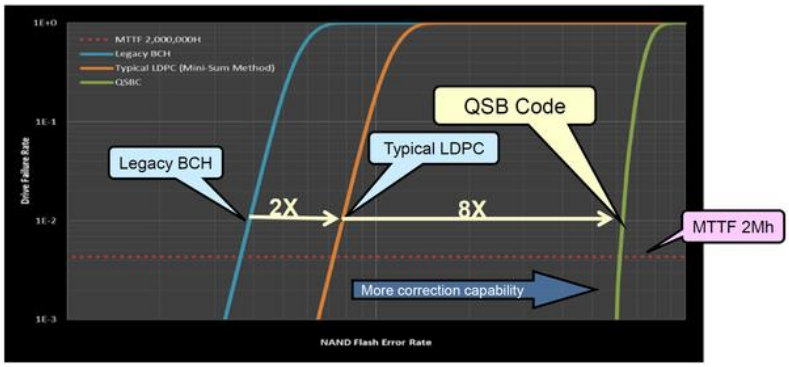

Рисунок 1.21 – Можливості корекції QSBC

На рисунку вище показана можливість виправлення помилок трьох типових реалізацій. Типовий Mini-Sum LDPC пропонує збільшення 2x в порівнянні з BCH, але QSBC від Toshiba перевершує Mini-Sum LDPC на 8x. Toshiba має довгу історію з виробництвом NAND і подоланням помилок, тому не дивно, що у них вже є передове приватне рішення, інтегроване в їх продукти.

Toshiba також використовує наскрізне виявлення і корекцію помилок для захисту кожного кроку по шляху даних і спалаху NAND рис.1.22. [10].

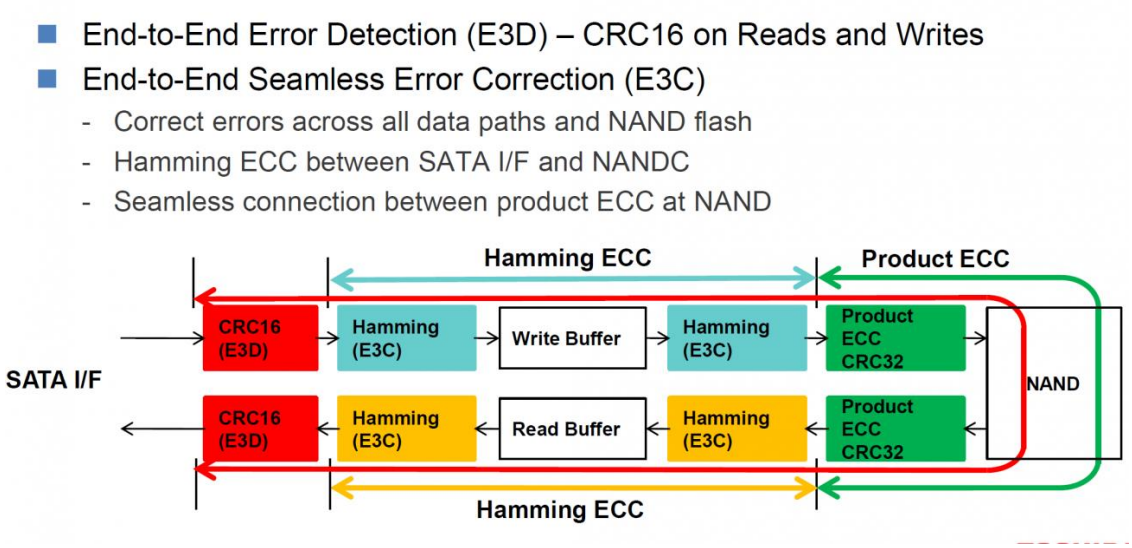

Рисунок 1.22 – Наскрізний захист даних

1.4 Висновки за розділом 1

Розглянуті матеріали дали змогу проаналізувати що таке завадостійке кодування, які існують його види, було розглянуте ,що таке твердотільні накопичувачі, а також алгоритми завадостійкого кодування що в них застосовуються.

2 ПРИНЦИПИ МОДЕЛЮВАННЯ АЛГОРИТМІВ ЗАВАДОСТІЙКОГО КОДУВАННЯ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Схема статистичного імітаційного моделювання функціонування завадостійкого кодека

Основним завданням завадостійкого кодування є вирішення проблеми забезпечення високої достовірності даних, які передаються за рахунок застосування пристроїв кодування / декодування в складі системи передачі цифрової інформації, структурна схема якої представлена на рис. 2.1 [4]. Дана схема широко використовується в теорії завадостійкого кодування, оскільки вона охоплює більшість ситуацій, які зустрічаються на практиці [4].

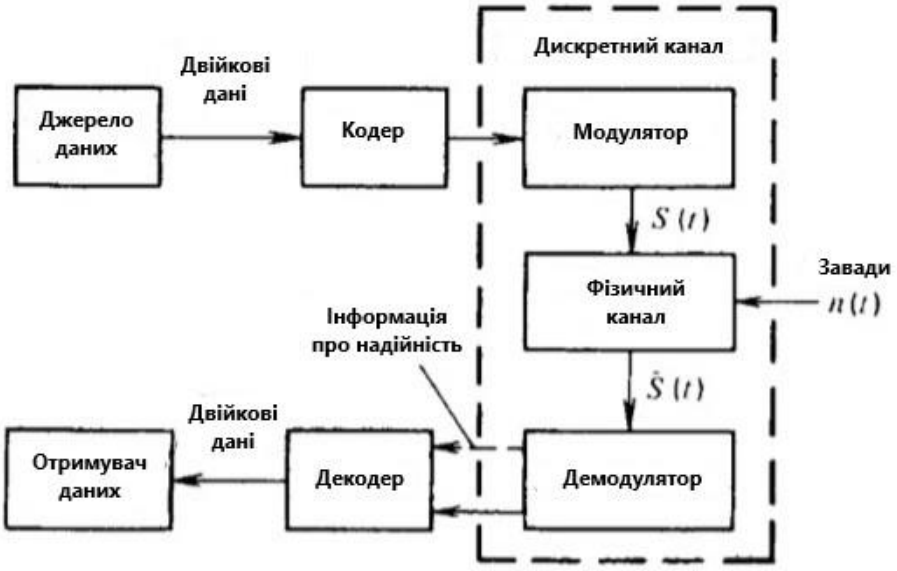

Рисунок 2.1 – Загальна модель цифрового зв'язку

Розглянемо основні принципи роботи представленої схеми. Спочатку джерело даних породжує дані у вигляді двійкових символів. Зазвичай припускають, що «нулі» і «одиниці» з'являються незалежно один від одного і з однаковими можливостями. Потім кодер каналу вносить в прийняту інформаційну послідовність деяку надмірність (даний процес називається

кодуванням), яку декодер зможе використовувати для виправлення виникаючих при передачі даних по каналу зв'язку помилок [4].

Закодовані дані з виходу кодера надходять на модулятор, який за допомогою будь-якого методу модуляції реалізує їх відображення в аналоговий сигнал S(t). Модулятор може просто відобразити кожен двійковий символ в один з M=2 можливих сигналів $s_0(t)$ і $s_1(t)$ (в цьому випадку говорять про двійкову модуляцію), а може передавати q-бітові блоки (q>1) за допомогою $M=2^q$ можливих сигналів (M-позиційна модуляція).

У фізичному каналі сигнал S(t) піддається впливу шуму n(t). Для кількісної оцінки ступеня впливу шуму n(t) на сигнал S(t) зазвичай використовують відношення сигнал-шум E_s/N_0, яке визначається як відношення потужності сигналу P_c до потужності шуму $P_ш$. асто дане відношення виражається в децибелах, тобто $E_s/N_0 = 10 \lg(P_c/P_ш)$. Зауважимо, що на всіх наведених в довіднику графіках відношення сигнал-шум виражено в децибелах, а у всіх формулах використовується безрозмірна величина $E_s/N_0=P_c/P_ш$. Далі демодулятор перетворює прийнятий з каналу сигнал R (t) в послідовність чисел, що представляють оцінку переданих даних. Після цього детектор квантує вихід демодулятора на Q-рівнів. У разі якщо Q = M, то кажуть, що детектор виносить жорстке рішення щодо переданих символів, якщо ж Q> M, то детектор виносить м'які рішення.

Потім квантований вихід детектора надходить на декодер каналу, який, використовуючи внесену кодером надмірність, визначає передане джерелом повідомлення (даний процес називається декодуванням).

2.1.1 Генератор вхідних впливів

Джерело даних породжує дані у вигляді двійкових символів. Звичайне припущення полягає в тому, що дані породжуються таким чином, що «нулі» і «одиниці» з'являються незалежно і з однаковими можливостями. Однак в деяких виникаючих на практиці ситуаціях деякі набори даних можуть

виникати частіше за інших, і це іноді може зробити невірними висновки, справедливі для повністю випадкових даних [4].

2.1.2 Кодер

Розглядаються два типи кодів: блокові і деревовидні. Визначальна відмінність між кодерами для кодів цих двох типів полягає в наявності або відсутності пам'яті. Кодер для блокового коду є пристроєм без пам'яті, що відображає послідовності з k вхідних символів в послідовності з n вихідних символів. Термін «без пам'яті» вказує, що кожен блок з n символів залежить тільки від відповідного блоку з k символів і не залежить від інших блоків. Це не означає, що кодер не містить елементів пам'яті. Важливими параметрами блокового коду є n,k, R=k/n і d_{min} . На практиці значення k лежать між 3 і декількома сотнями, а R = 1/4…7/8. начення, що лежать поза цими межами, є можливими, але часто призводять до деяких практичних труднощів. Вхідні і вихідні послідовності зазвичай складаються з двійкових символів, але іноді можуть складатися з елементів деякого алфавіту більшого обсягу. Кодер для деревовидного коду є пристроєм з пам'яттю, в яке надходять набори з m двійкових вхідних символів, а на виході з'являються набори з n двійкових вихідних символів. Кожен набір n вихідних символів залежить від поточного вхідного набору і від v опередніх вхідних символів. Таким чином, пам'ять кодера повинна містити v + m вхідних символів. Параметр v + m часто називають довжиною кодового обмеження даного коду і позначають k = v + m (не слід плутати з параметром k для блокового коду). Відзначимо, що позначення часто не узгоджуються один з одним. Деякі автори називають довжиною кодового обмеження параметр k в той час як інші - параметр v. Довжиною кодового обмеження будемо називати величину v оскільки це призводить до меншої плутанини для кодів з m >1 Параметр k = v + m майже не буде використовуватися. Деревовидні коди характеризуються також швидкістю R = m / n і вільною відстанню $d_{віл}$. Точне визначення $d_{віл}$ більш громіздке, ніж визначення d_{min} для блокових кодів, однак параметр $d_{віл}$ по

суті, містить ту ж інформацію про код, що і d_{min}. Типові значення параметрів деревовидних кодів такі: m, n = 1…8, R= 1/4…7/8, v= 2…60.

При іншому підході коди можна розділити на лінійні і нелінійні. Лінійні коди утворюють векторний простір і мають наступну важливу властивість: два кодових слова можна скласти, використовуючи відповідне визначення суми, і здобув третє кодове слово. Що стосується звичайних двійкових кодів ця операція є посимвольним складанням двох кодових слів по модулю 2 (тобто 1+1=0, 1+0=1, 0+0=0). Ця властивість призводить до двох важливих наслідків. Перше з них полягає в тому, що лінійність істотно спрощує процедури кодування та декодування, дозволяючи висловити кожне кодове слово в вигляді «лінійної» комбінації невеликого числа виділених кодових слів, так званих базисних векторів. Друга властивість полягає в тому, що лінійність істотно спрощує завдання обчислення параметрів коду, оскільки відстань між двома кодовими словами при цьому еквівалентна відстані між кодовим словом, що складається цілком з нулів, і деяким іншим кодовим словом. Таким чином, при обчисленні параметрів лінійного коду досить розглянути, що відбувається при передачі кодового слова, що складається цілком з нулів. Обчислення параметрів спрощується ще й тому, що відстань Хеммінга між даним кодовим словом і нульовим кодовим словом дорівнює числу ненульових елементів даного кодового слова. Це число часто називають вагою Хеммінга даного слова, і список, що містить число кодових слів кожної ваги, можна використовувати для обчислення характеристик коду за допомогою адитивного кордону. Такий список називають спектром коду [4].

Майже всі схеми кодування, що застосовуються на практиці, засновані на лінійних кодах. Подвійні лінійні блокові коди часто називають груповими кодами, оскільки кодові слова утворюють математичну структуру, яка називається групою. Лінійні деревовидні коди зазвичай називають згортковими кодами, оскільки операцію кодування можна розглядати як дискретну згортку вхідної послідовності з імпульсним відгуком кодера [4].

Нарешті, коди можна розбити на коди, що виправляють випадкові помилки, і коди, що виправляють пакети помилок. В основному будемо мати справу з кодами, призначеними для виправлення випадкових, або незалежних, помилок. Для виправлення пакетів помилок було створено багато кодів, що мають хороші параметри. Однак при наявності пачок помилок часто виявляється більш вигідним використовувати коди, що виправляють випадкові помилки, разом з пристроєм перемежения відновлення. Такий підхід включає в себе процедуру перемішування порядку символів в закодованій послідовності перед передачею і відновленням вихідного порядку символів після прийому з тим, щоб рандомізувати помилки, об'єднані в пакети.

2.1.3 Модулятор

Модулятор породжує безліч безперервних сигналів кінцевої тривалості і реалізує відображення вихідних послідовностей кодера в цю безліч сигналів. Для схем двійкової модуляції по кожному вихідному символу кодера обирається один з двох можливих сигналів. У схемах М-ічной модуляції вихідна послідовність декодера розбивається на безлічі, але j символів (де $M = 2^j$) і кожна безліч використовується для вибору одного з М сигналів. У цьому випадку існують різні відображення і вибір одного з них залежить від поставлених цілей. У двох важливих випадках набір з j символів відповідає М- ічному символу блокового коду або n вихідним символам деревовидного коду [4].

Є кілька цікавих окремих випадків. У системах з когерентною демодуляцією (коли є опорна частота несучої) часто використовується схема двійкової модуляції, звана фазовою модуляцією (ФМ). При цьому методі передається символ 1 представляється сигналом $s_1(t) = A(t)\cos\omega_0 t,$ а передаваний символ 0 представляється зворотнім сигналом $s_0(t) = -s_1(t) = A(t)\cos(\omega_0 t + \pi)$. У некогерентних системах визначення знака переданого сигналу неможливо, так що для подання переданих символів 0 і 1 зазвичай

використовується пара взаємно ортогональних сигналів $s_0(t)$ и $s_1(t)$. Випадок М-ічной модуляції в некогерентних системах є простим узагальненням випадку двійкової модуляції: використовується М ортогональних сигналів: $s_0(t)$, $s_1(t)$,…, $s_{M-1}(t)$.

2.1.4 Фізичний канал

Фізичний канал – це вся апаратура і все фізичне середовище, через яке проходить сигнал на шляху від виходу модулятора до входу демодулятора. Фізичний канал не обов'язково являє собою систему зв'язку, що працює в реальному часі; він може бути системою зберігання або запису даних. Зазвичай вихідний сигнал s(t) каналу є сумою вхідного сигналу s(t) помноженого на коефіцієнт передачі, і випадкового шуму n(t). Однак зустрічаються і більш загальні ситуації. Спотворення можуть відбуватися через сильну фільтрацію або наявність декількох шляхів поширення сигналу. Завади можуть викликати завмирання сигналу і приводити до коливання його амплітуди на виході; це еквівалентно зміні в часі параметрів самого каналу. Шум n(t) може бути власним шумом приймача. який моделюється адитивним гауссовским шумом, або промисловим шумом, або, завадою, організованої супротивником [4].

2.1.5 Демодулятор

Демодулятор - Демодулятор це пристрій, який на основі спостереження прийнятого сигналу s (t) оцінює, який з можливих символів був переданий. Імовірність того, що ця оцінка виявиться правильною, залежить від відносини потужності сигналу до потужності шуму в смузі частот, яка використовується, від спотворення сигналу, що викликається фільтрацією і нелінійними ефектами, і від використовуваної схеми демодулятора. У системах з кодуванням демодулятор часто виконує ще одну функцію, яка полягає в передачі декодеру інформації про ступінь надійності оцінки кожного символу. Ця інформація може бути отримана кількома різними

способами, і використовуваний в кожному випадку підхід суттєво залежить від природи шуму n(t). Одна з можливостей виникає, коли діє завада (наприклад, сигнал радіолокатора), присутність якої можна визначити незалежно. Інформація про надійність задається в цьому випадку одним символом, що вказує, включений радіолокатор чи ні. Інша можливість виникає, коли детектор є узгодженим фільтром з відліком на виході, а n(t) адитивним гауссовским шумом. У цьому випадку відношення значення відліку до граничного значення вирішальної схеми є хорошим індикатором надійності рішення [4].

2.1.6 Канал зв'язку

Найбільш важливою частиною структурної схеми системи передачі цифрової інформації для кодера і декодера є складовий або дискретний канал, укладений на рис. 2.1 в штриховую рамку. Розглянемо математичні моделі даного каналу, які найбільш часто зустрічаються [1].

Найпростішою є модель двійкового симетричного каналу (ДСК), яка надається графом на рис. 2.2 [3] і відповідна нагоді використання двійкової модуляції в каналі з адитивним шумом (тобто каналу, в якому вихідний сигнал R (t) дорівнює сумі вхідного сигналу S (t) і шуму n (t)) і жорсткого рішення демодулятора. Входом і виходом даного каналу є набори X={0, 1} і Y={0, 1} из двух возможных двоичных символов. з двох можливих двійкових символів. ДСК також характеризується набором перехідних ймовірностей P(Y │X) , що визначають ймовірність прийому з каналу символу Y при передачі символу X. Для ДСК перехідні ймовірності задаються виразами

$$\begin{matrix} P(Y=0|\mathrm{X}=0)=\mathrm{P}(\mathrm{Y}=1|\mathrm{X}=1)=1-\mathrm{p} \\ P(Y=0|\mathrm{X}=1)=\mathrm{P}(\mathrm{Y}=1|\mathrm{X}=0)=\mathrm{p} \end{matrix}\ , \qquad (2.1)$$

де p – середня ймовірність спотворення символу.

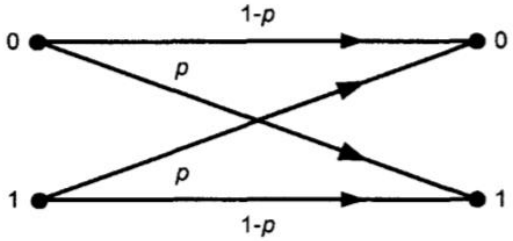

Рисунок 2.2 – Двійковий симетричний канал

Як випливає з представленого опису, основною характеристикою ДСК є ймовірність спотворення символу p. Запишемо вираз, що зв'язує цю величину з раніше згадуваним ставленням сигнал-шум E_s/N_0 для випадку використання двох протилежних сигналів $s_0(t) = -s_1(t)$

$$p = Q\left(\sqrt{2\frac{E_s}{N_0}}\right), \tag{2.2}$$

де Q(x) – функція,що визначається за формулою

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^{\infty} e^{-\frac{1}{2}}dt. \tag{2.3}$$

Більш загальною моделлю каналу є дискретний канал без пам'яті (ДКБП). Входом даного каналу є q-ічні символи $X=\{x_0,x_1,\ldots,x_{q-1}\}$, а выходом – Q-ічні символи $Y=\{y_0,y_1,\ldots,y_{q-1}\}$. Термін «без пам'яті» означає, що вихідний символ каналу залежить тільки від поточного вхідного символу. Для ДКБП перехідні ймовірності постійні в часі і переходи різних символів незалежні. Графічне представлення даної моделі каналу показано на рис. 2.3 [1].

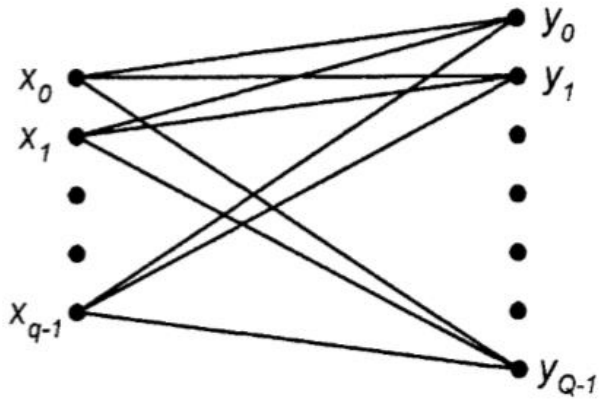

Рисунок 2.3 – Дискретний канал без пам'яті

Окремим випадком ДКБП є q-ічний симетричний канал, іменований далі qСК. Даний канал, що виходить при використанні модулятором q ортогональних сигналів і винесенні жорсткого рішення демодулятором, характеризується дискретним входом $X=\{x_0,x_1,\ldots,x_{q-1}\}$, дискретним виходом $Y=\{y_0,y_1,\ldots,y_{q-1}\}$ і набором перехідних ймовірностей [1].

Останньою розглянутої моделлю каналу є канал з адитивним білим гауссовским шумом (АБГШ), що виходить з ДКБП при нескінченному числі рівнів квантування виходу детектора (тобто квантування відсутне, $Q=\infty$). В даному випадку шум є гауссовской випадковою величиною з нульовим середнім і дисперсією $\sigma^2 = 1/(2E_s/N_0)$. Таким чином, канал з АБГШ характеризується дискретним входом. $X=\{x_0,x_1,\ldots,x_{q-1}\}$, безперервним виходом $Y=\{-\infty,+\infty\}$ і перехідними ймовірностями. Для даної моделі каналу залежність ймовірності помилки p від відносини сигнал-шум E_s/N_0 визначається відповідно до виразу (2.2).

2.1.7 Шум

Серед усіх випадкових процесів особливе місце займає процес з нормальним розподілом (гаусів процес). Справа в тому, що велика кількість дійсних випадкових процесів є гауссовими. Ця обставина пояснюється в відомій теоремі Ляпунова, згідно з якою розподіл суми незалежних

випадкових величин (при деяких досить широких умовах) сходиться до нормального, незалежно від характеру розподілу доданків [8].

Гаусів шум, або гаусів випадковий процес, виникає при підсумовуванні статистично незалежних білих шумів. Він переважає в практичних завданнях. Випадковий процес x(t) називається гаусовим, якщо для будь-якого набору фіксованих моментів часу t_n випадкові величини x_n підкоряються нормальному розподілу. Щільність ймовірностей миттєвих значень x(t) гауссова процесу визначається виразом:

$$w(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (2.4)$$

де μ – средне значення;

σ^2– стандартне (середньоквадратичне) відхилення.

Середнє значення для гауссова розподілу дорівнює математичного сподівання:

$$\mu = \int_{-\infty}^{+\infty} xw(x)dx. \quad (2.5)$$

Стандартне (середньоквадратичне) відхилення:

$$\sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 w(x)dx. \quad (2.6)$$

Отже, щільність ймовірностей гауссова процесу повністю характеризується спектральною щільністю, по якій можна визначити значення дисперсії процесу. На рисунку 2.4 [7] показана залежність форми розподілу Гаусса від середньоквадратичного відхилення [7].

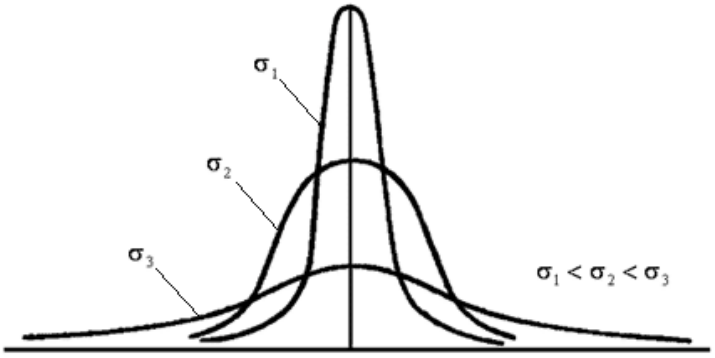

Рисунок 2.4 – Залежність форми розподілу Гаусса від середньоквадратичного відхилення

Заваду, що являє собою випадковий процес з рівномірним спектром, тобто

$$\mathrm{A(f)} = \mathrm{A_0} = \mathrm{const}, \tag{2.7}$$

Називають білим шумом. Потужність білого шуму в смузі F дорівнює

$$E_F = \int_0^F A(f)df = A_0F. \tag{2.8}$$

По-іншому, білий шум (рис.2.5) [7] можна визначити як стаціонарний випадковий процес x(t) з постійною спектральною щільністю $G(w) = \sigma^2$, що дорівнює дисперсії значень D(x) – усі спектральні складові білого шуму мають однакову енергію (звідси аналогія з білим кольором, який містить всі кольори видимого спектру).

Як вже було сказано вище, за своїм фізичним змістом спектральна щільність - це потужність процесу, яка припадає на 1 Гц смуги частот. Але тоді ідеального білого шуму на практиці не може існувати, так як для нього мало б виконуватися умова

$$R(0) = \int_0^{+\infty} G(w)dw = \frac{\sigma^2}{2}\delta(0) = +\infty, \tag{2.9}$$

де $\delta(0)$ – дельта-функція Дірака. Таким чином, потужність білого шуму дорівнює нескінченності, а значення шуму не корельовані для будь-яких τ≠0, так як кореляційна функція є дельта-імпульс. Тим не менш, багато завади в радіотехніці, в техніці зв'язку і в інших галузях розглядають як білий шум, якщо виконується наступне співвідношення між шириною спектрів корисних сигналів B_s і шумів B_n і спектральна щільність шумів слабо змінюється в інтервалі спектра сигналу [7].

$$\frac{B_s}{B_n} \ll 1. \qquad (2.10)$$

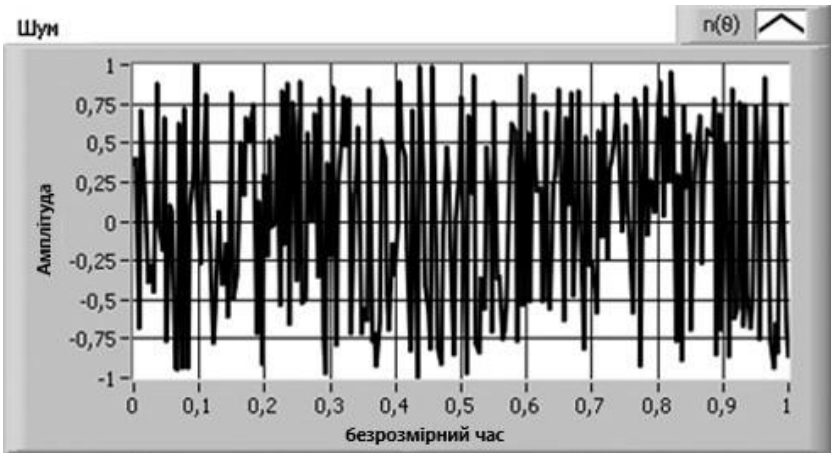

Рисунок 2.5 – Вибірка білого шуму

Прикладом білого шуму може служити тепловий шум. Відповідно до формули Найквіста потужність теплового шуму, що припадає на смугу довжиною B, дорівнює

$$E_F = 4kTB, \qquad (2.11)$$

де k – постійна Больцмана;

T – абсолютна температура.

Таким чином, $A_0 = akT$.

Періодичний випадковий шум являє собою результат підсумовування синусоїд з однаковими амплітудами і випадковими фазами. Шум містить усі частоти, які можуть бути представлені цілим числом періодів на певному числі вибірок [8].

Розподіл періодичного шуму при великій кількості вибірок прагне до гаусового шуму.

Часто періодичний шум розглядають як вибірку гаусова шуму, обмежену за величиною наступними значеннями

$$\begin{cases} A\left(\frac{N}{2}-1\right), N=2k, k=1,2,\dots; \\ A\left(\frac{N-1}{2}\right), N=2k-1, k=1,2,\dots, \end{cases} \quad (2.12)$$

де A – задана амплітуда спектра;

N – вибірка.

Таким чином, періодичний випадковий шум має імпульсний характер, а, отже, дискретну частотну характеристику (рис. 2.6) [6].

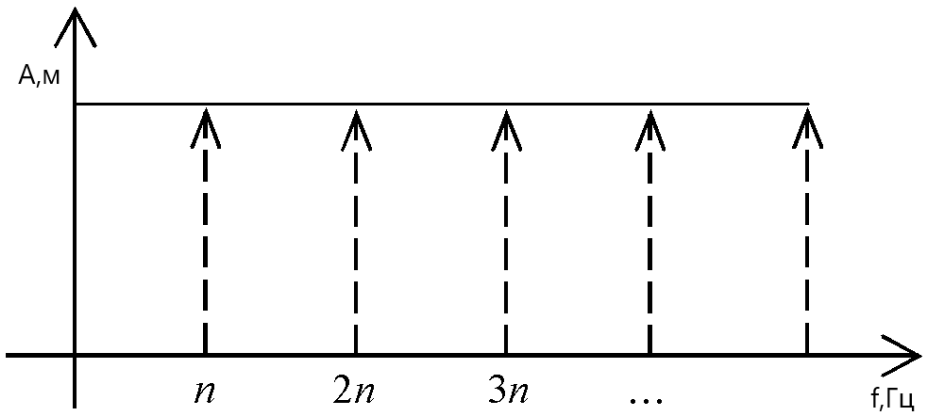

Рисунок 2.6 – Частотна характеристика періодичного випадкового шуму

Із рис. 2.7 [6] видно, що проблема шумопоглинання виникає при наявності трьох необхідних елементів: по-перше, повиннно бути джерело шумів; по-друге, повинна бути схема-приймач, чутлива до шумів; по-третє,

необхідна наявність каналу зв'язку для передачі шумів від джерела до приймача.

Рисунок 2.7 – Принципова схема системи з шумами

При аналізі проблеми шумів, перш за все, слід визначити, що є джерелом шумів, що служить їх приймачем і яким чином джерело і приймач пов'язані один з одним. Звідси випливає, що можливі три способи усунення проходження шумів:

- зменшення шуму в джерелі;
- створення приймача, нечутливого до шумів;
- мінімізація передачі шумів через канал зв'язку.

У деяких випадках необхідно застосовувати два або навіть усі три зазначених способу придушення шумів.

2.1.8 Декодер

Операція, яка здійснюється декодером, протилежна операції кодера. Оскільки послідовність символів, що породжується демодулятором, може містити помилки, декодер повинен виконувати відображення істотно складніше, ніж кодер. Незважаючи на те що в принципі це відображення можна задати оптимально, таке завдання швидко стає недоцільним при зростанні обсягу коду. На практиці часто доводиться задовольнятися неоптимальними методами. Щоб зробити декодування можливим, потрібно розробити обчислювальні процедури, які його реалізують. Алгоритми декодування класифікуються відповідно до того, чи застосовуються вони до блоковим або до деревовидних кодів. Тут розглядаються такі два основні класи алгоритмів декодування блокових кодів:

- процедури, засновані на елементарних властивостях кодів, які включають перестановочне декодування, декодування Меггітта і граничне декодування;

- алгебраїчні процедури, засновані на більш тонких властивостях алгебраїчної структури деяких кодів; вони, по суті, використовують рішення систем алгебраїчних рівнянь. Розглядаються також наступні чотири великі класи алгоритмів декодування деревовидних кодів;

- алгоритм Вітербі (декодування по максимуму правдоподібності);

- алгоритми послідовного декодування; порогове декодування; табличне декодування [4].

Інформація про надійність (або так звані м'які рішення) використовується як для блокових, так і для деревовидних кодів. У більшості алгоритмів декодування деревовидних кодів ця інформація використовується безпосередньо. Реалізація м'яких рішень в блокових кодах не так проста і зазвичай вимагає значної модифікації алгоритму [4].

2.2 Засоби програмного забезпечення

2.2.1 Загальна характеристика програмного забезпечення

У наш час у зв'язку з розвитком інформаційних технологій з'явилися так звані системи комп'ютерної математики, або їх ще називають математичні пакети, які полегшують виконання різних математичних задач, допомагають перевірити рішення задачі за допомогою комп'ютерної програми. Набагато скорочується час виконання завдань різної складності. Для сотень тисяч фахівців в різних галузях промисловості, зайнятих інженерними та науковими дослідженнями, системи комп'ютерної математики забезпечили чудову середовище для організації обчислень. Тому знайомство з основами організації математичних пакетів може бути корисно як фахівцям, що приступає до освоєння цієї системи, так і студентам вузів за

самимм різнимм спеціальностями. Вони мають надзвичайно широкий набір засобів, що переводять складні математичні алгоритми в програми, так звані елементарні функції і величезна кількість неелементарних, алгебраїчних та логічних операцій. Більшість вправ з курсу вищої математики може бути вирішено за допомогою всього лише однієї команди. Можна обчислювати інтеграли, вирішувати диференціальні рівняння, звичайні рівняння і системи лінійних рівнянь. Надано широкий вибір роботи з матрицями, векторами. Можливо побудова двовимірних і тривимірних графіків. Існує кілька математичних пакетів, таких як MathCad, MatLab, Mathematica, Maple, Statistica та інші табл.2.1 [4].

Таблиця 2.1 – Порівняльний аналіз систем комп'ютерної математики

Критерій порівняння	Mathcad	MATLAB	Mathematica	Maple
Інтерфейс	Типу "wysiwyg". Набір виразів походить від позиції курсора. Виразів з клавіатури доводиться вводити відносно небагато, так як в командному вікні є різні палітри інструментів.	Три вікна: командне вікно, всі змінні і їх типи і вікно підказок. Є рядок запрошення, позначається знаком ">>". На відміну від Mathcad всі функції доводиться вводити з клавіатури.	Рядок запрошення на відміну від MATLAB розділено на дві області: введення і виведення, які складають разом область всього виразу. Область введення можна редагувати. Також є палітра з грецькими буквами, різними символами і панель матаналізу.	Інтерфейс користувача підтримує концепцію робочих аркушів ("worksheets"), які об'єднують текст, вхідні команди, висновок і графіку в одному документі. Програма дозволяє одночасно працювати з декількома робочими листами і встановлювати між ними динамічні зв'язки, тобто переводити обчислення з одного аркуша на інший.

Продовження таблиці 2.1

Критерій порівняння	Mathcad	MATLAB	Mathematica	Maple
Рішення рівнянь	Розрізняє рішення рівнянь і систем рівнянь. Команди можна набирати з клавіатури, можна вставляти з меню.	Вирішує рівняння і системи рівнянь функцією з різними параметрами.	Містить кілька функцій для вирішення рівнянь і систем рівнянь. Функції можуть знаходити коріння рівнянь з параметром.	Вирішує рівняння і системи рівнянь
Робота з масивами і матрицями	Надано достатній набір функцій для проведення різних операцій з матрицями і векторами. Деякі операції можна брати з відповідної палітри, інші - вводити з клавіатури або вставляти з меню Вставка-функції.	Аналогічно Mathematica матриці і вектора формуються за допомогою списку елементів. Функції вводяться з клавіатури..	Багатовимірний набір даних створюється за допомогою списку, який вводиться з клавіатури. Також з клавіатури вводяться і функції для роботи з матрицями і векторами.	Аналогічно Mathematica матриці і вектора формуються за допомогою списку елементів. Функції вводяться з клавіатури.
Математичні оператори	Наведено в таблиці цілий ряд операторів, як простих типу додавання, так і обчислення суми, твори, інтегралів і похідних і т.д., які можна вводити з клавіатури або вставляти з відповідної палітри.	Тут на відміну від Mathcad всі оператори вводяться з клавіатури у вигляді окремих символів і функцій. Дано відносно детальний список операторів.	Також, як і в MATLAB операторі доводиться вводити з клавіатури, але деякі можна знайти і на палітрі інструментів.	Крім функцій в математичних системах для запису математичних виразів використовуються спеціальні знаки - оператори. Наприклад, обчислення квадратного кореня часто записується за допомогою його спеціального знака - V.

Продовження таблиці 2.1

Критерій порівняння	Mathcad	MATLAB	Mathematica	Maple
Вбудовані функції	Построены по принципу всех функций: название функции и параметры в скобках. Можно выделить функции упрощения выражения, раскрытия скобок, тригонометрические и целый ряд других.	Здесь в основном используются только функции, которые вводятся с клавиатуры.	Приведено множество функций различного назначения с различным числом параметров. Помогают пользователю в решении различного характера задач.	Maple имеет множество встроенных функций, включенных в его ядро и в пакеты.
Програмування	Надано шаблони для створення програм і підпрограм. В якості вихідного значення вказується останнє значення, обчислене програмою. Також всередині програми можна використовувати функції, описані раніше. Програми пишуться в тому ж файлі, що і всі обчислення.	Тут програми створюються у вигляді окремих М - файлів. Якщо написати програму якої-небудь функції, то цю функцію можна буде використовувати як стандартну. Також в програму можна вставляти коментарі.	Можна створювати різні функції і оперувати з ними. Дозволяє всередині одного блоку введення створювати програми. Результатом буде останнім обчислене значення. Програми пишуться в рядок.	Є можливість створювати власні функції, процедури на мові програмування, що нагадує Паскаль.
Графічні можливості	Графіки будуються на основі наявних шаблонів. Основні види: графік у декартовій площині, в полярній системі координат, тривимірний у вигляді гладкої поверхні, у вигляді контурних кривих і т.д. Спочатку задається функція графіка, діапазон, потім будується сам графік, який можна редагувати.	Функція графіка створюється з командного рядка. Графіки створюються на формах в певній системі координат. У команді побудови можна вказувати властивості графіка.	Функція, яка малює графік, закінчується на "PLOT" в двомірному випадку, і "PLOT 3D" в тривимірному випадку. Щоб побудувати графік, потрібно спочатку поставити функцію. Також можна і редагувати графік.	У бібліотеці ядра Maple є всього чотири графічні команди: plot - для виведення плоскою або 2D-графіки, plot3d - для виведення просторової або 3D-графіки, smartplot і smartplot3d - для "швидкого" виведення графіки.

2.2.2 Програма Maple

Програма Maple – свого роду патріарх в сімействі систем символьної математики і до сих пір є одним з лідерів серед універсальних систем символьних обчислень. Вона надає користувачеві зручне інтелектуальне середовище для математичних досліджень будь-якого рівня і користується особливою популярністю в науковому середовищі. Відзначимо, що символьний аналізатор програми Maple є найбільш сильною частиною цього ПО, тому саме він був запозичений і включений в ряд інших CAE-пакетів, таких як MathCad і MatLab, а також до складу пакетів для підготовки наукових публікацій Scientific WorkPlace і Math Office for Word .

Пакет Maple – спільна розробка Університету Ватерлоо (шт. Онтаріо, Канада) і Вищої технічної школи (ETHZ, Цюріх, Швейцарія). Для його продажу була створена спеціальна компанія - Waterloo Maple, Inc., яка, на жаль, більше прославилася математичної опрацюванням свого проекту, ніж рівнем його комерційної реалізації. В результаті система Maple раніше була доступна переважно вузькому колу професіоналів. Зараз ця компанія працює спільно з більш процвітаючою в комерції та в опрацюванні користувальницького інтерфейсу математичних систем фірмою MathSoft, Inc. – створювачкою вельми популярних і масових систем для чисельних розрахунків MathCad, що стали міжнародним стандартом для технічних обчислень.

Maple надає зручну середу для комп'ютерних експериментів, в ході яких досліджуються різні підходи до задачі, аналізуються приватні рішення, а при необхідності програмування відбираються фрагменти особливої швидкості. Пакет дозволяє створювати інтегровані середовища за участю інших систем і універсальних мов програмування високого рівня. Коли розрахунки проведені і потрібно оформити результати, то можна використовувати засоби цього пакета для візуалізації даних і підготовки ілюстрацій для публікації. Для завершення роботи залишається підготувати друкований матеріал (звіт, статтю, книгу) прямо в середовищі Maple, а потім

можна приступати до чергового дослідження. Робота проходить інтерактивно - користувач вводить команди і тут же бачить на екрані результат їх виконання. При цьому пакет Maple зовсім не схожий на традиційну середу програмування, де потрібна жорстка формалізація всіх змінних і дій з ними. Тут же автоматично забезпечується вибір відповідних типів змінних і перевіряється коректність виконання операцій, так що в загальному випадку не потрібен опис змінних і запис суворої формалізації.

Пакет Maple складається з ядра (процедур, написаних на мові С і добре оптимізованих), бібліотеки, написаної на Maple-мові, і розвиненого зовнішнього інтерфейсу. Ядро виконує більшість базових операцій, а бібліотека містить безліч команд - процедур, що виконуються в режимі інтерпретації. Інтерфейс Maple заснований на концепції робочого поля (worksheet) або документа, що містить рядки введення-виведення і текст, а також графіком.

Робота з пакетом відбувається в режимі інтерпретатора. У рядку введення користувач задає команду, натискає кнопку Enter і отримує результат - рядок (або рядка) виведення або повідомлення про помилково введеної команді. Тут же видається запрошення вводити нову команду і т.д.

2.2.3 Система Mathematica

Система Mathematica, створена років десять тому в своїх останніх версіях (Mathematica 3.0, Mathematica 4.0) має надзвичайно широкий набір засобів, що переводять складні математичні алгоритми в програми. Всі так звані елементарні функції і величезна кількість неелементарних. Алгебраїчні і логічні операції. По суті справи всі алгоритми, що містяться в курсі вищої математики провідного технічного вузу закладені в пам'ять комп'ютерної системи Mathematica. Це означає, між іншим, що більшість вправ з курсу вищої математики може бути вирішено за допомогою всього лише однієї команди. Дійсно, всі вправи з лінійної алгебри (включаючи такі нетривіальні речі як приведення квадратичних форм до канонічного вигляду, приведення

лінійного оператора до жорданової формі). Всі вправи з аналізу, теорії диференціальних рівнянь (як звичайних, так і в приватних похідних). За допомогою системи Mathematica можна обчислювати інтеграли (визначені і невизначені), вирішувати диференціальні рівняння (чисельно і аналітично). Mathematica знає про інтеграли і диференціальні рівняння більше, ніж будь-який товстий довідник. Крім того, Mathematica не тільки дає остаточну відповідь, але може описати проміжні обчислення (наприклад, розкладання правильної раціональної функції в суму елементарних дробів, що потрібно при інтегруванні раціональних функцій).

Mathematica має потужний графічний пакет. З її допомогою можна будувати графіки дуже складних функцій однієї і двох змінних.

2.2.4 Програмний засіб MathCAD

Mathcad – програмний засіб, середовище для виконання на комп'ютері різноманітних математичних і технічних розрахунків, забезпечена простим в освоєнні і в роботі графічним інтерфейсом, яка надає користувачеві інструменти для роботи з формулами, числами, графіками і текстами. У середовищі Mathcad доступні більше сотні операторів і логічних функцій, призначених для чисельного і символьного розв'язання математичних задач різної складності. Меню в Mathcad не являє собою нічого незвичайного: як і в багатьох інших програмах є різні панелі інструментів, панель форматування. Крім того є панель "Математика", яка включає в себе такі панелі як "Калькулятор", "Графіка", "Матриці", "Обчислення", "Обчислення", "Логічний", "Програмування", "Грецький" і "Символьний" . Ці панелі містять різні символи, які не набираються з клавіатури, а також функції.

2.2.5 Система Mathlab Simulink

MATLAB – це інтерактивна система, основним об'єктом якої є масив, для якого не потрібно вказувати розмірність. Це дозволяє вирішувати різні обчислювальні завдання, пов'язані з векторно-матричними формулюваннями,

істотно скорочуючи час, який знадобився б для програмування на скалярних мовах типу С або FORTRAN.

Система MATLAB – це одночасно і операційне середовище і мова програмування. Одна з найбільш сильних сторін системи полягає в тому, що на мові MATLAB можуть бути написані програми для багаторазового використання. Користувач може сам написати спеціалізовані функції і програми, які оформляються у вигляді М-файлів [9].

Пакет прикладних програм MATLAB пр1изначений для вирішення різноманітних математичних і технічних завдань, виконання математичних розрахунків, програмування та імітаційного моделювання.

Simulink є додаткове програмне забезпечення до MATLAB, що дозволяє значно спростити створення моделей систем з використанням принципу візуального програмування: модель будь-якої системи збирається з безлічі доступних готових блоків, що містяться в бібліотеці стандартних блоків, які з'єднуються між собою певним чином. При цьому абсолютно не потрібне знання високорівневої інтерпретованої мови програмування MATLAB, хоча при цьому сам Simulink не виключає можливості використання цієї мови. Крім бібліотеки стандартних блоків Simulink також містить велику кількість додаткових бібліотек блоків для різних областей застосування: цифрової обробки сигналів, завдань галузі зв'язку, лінійної і нелінійної обробки аналогових сигналів. Більш того, існує можливість створення своїх власних блоків, що робить дану систему моделювання розширюваною. Серед всіх доступних бібліотек Simulink особливо важливою для моделювання систем передачі дискретних повідомлень є бібліотека Communications System Toolbox, що включає в себе такі розділи:

- Communications Sources – джерела різних сигналів, таких як джерела випадкових даних (Random Data Sources), генератори шуму (Noise Generators), генератори послідовностей (Sequence Generators).
- Error Detection and Correction - блоки виявлення і виправлення помилок, що містять блокові, згорткові і CRC коди.

- Communications Sinks – блоки збору, обробки і відображення інформації, такі як лічильники помилок (Error Rate Calculation), осцилографи і індикаторні діаграми дискретного часу.
- Modulation – модулятори і демодулятори цифрових сигналів (амплітудна, частотна, фазова маніпуляція, КАМ, АФМ і т.д.).
- Communications Filters – телекомунікаційні фільтри різного призначення.
- Channels – блоки, що моделюють дискретні і безперервні канали зв'язку, зокрема, двійковий симетричний канал (Binary Symmetric Channel), канал з адитивним білим Гауссовским шумом (AWGN Channel).

2.3 Висновки за розділом 2

Для проведення експерименту з аналізу ефективності алгоритмів завадостійкого кодування було обрано систему комп’ютерної математики MATLAB, зважаючи на його поширеності як засобу моделювання, зручності використання різних розширень і їх різноманітності, а так само загальної зручності та доступності. Завдяки середі імітаційного моделювання Simulink, можна побудувати схему статистичного імітаційного моделювання функціонування завадостійкого кодека. В ній існують бібліотеки завдяки яким можна провести аналіз ефективності використовуючи вже вбудовані функції та блоки.

3 ЕКСПЕРИМЕНТ З АНАЛІЗУ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ЗАВАДОСТІЙКОГО КОДУВАННЯ

3.1 Розробка моделі перевірки на ефективність

Для аналізу ефективності алгоритмів завадостійких кодів було побудовано декілька схем з застосування блоків Simulink [10].

Двійковий генератор Бернуллі (рис.3.1) [10].

Рисунок 3.1 – Графічне зображення блоку

Блок Двійковий генератор Бернуллі генерує випадкові двійкові числа, використовуючи розподіл Бернуллі. Розподіл Бернуллі з параметром p дає нуль з імовірністю p і один з ймовірністю 1-p. Розподіл Бернуллі має середнє значення 1-p і дисперсію p (1-p). Імовірність нульового параметра вказує p і може бути будь-яким дійсним числом між нулем і одиницею.

Атрибути вихідного сигналу

Вихідний сигнал може бути вектором стовпчика або рядка, двовимірною матрицею або скаляром. Кількість рядків у вихідному сигналі відповідає кількості вибірок в одному кадрі і визначається параметрами зразків на кадр. Кількість стовпців у вихідному сигналі відповідає кількості каналів і визначається кількістю елементів в ймовірності нульового параметра.

Параметри:

1. Імовірність нуля. Імовірність, з якою відбувається нульовий вихідний сигнал. Імовірність встановлюється між 0 і 1. Кількість елементів в

ймовірності параметра відповідає числу незалежних вихідних каналів з блоку.

2. Джерело вихідного зерна. Джерело вихідного зерна для генератора випадкових чисел.

3. Початкове зерно. Початкове значення зерна для генератора випадкових чисел. Вкажіть зерно як невід'ємний цілочисельний скаляр. Початкове зерно є, якщо для параметра початковий параметр зерна заданий параметр.

4. Час вибірки. Час між кожним зразком стовпця вихідного сигналу.

5. Зразки на кадр. Кількість вибірок на кадр в одному каналі вихідного сигналу. Вкажіть вибірки на кадр в якості позитивного цілого числа скаляра.

Лічильник помилок (рис.3.2) [10].

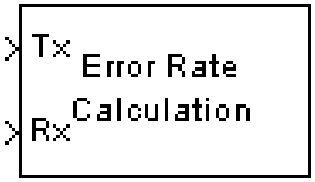

Рисунок 3.2 – Графічне зображення блоку

Обчислює коефіцієнт помилок в послідовності бітів або символів. Для цього блок знаходить відношення розбіжностей символів на входах Rx і Tx до загальної кількості переданих символів. Як правило, Rx і Tx є сигнали на вході і виході системи, причому число входів блоку не обмежується двома, а може бути від 2 до 4. В якості вхідних сигналів даний блок підтримує скалярні чи векторні (векторстолбец) величини, причому частоти сигналів на Rx і Tx входах повинні збігатися. Якщо ж на один вхід подається скалярна величина, а на інший вхід - вектор, то відбувається послідовне порівняння даної скалярної величини з кожним з елементів вектора. На виході блоку виходить вектор з трьох значень:

- коефіцієнта помилок;

- загальне число помилок, тобто число розбіжностей значень на вхідних портах;

- загальне число порівнянь, зроблених блоком.

Двійковий симетричний канал (рис.3.3) [10].

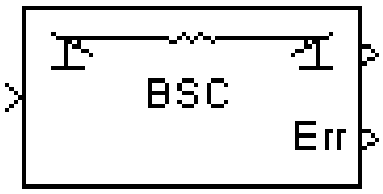

Рисунок 3.3 – Графічне зображення блоку

Блок Двійковий симетричний канал вводить виконавчі помилки в сигналі, переданому через цей канал.

Вхідний порт є переданий двійковий сигнал. Цей блок приймає скалярний або векторний вхідний сигнал. Блок обробляє кожен векторний елемент незалежно і вводить помилку в даному місці з ймовірністю Імовірність помилки. Блок Двійковий симетричний канал вводить виконавчі помилки в сигналі, переданому через цей канал [9].

Вхідний порт є переданий двійковий сигнал. Цей блок приймає скалярний або векторний вхідний сигнал. Блок обробляє кожен векторний елемент незалежно і вводить помилку в даному місці з ймовірністю.

Перший вихідний порт - це двійковий сигнал, який обробляє канал. Другий вихідний порт - це вектор помилок, які вводить блок. Щоб придушити другий вихідний порт, очистіть вектор помилки виведення.

Параметри:

1. Імовірність помилки. Імовірність виникнення двійковій помилки. Задайте значення цього параметра між 0 і 1.

2. Початкове зерно. Початкове значення зерна для генератора випадкових чисел.

3. Вектор помилки виведення. При виборі цього блоку блок виводить вектор помилок.

Дисплей (рис.3.4) [10].

Рисунок 3.4 – Графічне зображення блоку

Блок дисплею підключається до сигналу в моделі і відображає його значення під час моделювання. Можна редагувати параметри блоку дисплея під час імітації.

Канал аддитивного білого гауссовского шуму (рис.3.5) [10].

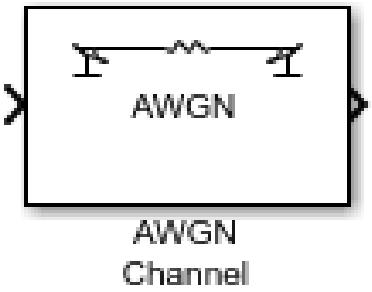

Рисунок 3.5 – Графічне зображення блоку

Додає до вхідного сигналу білий гауссовский шум. Вхідний сигнал може бути реальним або складним. Цей блок підтримує багатоканальну обробку.

При використанні будь-якого з режимів дисперсії зі складними входами значення дисперсії розподіляються порівну між реальними і уявними компонентами вхідного сигналу [10].

Параметри:

1. Eb / No (дБ) - Відношення енергії інформаційного біта на символ до спектральної щільності потужності шуму.

Відношення енергії інформаційного біта на символ до спектральної щільності потужності шуму в децибелах, заданий як скаляр або вектор. Енергія інформаційного біта являє собою величину без канального кодування.

2. Es / No (дБ) - відношення енергії інформаційного символу на символ до спектральної щільності потужності шуму.

Відношення енергії інформаційного символу на символ до спектральної щільності потужності шуму в децибелах, заданий як скаляр або вектор. Енергія інформаційного біта являє собою величину без канального кодування.

3. SNR (дБ) - Відношення потужності сигналу до потужності шуму

Відношення потужності сигналу до потужності шуму в децибелах, заданий як скаляр або вектор.

4. Кількість біт на символ - Кількість біт в кожному вхідному символі

Кількість біт в кожному вхідному символі, заданий як скаляр або вектор.

5. Потужність вхідного сигналу, пов'язана з 1 Ом (Вт) - Середня квадратна вхідна потужність.

Середня квадратна потужність входу в ватах, задана як скаляр або вектор:

- якщо в режимі Eb / No або Es / No, параметр являє собою середню квадратну потужність вхідних символів;

- коли обрано SNR, цей параметр являє собою середню квадратну потужність вхідних вибірок.

6. Період символу (s) - тривалість інформаційного канала.

Тривалість символу інформаційного каналу в секундах, задана як позитивний скаляр або вектор. Тривалість інформаційного каналу вимірюється без канального кодування [9].

3.1.1 Код Хеммінга

Кодер Хеммінга (рис.3.6) [10].

Кодер Хеммінга - використовується для створення кодової комбінації коду Хеммінга на основі двійкового вектора інформаційних розрядів.

Як параметри кодера Хеммінга вказується всього два числа - N і K, де N - загальна довжина кодової комбінації, включаючи перевірочні та інформаційні елементи, має дорівнювати 2^M-1, де M більше або дорівнює 3, а K - кількість інформаційних розрядів кодової комбінації , K = N-M.

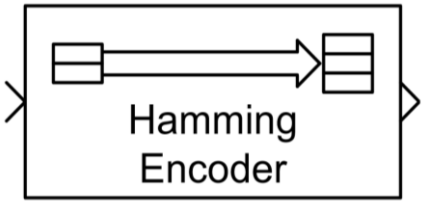

Рисунок 3.6 – Графічне зображення блоку

В якості вихідних даних блок приймає векторстовбець інформаційних символів довжиною K і повертає векторстовбець кодової комбінації довжиною N.

Декодер Хеммінга (рис.3.7) [10].

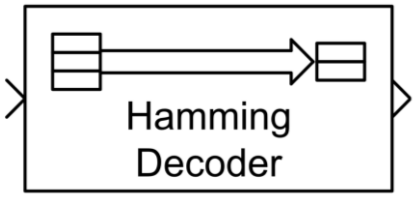

Рисунок 3.7 – Графічне зображення блоку

Декодер Хеммінга, здійснює відновлення даних в кодових комбінаціях. Всі параметри декодера повинні повністю збігатися з параметрами кодера Хеммінга.

В якості вихідних даних блок приймає векторстовбець кодової комбінації довжиною N і повертає векторстовбець інформаційних символів довжиною K.

За допомогою описаних блоків були побудовані дві схеми для завадостікого кодування (рис.3.8) та (рис.3.9) [10].

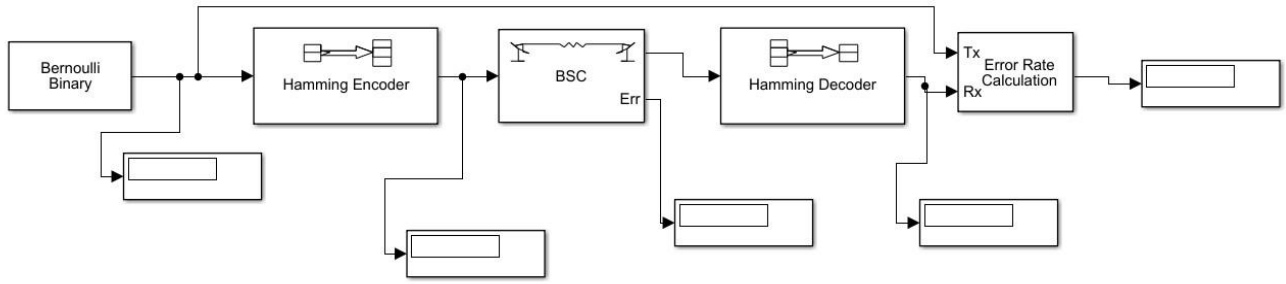

Рисунок 3.8 – Схема для коду Хеммінга з ДСК

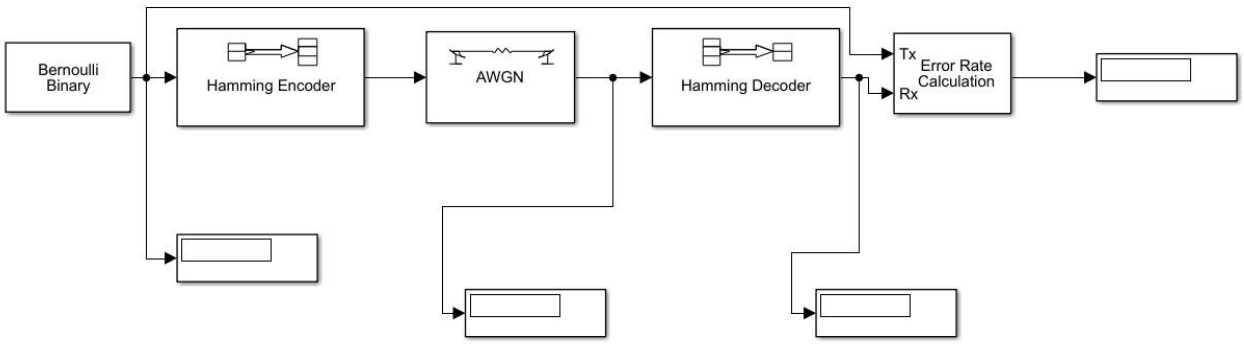

Рисунок 3.9 – Схема для коду Хеммінга з АБГШ

3.1.2 Код Ріда-Соломона

Двійковий вхід кодера Ріда-Соломона (рис.3.10) [10].

Кодує повідомлення у вхідному векторі за допомогою кодера (N, K) Ріда-Соломона з поліномом генератора вузького сенсу. Цей блок приймає вхідний сигнал вектора стовпця з цілим числом, кратним бітам $K \cdot ceil\,(log2\,(N + 1))$. Кожна група вхідних біт $K \cdot ceil\left(log2\,(N + 1)\right)$являє собою одне слово повідомлення, яке повинно бути закодовано.

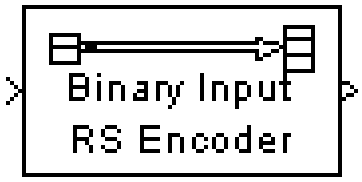

Рисунок 3.10 – Графічне зображення блоку

Двійковий вихід декодера Ріда-Соломона (рис.3.11) [10].

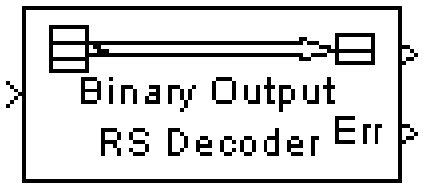

Рисунок 3.11 – Графічне зображення блоку

Декодує вхідний сигнал з використанням декодера Ріда-Соломона (N, K) за допомогою многочлена генератора з вузьким сенсом. Цей блок приймає вхідний сигнал вектора стовпця з цілим числом, кратним біт $N \cdot ceil\,(log2\,(N + 1))$. Кожна група вхідних біт $N \cdot ceil\,(log2\,(N + 1))$ представляє одне прийняте слово, яке повинно бути декодовано.

За допомогою описаних блоків були побудовані дві схеми для завадостікого кодування (рис.3.12) та (рис.3.13) [10].

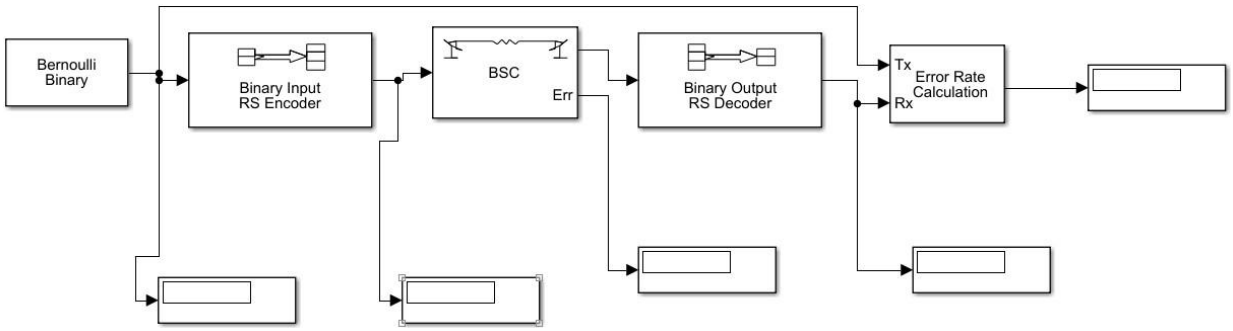

Рисунок 3.12 – Схема для коду Ріда-Соломона з ДСК

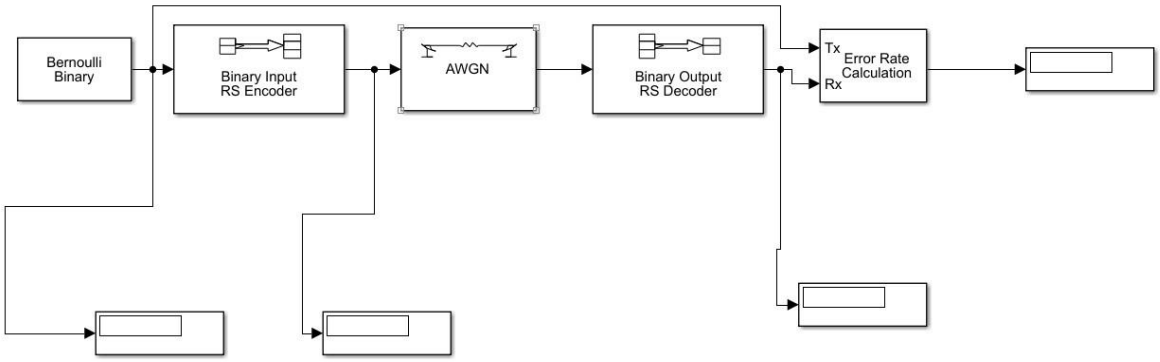

Рисунок 3.13 – Схема для коду Ріда-Соломона з АБГШ

3.1.3 Код Боуза - Чоудхурі – Хоквінгема

Кодер БЧХ (рис.3.14) [10].

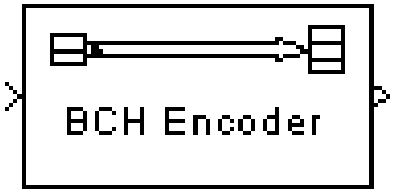

Рисунок 3.14 – Графічне зображення блоку

Кодер БЧХ - використовується для створення кодової комбінації коду Боуза-Чоудхурі-Хоквінгема на основі двійкового вектора інформаційних розрядів.

Як параметри кодера БЧХ вказується всього два числа - N і K, де N - загальна довжина кодової комбінації, включаючи перевірочні та інформаційні елементи, має дорівнювати 2M-1, де 3≤M≤16, а K - кількість інформаційних розрядів кодової комбінації , причому не для кожного значення N можна знайти K. для пошуку пар N і K рекомендується використовувати довідкові таблиці для функції bchenc табл.3.1 [10].

Код можна скоротити, встановивши параметр довжини скороченого повідомлення S. У цьому випадку використовуйте значення N і K повної довжини, щоб вказати код (N, K), який скорочений до коду (N - K + S, S).

Таблиця 3.1 – Функція bchenc

n	k	$t_{випр}$
7	4	1
15	11	1
15	7	2
15	5	3
31	26	1
31	21	2
31	16	3
31	11	5
31	6	7

Кожне вхідне слово повідомлення являє собою двійковий вектор довжиною 6, який представляє 2 трьохбітових цілих числа. Кожне відповідне вихідне кодове слово являє собою двійковий вектор довжиною 21, який представляє 7 трьохбітових цілих чисел.

Декодер БЧХ (рис.3.15) [10].

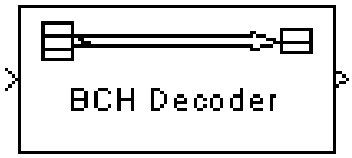

Рисунок 3.15 – Графічне зображення блоку

Декодер для коду Боуза-Чоудхурі-Хоквінгема, який здійснює відновлення даних в кодових комбінаціях. Всі параметри декодера повинні повністю збігатися з параметрами кодера БЧХ.

В якості вихідних даних блок приймає векторстовбець кодової комбінації довжиною N і повертає векторстовбець інформаційних символів довжиною K.

За допомогою описаних блоків були побудовані дві схеми для завадостікого кодування

Для отримання інформації про помилки, слід вибрати Output number of corrected errors. В цьому випадку у блоку декодера з'явиться другий вихід, який буде показувати кількість помилок, що виникли в результаті декодування. Негативне ціле число сигналізує про те, що блок виявив число помилок, що перевищує виправляє здатність коду (рис.3.16) та (рис.3.17) [10].

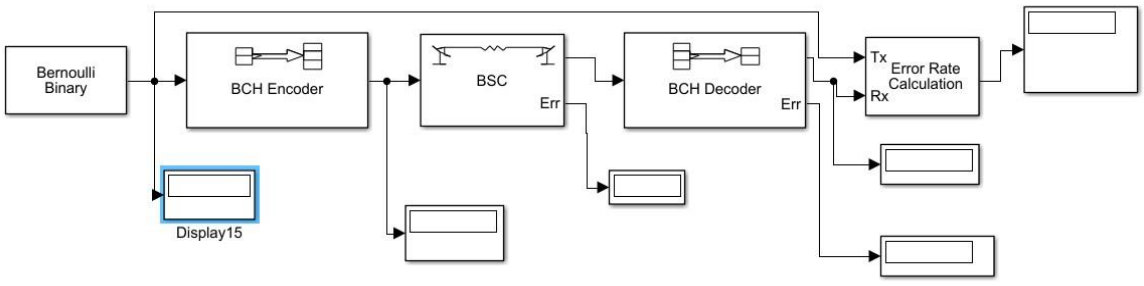

Рисунок 3.16 – Схема для коду БЧХ з ДСК

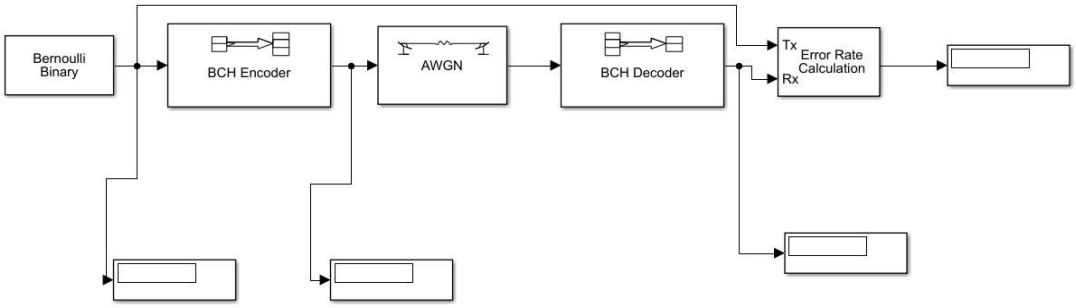

Рисунок 3.17 – Схема для коду БЧХ з АБГШ

3.1.3 Код з малою щільністю перевірок на парність

Кодер LDPC (рис.3.18) [10].

Цей блок підтримує кодування кодів парності з низькою щільністю (LDPC), які представляють собою лінійні коди управління помилками з розрідженими матрицями контролю парності і довгими блоками, які можуть досягти продуктивності поблизу межі Шеннона.

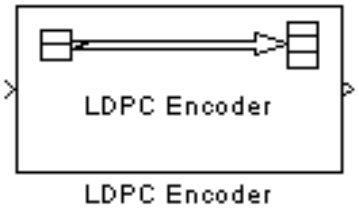

Рисунок 3.18 – Графічне зображення блоку

Як вхід, так і вихід є сигнали дискретного часу. Ставлення вихідного часу вибірки до часу вибірки становить k / n. Вхід повинен бути реальним векторних сигналом k × 1.

Вихідний сигнал успадковує тип даних від вхідного сигналу, а вхід повинен бути двійковим (0 або 1).

Єдиним параметром є матриця перевірки на парність. Цей блок може прийняти розріджену матрицю з розміром n -k на n (де n> k> 0) дійсних чисел. Усі ненульові елементи повинні бути рівними 1. Верхня межа для значення n становить 231-1

Значення за замовчуванням - це матриця перевірки на перевірку парності коду LDPC з половиною швидкості з стандарту DVB-S.2.

H = dvbs2ldpc (r) повертає матрицю перевірки на перевірку парності коду LDPC за допомогою коду швидкості r від стандарту DVB-S.2. H – рідша логічна матриця.

Можливі значення для r – 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9 та 9/10. Довжина блоку коду - 64800.

Декодер LDPC (рис.3.19) [10].

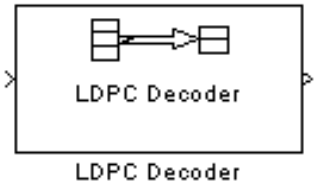

Рисунок 3.19 – Графічне зображення блоку

Декодер LDPC.

Цей блок реалізує алгоритм передачі повідомлень для декодування кодів з низькою щільністю перевірки на рівність (LDPC), які є кодами керування лінійними помилками з матрицями з нестабільною перевіркою парності та довгою довжиною блоків, які можуть досягти продуктивності в межах межі Шеннона.

Блок декодера LDPC призначений для:

- декодування загальних бінарних кодів LDPC, де немає шаблонів у матриці перевірки на пари;

- виконання рядів ітерацій, які вказуються або запускаються, доки всі перевірки на перевірку парності не задовольняються;

- виведення жорстких або м'яких рішень (логарифмічні коефіцієнти правдоподібності) для розшифруваних бітів;

- (n - k) і n – кількість рядків і стовпчиків, відповідно, в матриці перевірки на паритет.

Цей блок приймає дійсний векторний вхідний сигнал з кодом n × 1 двозначного типу. Кожен елемент є співвідношенням логічної правдоподібності для прийнятого біта (більш імовірно, це 0, якщо співвідношення логічної правдоподібності є позитивним). Перші k елементи відповідають інформаційній частині кодового слово.

І вхід, і вихід є сигналами дискретного часу. Співвідношення часу вихідного зразка до часу вхідного зразка становить n / k, якщо декодується тільки інформаційна частина, і 1, якщо проводиться декодування коду всього слова.

Параметри:

- матриця перевірки на парність. Цей параметр приймає розріджену матрицю з розмірністю n -k на n (де n> k> 0) дійсних чисел. Усі ненульові елементи повинні бути рівними 1. Верхня межа для значення n становить 231-1;

- формат виводу. Вихід є вектор-стовпець сигналу. Параметри є інформаційною частиною та повним кодовим словом. Коли ви вводите цей параметр в Інформаційну частину, вивід містить k елементів. Якщо ви встановите цей параметр у всьому кодовому слову, вихід містить n елементів;

- тип рішення жорстке рішення та м'яке рішення. Якщо ви встановите параметр тверде рішення, виходом буде декоровано біт (типу double або boolean). Якщо ви встановите параметр м'яке» рішення, виходом буде коефіцієнт логічної правдоподібності (типу double);

- кількість ітерацій. Це може бути будь-яке натуральне число.

За допомогою описаних блоків були побудовані дві схеми для завадостікого кодування (рис.3.20) та (рис.3.21) [10].

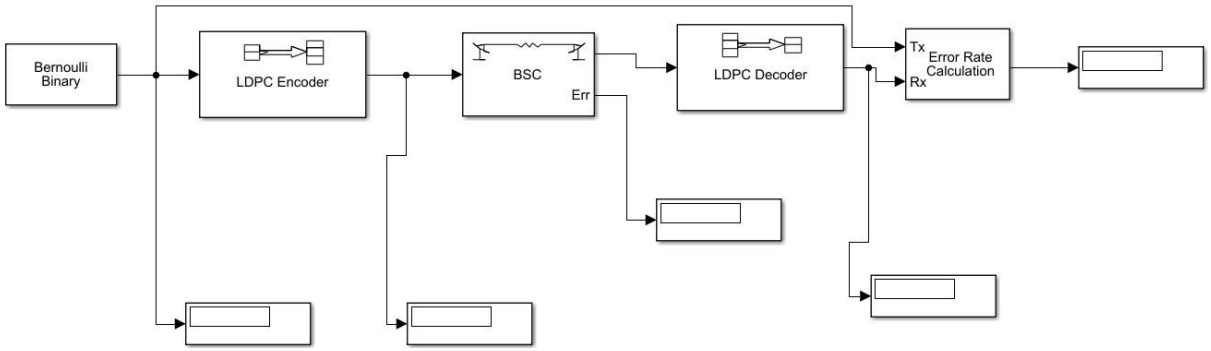

Рисунок 3.20 – Схема для коду LDPC з ДСК

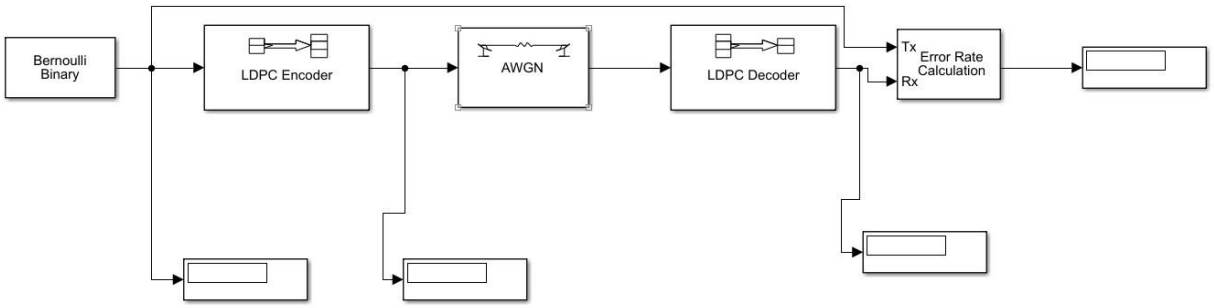

Рисунок 3.21 – Схема для коду LDPC з АБГШ

3.2 Результати експерименту

Завдяки побудованим схемам були проведені розрахунки для різних видів алгоритмів з різною довжиною кодового слова.

Першим було проведено експеримент з кодом Хеммінга. Для цього в блоках кодера та декодера Хеммінга було задано довжину кодового слова, а в двійковому симетричному каналі задано помилку та її ймовірність. У ході експерименту довжина кодового слова та ймовірність помилки змінювались. Результати експерименту показані на рисунках 3.22 – 3.24.

На рис. 3.22 приведено графік для кодів Хеммінга (7,4) та (15,11). На ньому бачимо, що при зміні ймовірності помилок їх кількість знаходиться не лінійно. Не всі помилки знайдені та виправлені. На рис. 3.23 графік кодів Хеммінга (63,57) та (31,26). На ньому бачимо, що на відміну від попереднього графіку тут кількість знайдених помилок більш лінійна. На

рис. 3.24 графік кода Хеммінга (255, 247), на ньому становиться зрозуміло, що при збільшенні ймовірності помилки кількість знайдених помилок теж стає більшою. Також бачимо, що при збільшенні довжини кодового слова кількість виправлених помилок набагато більше.

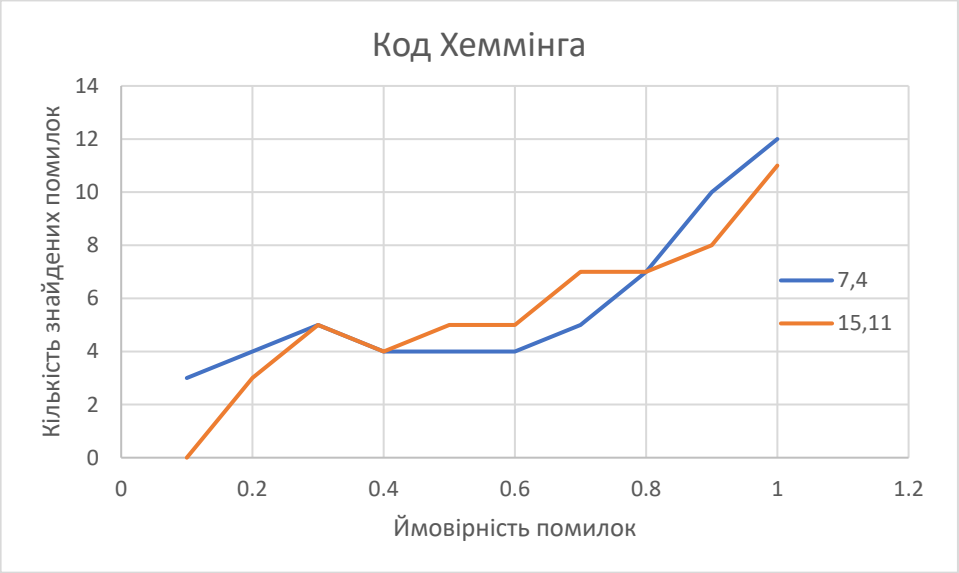

Рисунок 3.22 – Код Хеммінга

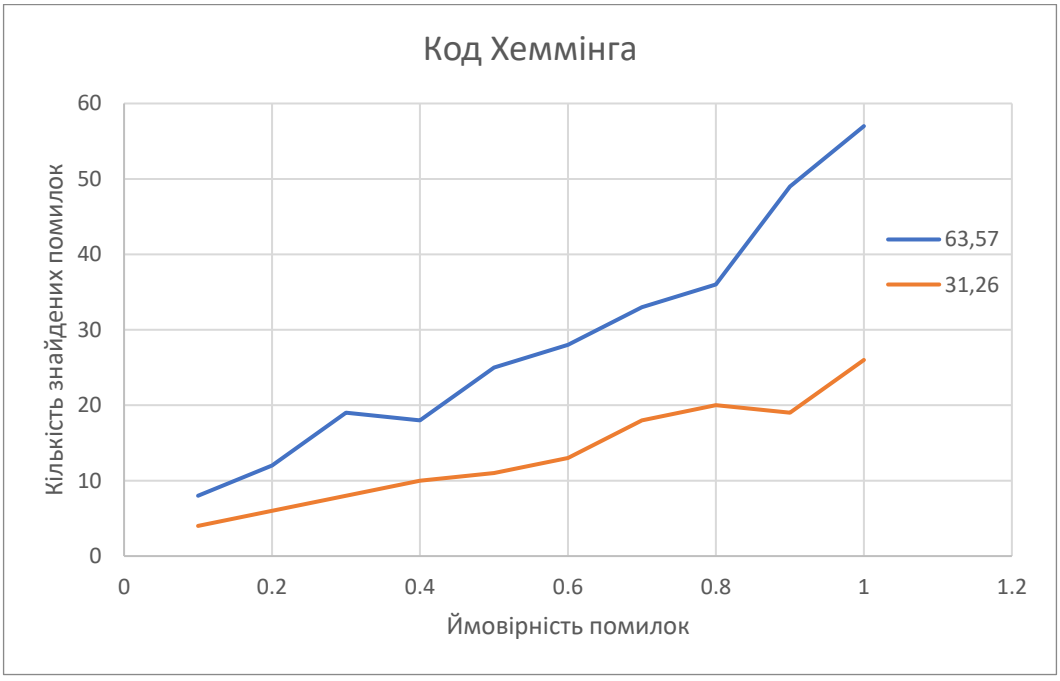

Рисунок 3.23 – Код Хеммінга

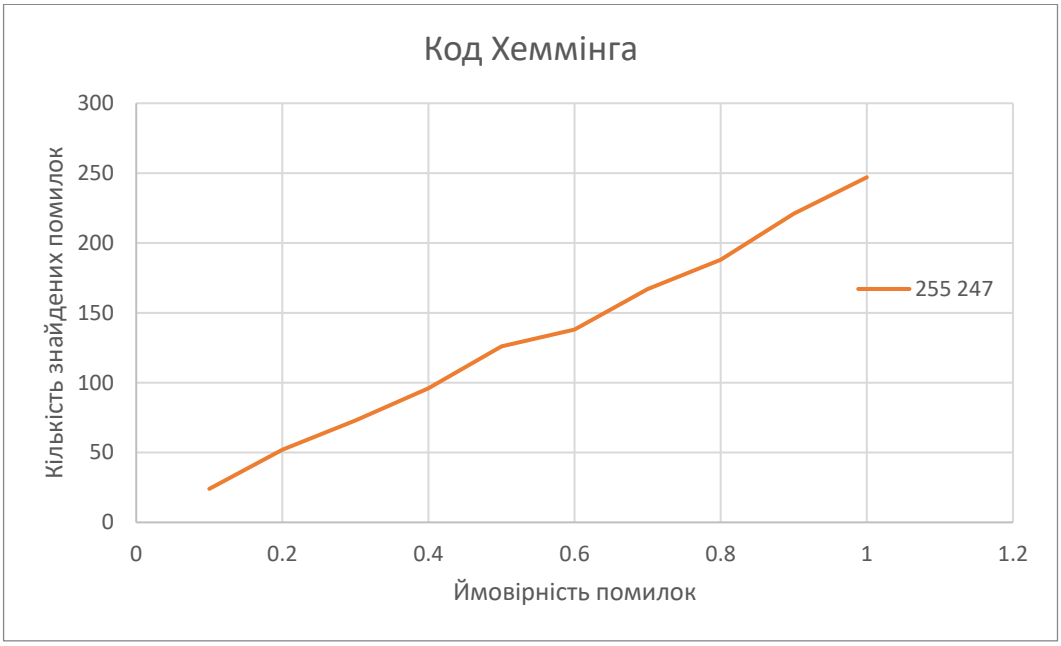

Рисунок 3.24 – Код Хеммінга

Далі було проведено експеримент з кодом БЧХ. Для цього в блоках кодера та декодера БЧХ було задано довжину кодового слова, а в двійковому симетричному каналі задано помилку та її ймовірність. У ході експерименту довжина кодового слова та ймовірність помилки змінювались. Результати експерименту показані на рисунках 3.25 – 3.27.

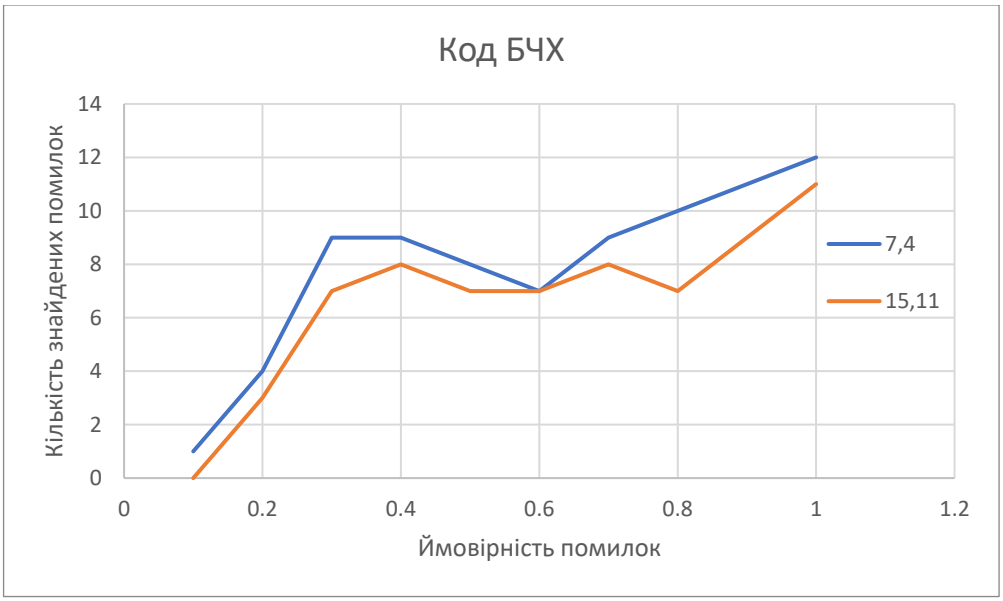

Рисунок 3.25 – Код БЧХ

На рис. 3.25 приведено графік для кодів БЧХ (7,4) та (15,11).На ньому бачимо, що на відміну від кодів Хеммінга при зміні ймовірності помилок їх кількість знаходиться більш лінійно. Використовуючи цей код можна знайти більшу кількість помилок при невеликому кодовому слові. На рис. 3.26 графік кодів БЧХ (255,247) та (255,187). На ньому бачимо, що на відміну від попереднього графіку тут кількість знайдених помилок більш лінійна, лінії графіку мають невеликі перегини. На рис. 3.27 графік кодів БЧХ (511,502) та (511,250), на ньому становиться зрозуміло ,що при збільшенні ймовірності помилки кількість знайдених помилок теж стає більшою. Також бачимо, що при збільшенні довжини кодового слова кількість виправлених помилок набагато більше. При використанні кода БЧХ частота знаходження помилок становиться більше.

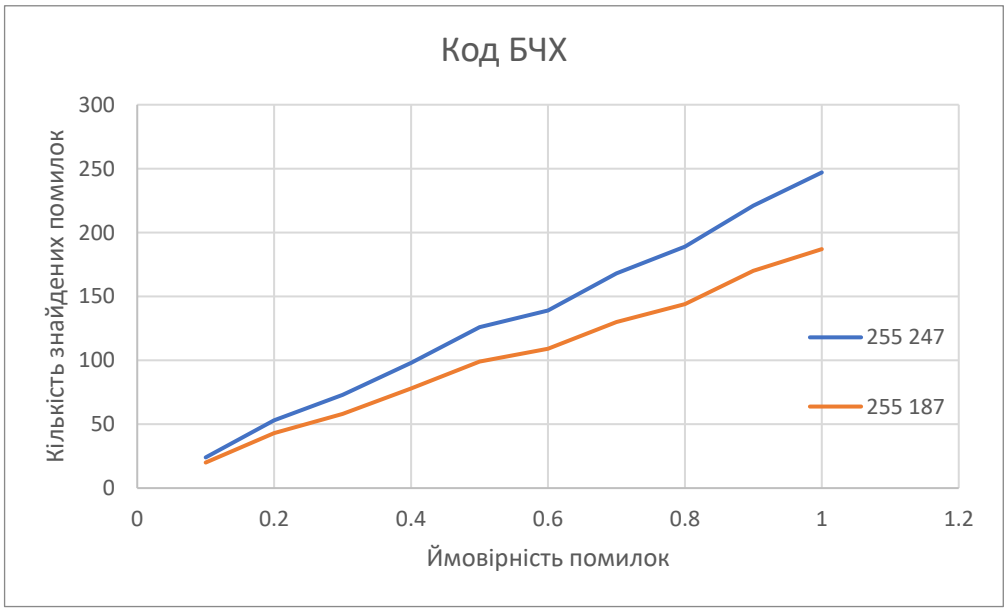

Рисунок 3.26 – Код БЧХ

Третім було проведено експеримент з кодом Хеммінга. Для цього в блоках кодера та декодера Хеммінга було задано довжину кодового слова, а в двійковому симетричному каналі задано помилку та її ймовірність. У ході

експерименту довжина кодового слова та ймовірність помилки змінювались. Результати експерименту показані на рисунках 3.28 – 3.31.

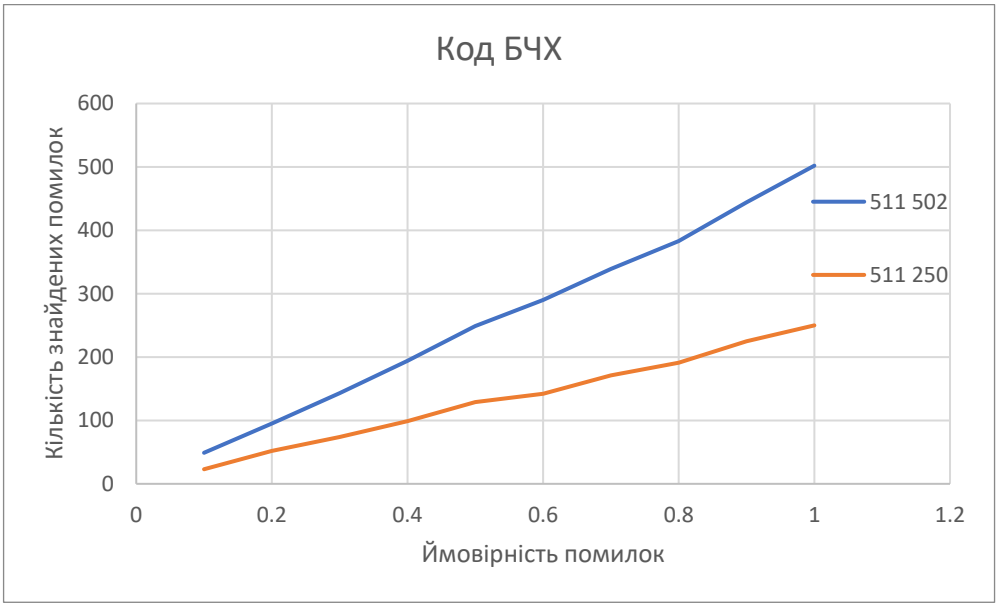

Рисунок 3.27 – Код БЧХ

Рисунок 3.28 – Код Ріда-Соломона

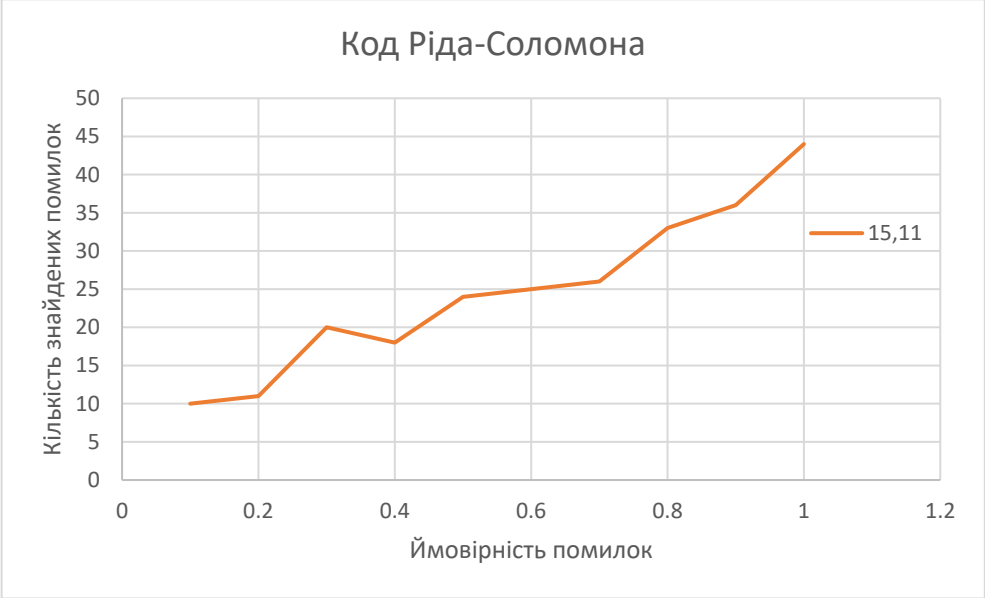

Рисунок 3.29 – Код Ріда Соломона

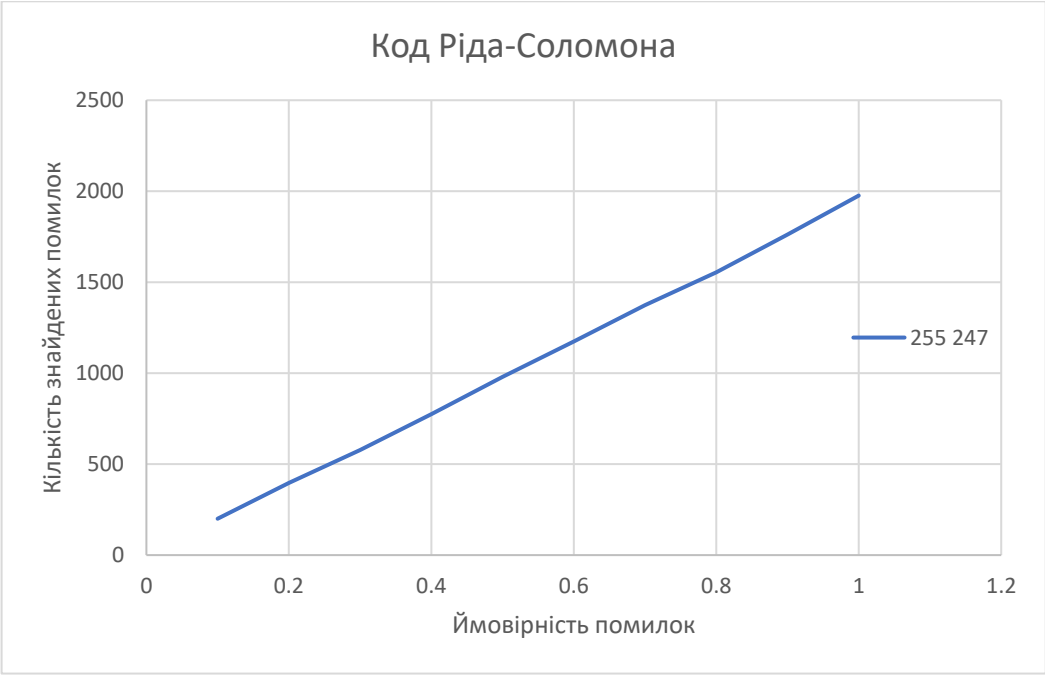

Рисунок 3.30 – Код Ріда-Соломона

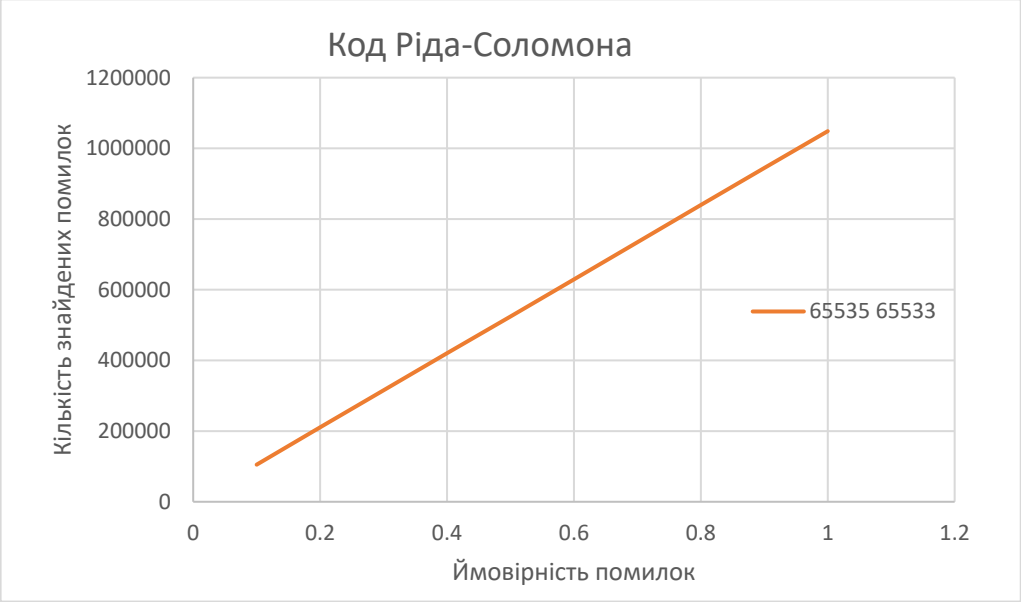

Рисунок 3.31 – Код Ріда-Соломона

На рис. 3.28 та 3.29 приведено графік для кодів Ріда-Соломона (7,4) та (15,11). На ньому бачимо, що на відміну від кодів Хеммінга та БЧХ при зміні ймовірності помилок їх кількість знаходиться більш лінійно. Використовуючи цей код можна знайти більшу кількість помилок при невеликому кодовому слові. На рис. 3.30 графік кодів Ріда-Соломона (255,247). На ньому бачимо, що на відміну від попереднього графіку тут кількість знайдених помилок більш лінійна, лінії графіку майже не мають перегини. На рис. 3.31 графік Кода Ріда-Соломона максимальної довжини, на ньому становиться зрозуміло ,що при збільшенні ймовірності помилки кількість знайдених помилок теж стає більшою. Також бачимо, що при збільшенні довжини кодового слова кількість виправлених помилок набагато більше.

Останнім було проведено експеримент з кодом LDCP. Для цього в блоках кодера та декодера LDCP було використану функцію для кодів LDCP, а в двійковому симетричному каналі задано помилку та її ймовірність. У ході експерименту довжина кодового слова та ймовірність помилки змінювались. Результати експерименту показані на рисунках 3.32 – 3.34.

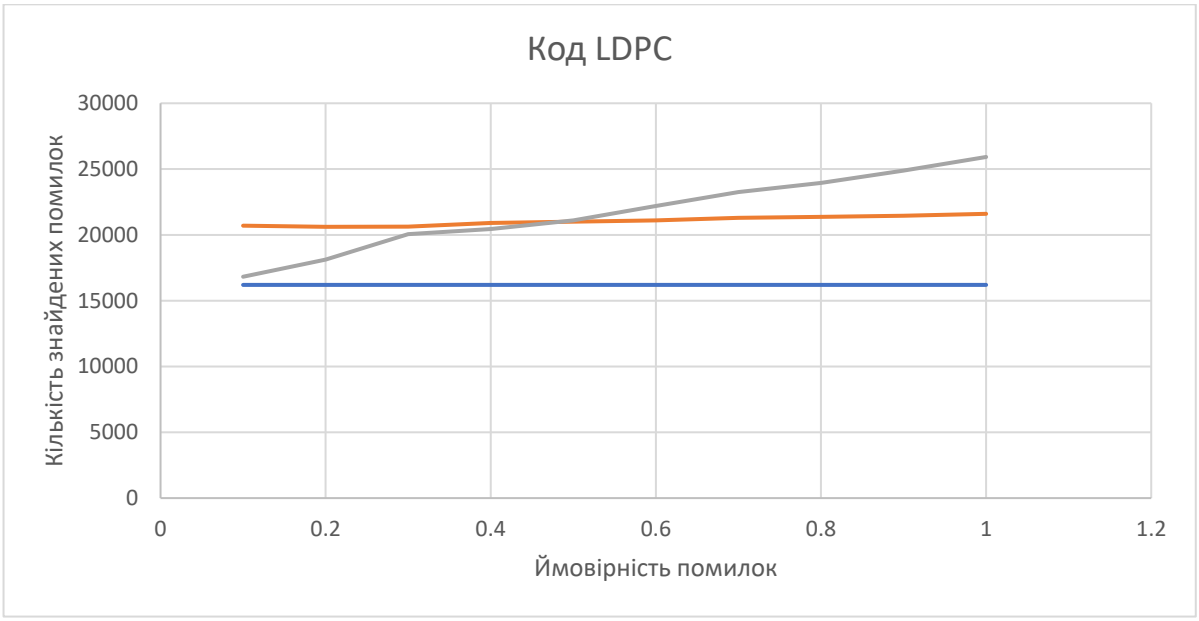

Рисунок 3.32 – Код LDPC зі швидкістю 1/4,1/3,2/5

На рисунку 3.32 зображено, що при малих швидкостях майже все помилки були знайдені. Далі на рис 3.33 та 3.34 видно, що зі збільшенням швидкості помилки знаходяться більш лінійно.

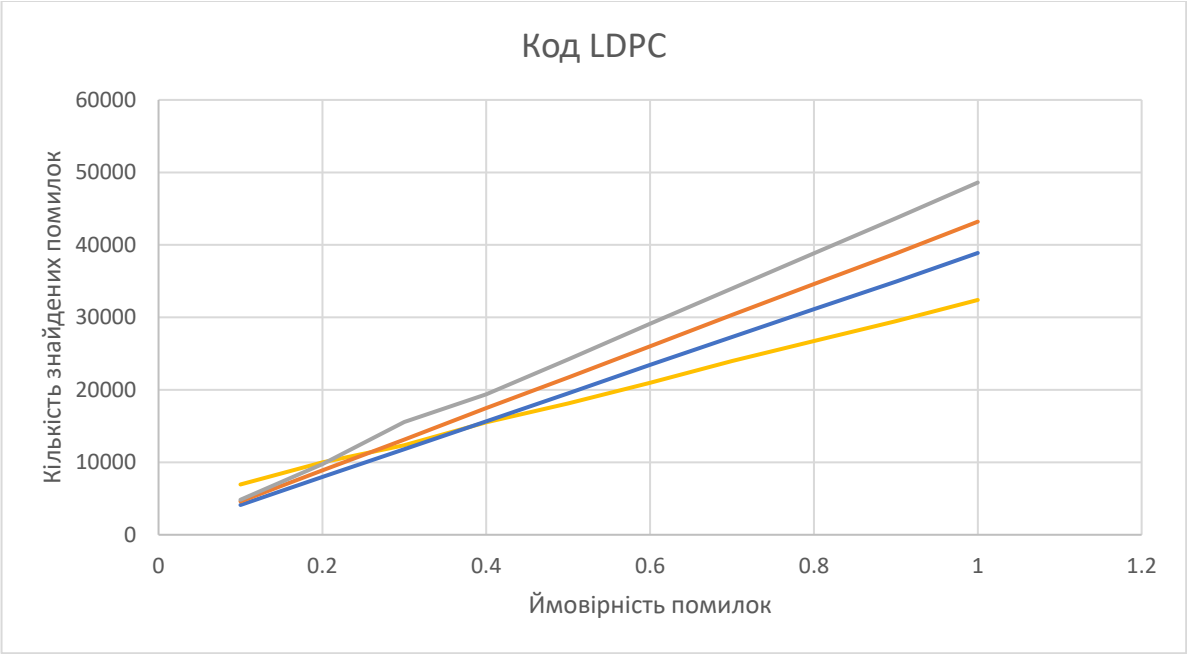

Рисунок 3.33 – Код LDPC зі швидкістю 1/2,3/5,2/3,3/4

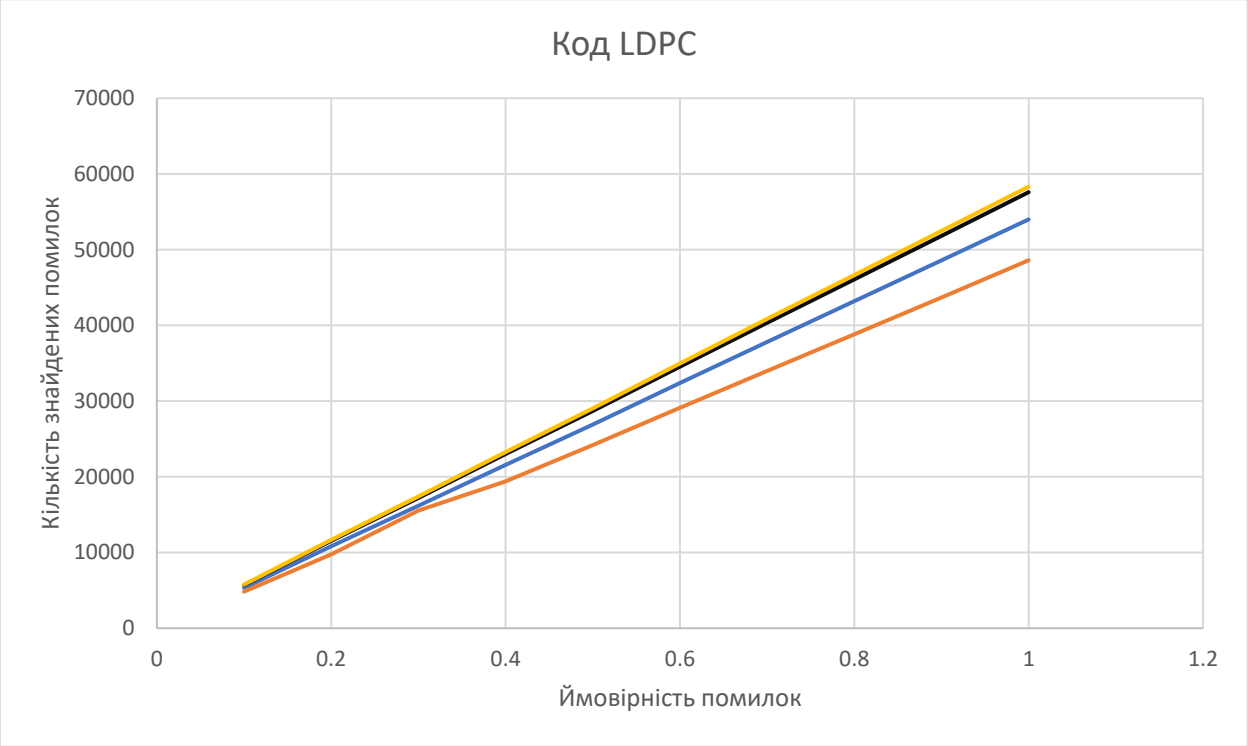

Рисунок 3.34 – Код LDPC зі швидкістю 4/5,5/6,8/9,9/10

3.3 Висновки за розділом 3

В підсумку на прикладі кода Хеммінга з різною розмірністю було вставлено, що при збільшенні розмірності кода ріст кількості помилок з ростом ймовірність їх появи становиться більш плавним. Також це дозволяє досягти меншої кількості помилок при великій вірогідності їх появи. На наступному етапі порівняли між собою різні коди. Було виявлено, що найбільш завадостійкими є коди Ріда-Соломона та LDPC. Навіть при великій послідовності їх частота помилок нижче ніж у інших. Також ці коди показують найбільш плавний ріст кількості помилок при збільшенні їх появи. Таким чином, якщо обирати серед усіх кодів, та переходи до реалізації на практиці, то реалізацію завадостійкого кодування слід виконувати одним з цих методів.

ВИСНОВКИ

Результатом дипломної роботи є дослідження та порівняння декількох методів та алгоритмів завадостійкого кодування та з’ясування для яких цілей їх краще використовувати.

Піл час роботи було описано та проаналізовано, що таке завадостійке кодування, які існують його види, було розглянуте ,що таке твердотільні накопичувачі, а також алгоритми завадостійкого кодування що в них застосовуються. Також проаналізовані основні види ПЗ, яке дозволяє реалізувати завадостійке кодування, описано декілька систем комп’ютерної математики, також було розглянуто схему імітаційного моделювання та її складові частини.

Проведено експеримент з ефективності завадостійкого кодування, розглянуто блоки, що використовуються для реалізації завадостійкого кодування у системі Simulink. У ході роботи було виявлено, що найбільш завадостійкими є коди Ріда-Соломона та LDPC. Навіть при великій послідовності їх частота помилок нижче ніж у інших. Також ці коди показують найбільш плавний ріст кількості помилок при збільшенні їх появи. Таким чином, якщо обирати серед усіх кодів , та переходи до реалізації на практиці , то реалізацію завадостійкого кодування слід виконувати одним з цих методів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Майданюк В. П. Кодування та захист інформації. - Вінниця: ВНТУ, 2009. - 164 с.

2. Каплун В. А., Майданюк В. П. Захист операційних систем. - Вінниця: ВНТУ, 2007. – 185 с.

3. Білецький А.Я. Алгебраїчні основи теорії кодування і криптографії. /А.Я. Білецький. − К.: Agrar Media Group, 2017. – 176 с.

4. Івашко А.В., Крилова В.А. Теорія інформації та кодування в прикладах і задачах. – Харків : НТУ «ХПІ», 2022. – 317 с.

5. Жураковський Ю. П. Теорія інформації та кодування. / Ю. П. Жураковський, В. П. Полторак. – К. Вища школа, 2001.– 255 с.

6. Кузьмін І. В. Основи теорії інформації та кодування / І. В. Кузьмін, І. В. Троцишин, А. І. Кузьмін, В. О. Кедрус, В. Р. Любчик. – Хмельницький : ХНУ, 2009. – 373 с.

7. Сорока Л. С. Основи теорії інформації / Л.С. Сорока.– Харків: ХНУ ім. В.Н.Каразіна, 2007. – 264 с.

8. Іващенко П.В. Основи теорії інформації: навч. посіб. / П.В. Іващенко –Одеса: ОНАЗ ім. О.С. Попова, 2015. – 53 с

9. Беркман Л.Н., Бондарчук А.П., Гайдур Г.І., Чумак Н.С. Кодування джерел інформації та каналів зв’язку. – Київ: ННІТІ ДУТ, 2018. – 91с.

10. Майданюк В. П. Методи і засоби комп’ютерних інформаційних технологій. Кодування зображень. – Вінниця: ВДТУ, 2001. – 63 с.

11. Рябенький В.М., Жуйков В.Я., Гулий В.Д. Цифрова схемотехніка: Навч. посібник. – Львів: Новий світ- 2000, 2009, - 736 с.

12. Коваленко А.Є. Теорія інформації та кодування: Практикум для студентів напряму підготовки 6.040303 «Системний аналіз». Київ: НТУУ«КПІ», 2014. – 198 с.

13. Бабак В.П. Обробка сигналів: Підручник / В.П. Бабак, В.С. Хандецький, Е. Шрюфер. – К.:Либідь. – 1996. – 392 с.

14. Тулякова Н.О. Теорія інформації: Навчальний посібник. Суми: СумДУ, 2008. 212 с. 2010. – 248с.

15. Mauro Barni, Benedetta Tondi Lecture notes on Information Theory and Coding. Siena: Universit` a degli Studi di Siena Facolt`a di Ingegneria, 2012. – 156 p.

16. В.Н. Олейников, О.В. Зубков, В.М. Карташов, И.В. Корытцев, С.И. Бабкин, С.А. Шейко, И.С. Селезнев. Экспериментальная оценка эффективности алгоритмов пеленгования беспилотных летательных аппаратов по акустическому излучению. Радиотехника: Всеукр. межвед. науч.-техн. сб. – 2019. – Вып. 199. – С. 29 – 37.

17. V. Kartashov, V. Oleynikov, I. Koryttsev, S. Sheiko, O. Zubkov, S. Babkin, I. Selieznov. Use of Acoustic Signature for Detection, Recognition and Direction Finding of Small Unmanned Aerial Vehicles. 2020 IEEE 15th International Confer-ence on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET). 2020. 4 p.

18. Kartashov V.M., Oleynikov V.N, Zubkov O.V., Korytsev I.V., Babkin S. I., Sheiko S.A., Kolendovskaya M.M. Spatial-temporal Processing of acoustic Signals of Unmanned Aerial Vehicles/ Telecommunications and Radio Engineering. – New York. – 2020. – Vol. 79, №9. – P.769-780.

19. V. Kartashov, V. Oleynikov , I. Koryttsev, S. Sheiko, O. Zubkov, S. Babkin. Processing of Wide Band Acoustic Signals During Detection of Unmanned Aerial Vehicles // 2020 IEEE Ukrainian Microwave Week (UkrMW). Kharkiv, Ukraine, September 21 - 25, 2020. Volume 1 on 2020 IEEE 12th International Conference on Antenna Theory and Techniques (ICATT). pp. 35-39.

20. V.M. Kartashov, G.I. Sidorov, S.A. Sheiko, M.M. Kolendovskaya, O.Yu. Sergienko. Principles of construction and assessment of technical characteristics of multi-frequency atmospheric sodar in the humidity measurement mode. Telecommunications and Radio Engineering. Vol. 79. N.4. 2020. – pp. 323-333.

21. S. Sheiko. Study of the method for assessing atmospheric turbulence by the envelope of sodar signals // Eastern-European Journal of Enterprise Technologies. – 2/5 (92). – April, 2018. – p. 33–40.

22. Сідоров Г.І., Шейко С.О., Шаповалов С.В., Полонська А.С., Дмитренко А.І. Акустичний метод вимірювання турбулентного стану атмосферного прикордонного шару // Радиотехніка: Всеукр. міжвід. наук.-техн. зб. 2018. – Вип. 192. – С. 46–50.

23. Valerii V. Semenets, V. M. Kartashov, V. I. Leonidov. Registration of refraction phenomenon in the problem of acoustic sounding of atmosphere in airports zone. Telecommunications and Radio Engineering. Volume 77, Issue 5, 2018. – P. 461-468.

24. Буйницкий Д.В. Аппаратно-программный комплекс для исследования акустических устройств // 23-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». – Харків, 16–18 квітня 2019 р. – с. 92-93.

25. Ашихмин В.О. Исследование системы коррекции звука с учётом характеристик помещения // 23-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». – Харків, 16–18 квітня 2019 р. – с. 96-97.

26. Тушев В.О. Исследование влияния фазовых искажений аудиоаппаратуры на качество звучания // 23-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». – Харків, 16–18 квітня 2019 р. – с. 84-85.