RoboLab

# Assignments

01 - Hamming Codes

**Florian Lubitz**
Fakultät Informatik
5106912
florian.lubitz@mailbox.tu-dresden.de

## Task 1

To create a generator matrix $G_{4,8}$ for the extended Hamming Code, we have to create a generator matrix $G'_{4,7}$ for the Hamming Code first. The generator matrix $G'_{4,7}$ for the Hamming Code can be constructed by following the following rules:

- no two columns are equal, only distinct columns
- no column may be all zeros
- the number of ones in each column is odd
- each column can be read as binary numbers where top is the most significant bit
- the number represented by the columns should be as small as possible
- the columns are ordered by their binary number descending from left to right (hint: start from the right and work your way to the left)

The constructed generator matrix $G'_{4,7}$ and the corresponding parity-check matrix $H'_{3,7}$ without extension look like this:

$$
\begin{array}{c}
\begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \end{array} \\
G'_{4,7} := \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}
\end{array}
$$

$$
\Rightarrow A^T = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \rightarrow A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}
$$

$$
\begin{array}{c}
\begin{array}{ccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 \end{array} \\
\Rightarrow H'_{3,7} := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}
\end{array}
$$

Adding the additional parity bit $p_4$ for the extended Hamming Code for the generator matrix $G_{4,8}$ looks like this:

$$G'_{4,8} := \begin{array}{cccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 & p_4 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

And the corresponding parity-check matrix $H_{4,8}$ looks like this:

$$\Rightarrow H'_{4,8} := \begin{array}{cccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 & p_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

## Task 2

$$G'_{4,8} := \begin{array}{cccccccc} p_1 & p_2 & d_1 & p_3 & d_2 & d_3 & d_4 & p_4 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

To convert this non-systematic generator matrix into a systematic one, we have to apply the following steps:

Step 1:

$$\begin{array}{r} \\ r_2 + r_1 \\ \\ r_4 + r_1 \end{array} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Step 2:

$$\begin{array}{r} r_1 + r_2 \\ \\ r_3 + r_2 \\ \\ \end{array} \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Step 3:

$$\begin{array}{r} \\ r_2 + r_3 \\ \\ r_4 + r_3 \end{array} \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Step 4:

$$\begin{array}{r} r_1 + r_4 \\ r_2 + r_4 \\ \\ \\ \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\Rightarrow G_{4,8} \begin{array}{c} \begin{array}{cccccccc} d_1 & d_2 & d_3 & d_4 & p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

As G is defined as $G := (I_k | -A^T)$, we can read the parity-check matrix $H$ directly from the matrix $A$: $H := (-A | I_{n-k})$ and add the parity bit row to the end of the matrix:

$$H_{4,8} := \begin{array}{c} \begin{array}{cccccccc} d_1 & d_2 & d_3 & d_4 & p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

## Task 3

$$\vec{x_1} = (\vec{a_1} \cdot G') \mod 2 = (0100) \cdot G' \mod 2 = (10011001) \mod 2 = (10011001)\, p_4 = 1$$
$$\vec{x_2} = (\vec{a_2} \cdot G') \mod 2 = (1001) \cdot G' \mod 2 == (22110011) \mod 2 = (00110011)\, p_4 = 1$$
$$\vec{x_3} = (\vec{a_3} \cdot G') \mod 2 = (0011) \cdot G' \mod 2 == (10000111)\, p_4 = 1$$
$$\vec{x_4} = (\vec{a_4} \cdot G') \mod 2 = (1101) \cdot G' \mod 2 == (10101010)\, p_4 = 0$$

## Task 4

$$\vec{x_1} = (11001101)$$
$$\vec{x_2} = (10011001)$$
$$\vec{x_3} = (11011011)$$
$$\vec{x_4} = (11010101)$$

### Parity bit $p_4$

$$\vec{x_1} = (11001101) \rightarrow p_4 = 1 \rightarrow \text{error}$$
$$\vec{x_2} = (10011001) \rightarrow p_4 = 1 \rightarrow \text{correct}$$
$$\vec{x_3} = (11011011) \rightarrow p_4 = 1 \rightarrow \text{correct}$$
$$\vec{x_4} = (11010101) \rightarrow p_4 = 1 \rightarrow \text{error}$$

### Syndrome vector

The syndrome vector $\vec{z}$ is calculated by multiplying the received vector $\vec{x}$ with the parity-check matrix $H$, where the last bit is the parity bit $p_4$ and the first three bits are the syndrome vector $\vec{z}$:

$$\vec{z} = (\vec{x} \cdot H) \mod 2$$

This results in the following syndrome vectors:

$$\begin{aligned}
\vec{z_1} &= (\vec{x_1} \cdot H) \mod 2 = ((11001101) \cdot H) \mod 2 \\
&= \begin{pmatrix} 2 & 2 & 2 & 5 \end{pmatrix} \mod 2 = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \\
&\Rightarrow \text{error in parity bit} \\
\vec{z_2} &= (\vec{x_2} \cdot H) \mod 2 = ((10011001) \cdot H) \mod 2 \\
&= \begin{pmatrix} 2 & 2 & 2 & 4 \end{pmatrix} \mod 2 = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} \\
&\Rightarrow \text{no error} \\
\vec{z_3} &= (\vec{x_3} \cdot H) \mod 2 = ((11011011) \cdot H) \mod 2 \\
&= \begin{pmatrix} 3 & 2 & 4 & 6 \end{pmatrix} \mod 2 = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \\
&\Rightarrow \text{multiple errors} \\
\vec{z_4} &= (\vec{x_4} \cdot H) \mod 2 = ((11010101) \cdot H) \mod 2 \\
&= \begin{pmatrix} 2 & 3 & 3 & 5 \end{pmatrix} \mod 2 = \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix} \\
&\Rightarrow \text{error, try correction}
\end{aligned}$$

### Error correction

For error correction, we have to find the syndrome $\vec{z_4}$ as a column in $H$ and flip the bit in the received vector $\vec{x_4}$ at the matching column:

The first column of $H$ is $\begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}^T$, and matches the syndrome $\vec{z_4} = \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}$ so we have to flip the bit at the first position in $\vec{x_4}$:

$$\vec{x_4} = (11010101) \rightarrow \text{flip bit at position 0} \rightarrow \vec{x'_4} = (01010101)$$

To validate the correction, we calculate the syndrome vector again:

$$\begin{aligned}
\vec{z'_4} &= (\vec{x'_4} \cdot H) \mod 2 = ((01010101) \cdot H) \mod 2 \\
&= \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \text{no error}
\end{aligned}$$

# Appendix

## Python code for encoding and decoding

```python
# %%
from typing import Tuple, Dict
from functools import reduce

Matrix = Tuple[Tuple[int, ...], ...]


def transpose(matrix: Matrix) -> Matrix:
    """Return the transpose of a matrix."""
    return tuple(zip(*matrix))


# %%
G_non_sys: Matrix = (
    (1, 1, 1, 0, 0, 0, 0, 1),
    (1, 0, 0, 1, 1, 0, 0, 1),
    (0, 1, 0, 1, 0, 1, 0, 1),
    (1, 1, 0, 1, 0, 0, 1, 0),
)
H_non_sys: Matrix = (
    (1, 0, 1, 0, 1, 0, 1, 0),
    (0, 1, 1, 0, 0, 1, 1, 0),
    (0, 0, 0, 1, 1, 1, 1, 0),
    (1, 1, 1, 1, 1, 1, 1, 1),
)
G_sys: Matrix = (
    (1, 0, 0, 0, 0, 1, 1, 1),
    (0, 1, 0, 0, 1, 0, 1, 1),
    (0, 0, 1, 0, 1, 1, 0, 1),
    (0, 0, 0, 1, 1, 1, 1, 0),
)
H_sys: Matrix = (
    (0, 1, 1, 1, 1, 0, 0, 0),
    (1, 0, 1, 1, 0, 1, 0, 0),
    (1, 1, 0, 1, 0, 0, 1, 0),
    (1, 1, 1, 1, 1, 1, 1, 1),
)
```

```python

# %%
Vector = Tuple[int, ...]


def encode(a: Vector, G: Matrix) -> Vector:
    """Encode a message vector using a generator matrix."""
    return tuple(sum(a * b for a, b in zip(a, col)) % 2 for col in transpose(G))


# %%
words: Dict[str, Vector] = {
    "1": (0, 1, 0, 0),
    "2": (1, 0, 0, 1),
    "3": (0, 0, 1, 1),
    "4": (1, 1, 0, 1),
}

print("Task 3, encode with non-systematic matrix")
for word, vector in words.items():
    encoded = encode(vector, G_non_sys)
    print(f"{word}: {encoded}, Parity: {encoded[-1]} -> {'correct' if sum(encoded) % 2 == 0 else 'incorrect'}")

# %%


# %%
words_2: Dict[str, Vector] = {
    "1": (1, 1, 0, 0, 1, 1, 0, 1),
    "2": (1, 0, 0, 1, 1, 0, 0, 1),
    "3": (1, 1, 0, 1, 1, 0, 1, 1),
    "4": (1, 1, 0, 1, 0, 1, 0, 1),
}


# %%
def get_syndrome(x: Vector, H: Matrix) -> Vector:
    """Check if a vector has the correct parity."""
    return tuple(sum(a * b for a, b in zip(x, col)) % 2 for col in H)

```

```python

print("Task 4, check with non-systematic matrix")
for word, vector in words_2.items():
    *syndrome, overall_parity = get_syndrome(vector, H_sys)
    syndrome_number = sum(2**i * bit for i, bit in enumerate(syndrome))
    syndrome_bits = sum(syndrome)

    print(
        f"Word {word}: Syndrome with parity: {syndrome + [overall_parity]} ({syndrome_bits}), overall_parity: (p4:{vector[-1]}) -> {'correct' if overall_parity == 0 else 'incorrect'}"
    )
    if syndrome_bits == 0 and overall_parity == 0:
        print("\tno error")
    elif syndrome_bits == 0 and overall_parity == 1:
        print("\tError in p4")
    elif syndrome_bits >= 1 and overall_parity == 1:
        print("\terror, try to correct it")
        vector = list(vector)
        # Search for the column in H that matches the syndrome
        error_position_mask = reduce(
            # and all the columns, resulting in a tuple of 1s and 0s where 1s indicate the error positions
            lambda total, new: tuple(a & b for a, b in zip(total, new)),
            # for each row in H, return the row if the syndrome bit is 1, otherwise return the row with all bits flipped
            # when anding the rows, the result will be a tuple of 1s and 0s where 1s indicate the positions the syndrome matches the column
            (
                mask if s == 1 else tuple(val ^ 1 for val in mask)
                for s, mask in zip(syndrome + [overall_parity], H_sys)
            ),
        )
        # get the positions of the 1s in the mask
        error_positions = tuple(i for i, v in enumerate(error_position_mask) if v == 1)
        # if there are no error positions or more than 1, the error is uncorrectable
        if len(error_positions) == 0:
            print("\tsyndrome does not match any column in H, multiple errors")
        elif len(error_positions) != 1:
```

```python
            print("\tsyndrome matches multiple columns in H, multiple errors")
        # if there is exactly one error position, flip the bit at that position
        else:
            print(f"\t error position: {error_positions[0]}")
            vector[error_positions[0]] = vector[error_positions[0]] ^ 1
            print(f"\tcorrected vector: {vector}")
            print(f"\tcorrected syndrome: {get_syndrome(vector, H_sys)}")
    else:
        print("\tMultiple errors")
    print()
```