МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Крымский федеральный университет имени В.И. Вернадского»

Физико-технический институт

Кафедра компьютерной инженерии и моделирования

Лабораторная работа № 4
«Систематический код»
по дисциплине
«Теория информации и кодирование»

Выполнил:
студент 3 курса
группа ИВТ-222
Гоголев В. Г.

Проверил:
Филиппов Д.М.
«____» _____20___г.
Подпись: _____

Симферополь, 2024

Цель: построить помехоустойчивый систематический код, позволяющий обнаруживать и исправлять все однократные ошибки.

Техническое задание: источник информации вырабатывает сообщения, содержащие k информационных разрядов.

Значения разрядов генерируются в двоичной системе счисления счетчиком случайных чисел. Необходимо: 1. разработать программное обеспечение для передатчика, которое будет строить систематический код с заданной исправляющей способностью; 2. разработать программное обеспечение на приемной стороне, позволяющее корректировать принятую ошибочную кодовую комбинацию; 3. провести комплекс численных экспериментов, в ходе которых на передающей стороне построить систематический код с заданной исправляющей способностью, сгенерировать ошибочный систематический код, на приемной стороне вычислить позицию ошибки и скорректировать принятую кодовую комбинацию.

Ход работы:

Вариант № 4

Задание I. С использованием разработанного программного обеспечения для передатчика необходимо сгенерировать производящую матрицу систематического кода Pn,k и построить проверочную матрицу H, при помощи которой можно обнаруживать и исправлять все однократные ошибки.

Задание II. Провести цикл комплексных экспериментов (не менее 6), в ходе которого необходимо:

а) сгенерировать случайным образом информационную кодовую комбинацию, состоящую из k разрядов, на передающей стороне;

б) построить для информационной кодовой комбинации на передающей стороне систематический код, позволяющий обнаруживать и исправлять все однократные ошибки;

в) передать проверочную матрицу с выходы программного обеспечения на передающей стороне на вход программного обеспечения приемной стороны;

г) передать систематический код от передатчика к приемнику, сгенерировав случайным образом однократную ошибку в любом разряде систематического кода;

д) при помощи программного обеспечения на приемной стороне определить синдром ошибки принятого систематического кода;

е) с использованием проверочной матрицы на приемной стороне по синдрому ошибки определить позицию ошибки и откорректировать систематический код.

```python
def create_verification_matrix(P, K, d_min):
    verification_matrix = np.zeros((K + P, P), dtype=int)
    decrescent = P + K
    for i in range(K):
        matrix_line = decrescent
        one_counter = 0
        for j in range(P - 1, -1, -1):
            verification_matrix[i, j] = matrix_line % 2
            one_counter += matrix_line % 2
            matrix_line //= 2
        if one_counter + 1 < d_min:
            i -= 1
        decrescent -= 1
    for i in range(K, K + P):
        for j in range(P - 1, -1, -1):
            verification_matrix[i, j] = 1 if i - K == j else 0
    return verification_matrix
```

Рисунок 1 – функция creatre_verification_matrix

Назначение: создает проверочную матрицу для систематического кода.

Инициализация матрицы: создает нулевую матрицу размером (K + P, P), где K — количество информационных битов, P — количество проверочных битов.

Заполнение строки матрицы:

- Переменная decrescent инициализируется как P + K.
- В цикле строки матрицы заполняются двоичными представлениями чисел в убывающем порядке.
- Подсчитывается количество единиц в строке (one_counter). Если их недостаточно, строка пересчитывается.
- Добавление единичной матрицы: последние строки матрицы заполняются единицами по диагонали, образуя единичную матрицу.

Возврат результата: возвращает готовую проверочную матрицу.

```
def create_systemic_code(P, d_min, info_array):
    K = len(info_array)
    systemic_code = np.zeros(P + K, dtype=int)
    systemic_code[:K] = info_array
    verification_matrix = create_verification_matrix(P, K, d_min)
    for i in range(P):
        systemic_code[K + i] = np.sum(info_array * verification_matrix[:K, i]) % 2
    return systemic_code, verification_matrix
```

Рисунок 2 – функция create_systematic_code

Назначение: создаёт систематический код на основе входного информационного массива.

Как работает:

- инициализация системного кода: создаёт нулевой массив длиной P + K и копирует информационный массив в начало этого массива;
- вычисление проверочных битов: использует проверочную матрицу для вычисления проверочных битов, добавляя их к системному коду;
- возврат результата: возвращает систематический код и проверочную матрицу.

```python
def search_error(P, d_min, systemic_code):
    verification_matrix = create_verification_matrix(P, len(systemic_code) - P, d_min)
    syndrom = np.array([(np.sum(systemic_code * verification_matrix[:, i]) % 2) for i in range(P)])
    for i in range(len(systemic_code)):
        if np.array_equal(syndrom, verification_matrix[i]):
            return i + 1, i+2
    return -1, i
```

Рисунок 3 – функция search_error

Назначение: обнаруживает позицию ошибки в систематическом коде и вычисляет синдром ошибки.

Как работает:

- создание проверочной матрицы: создаёт проверочную матрицу для систематического кода;
- вычисление синдрома ошибки: вычисляет синдром ошибки путем умножения систематического кода на проверочную матрицу и взятия остатка от деления на 2;
- поиск ошибки: сравнивает синдром с каждой строкой матрицы. Если они совпадают, возвращает позицию ошибки и синдром. Если нет, возвращает -1 и синдром;
- возврат результата: возвращает позицию ошибки и синдром.

```
Эксперимент 5
Входное сообщение: 1010101001111
Систематический код: 1010101001111100000
Производящая матрица (Pn,k):
 [[1 0 0 1 0]
 [1 0 0 0 1]
 [1 0 0 0 0]
 [0 1 1 1 1]
 [0 1 1 1 0]
 [0 1 1 0 1]
 [0 1 1 0 0]
 [0 1 0 1 1]
 [0 1 0 1 0]
 [0 1 0 0 1]
 [0 1 0 0 0]
 [0 0 1 1 1]
 [0 0 1 1 0]
 [1 0 0 0 0]
 [0 1 0 0 0]
 [0 0 1 0 0]
 [0 0 0 1 0]
 [0 0 0 0 1]]
Систематический код с ошибкой: 1010101001111100100
Позиция ошибки: 16
Синдром ошибки: 17
Исправленный систематический код: 1010101001111100000
Исходное сообщение: 1010101001111
Совпадает ли с входным: True
```

Рисунок 4 – результат работы программы

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были получены навыки по формированию Систематического кода, по проверочной матрицы H и производящей матрицы H. Изучены принципы работы проверочных столбцов, алгоритм работы Систематического кода, термины позиции и синдрома ошибки

ПРИЛОЖЕНИЕ

```
import numpy as np
import random

def create_verification_matrix(P, K, d_min):
    verification_matrix = np.zeros((K + P, P), dtype=int)
    decrescent = P + K
    for i in range(K):
        matrix_line = decrescent
        one_counter = 0
        for j in range(P - 1, -1, -1):
            verification_matrix[i, j] = matrix_line % 2
            one_counter += matrix_line % 2
            matrix_line //= 2
        if one_counter + 1 < d_min:
            i -= 1
        decrescent -= 1
    for i in range(K, K + P):
        for j in range(P - 1, -1, -1):
            verification_matrix[i, j] = 1 if i - K == j else 0
    return verification_matrix

def create_systemic_code(P, d_min, info_array):
    K = len(info_array)
    systemic_code = np.zeros(P + K, dtype=int)
    systemic_code[:K] = info_array
    verification_matrix = create_verification_matrix(P, K, d_min)
    for i in range(P):
        systemic_code[K + i] = np.sum(info_array * verification_matrix[:K, i]) % 2
    return systemic_code, verification_matrix

def search_error(P, d_min, systemic_code):
    verification_matrix = create_verification_matrix(P, len(systemic_code) - P,
d_min)
    syndrom = np.array([(np.sum(systemic_code * verification_matrix[:, i]) % 2)
for i in range(P)])
    for i in range(len(systemic_code)):
        if np.array_equal(syndrom, verification_matrix[i]):
            return i + 1, i+2
    return -1, i

def main():
    num_experiments = 6

    for experiment in range(1, num_experiments + 1):
        # Генерация случайного входного сообщения
        K = 13
        input_str = ''.join(random.choice('01') for _ in range(K))
        info_array = np.array([int(bit) for bit in input_str])
```

```
        print(f"Эксперимент {experiment}")
        print("Входное сообщение:", input_str)

        # Создание систематического кода
        N = K
        d_min = 2 * 1 + 1
        while 2**K > (2**N / (N + 1)):
            N += 1
        systemic_code, verification_matrix = create_systemic_code(N - K, d_min,
info_array)
        print("Систематический код:", ''.join(map(str, systemic_code)))
        print("Производящая матрица (Pn,k):\n", verification_matrix)

        # Внесение ошибки в случайный бит систематического кода
        error_index = random.randint(0, N - 1)
        systemic_code_with_error = systemic_code.copy()
        systemic_code_with_error[error_index] = 1 -
systemic_code_with_error[error_index]
        print("Систематический код с ошибкой:", ''.join(map(str,
systemic_code_with_error)))

        # Обнаружение и исправление ошибки
        error_position, error_syndrom = search_error(N - K, d_min,
systemic_code_with_error)
        print("Позиция ошибки:", error_position)
        print("Синдром ошибки:", error_syndrom)
        systemic_code_with_error[error_position - 1] = 1 -
systemic_code_with_error[error_position - 1]
        print("Исправленный систематический код:", ''.join(map(str,
systemic_code_with_error)))

        # Вывод исходного сообщения
        output_str = ''.join(map(str, systemic_code_with_error[:K]))
        print("Исходное сообщение:", output_str)
        print("Совпадает ли с входным:", input_str == output_str)
        print("\n" + "="*50 + "\n")

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

Приложение 1 – листинг программного кода