



**TECHNISCHE
UNIVERSITÄT
DRESDEN**

RoboLab

Assignments

01 - Hamming Codes

Florian Lubitz

Fakultät Informatik

5106912

florian.lubitz@mailbox.tu-dresden.de

Task 1

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

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

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

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

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

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

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

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

Task 2

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

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

Step 1:

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

Step 2:

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

Step 3:

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

Step 4:

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

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

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:

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

Task 3

$$\begin{aligned} \vec{x}_1 &= (\vec{a}_1 \cdot G') \bmod 2 = (0100) \cdot G' \bmod 2 = (10011001) \bmod 2 = (10011001) p_4 = 1 \\ \vec{x}_2 &= (\vec{a}_2 \cdot G') \bmod 2 = (1001) \cdot G' \bmod 2 = (22110011) \bmod 2 = (00110011) p_4 = 1 \\ \vec{x}_3 &= (\vec{a}_3 \cdot G') \bmod 2 = (0011) \cdot G' \bmod 2 = (10000111) p_4 = 1 \\ \vec{x}_4 &= (\vec{a}_4 \cdot G') \bmod 2 = (1101) \cdot G' \bmod 2 = (10101010) p_4 = 0 \end{aligned}$$

Task 4

$$\begin{aligned} \vec{x}_1 &= (11001101) \\ \vec{x}_2 &= (10011001) \\ \vec{x}_3 &= (11011011) \\ \vec{x}_4 &= (11010101) \end{aligned}$$

Parity bit p_4

$$\begin{aligned} \vec{x}_1 &= (11001101) \rightarrow p_4 = 1 \rightarrow \text{error} \\ \vec{x}_2 &= (10011001) \rightarrow p_4 = 1 \rightarrow \text{correct} \\ \vec{x}_3 &= (11011011) \rightarrow p_4 = 1 \rightarrow \text{correct} \\ \vec{x}_4 &= (11010101) \rightarrow p_4 = 1 \rightarrow \text{error} \end{aligned}$$

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:

$$\begin{aligned}\vec{z}_1 &= (\vec{x}_1 \cdot H) \mod 2 = ((11001101) \cdot H) \mod 2 \\ &= \begin{pmatrix} 2 & 2 & 2 & 5 \end{pmatrix} \mod 2 = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}\end{aligned}$$

\Rightarrow error in parity bit

$$\begin{aligned}\vec{z}_2 &= (\vec{x}_2 \cdot H) \mod 2 = ((10011001) \cdot H) \mod 2 \\ &= \begin{pmatrix} 2 & 2 & 2 & 4 \end{pmatrix} \mod 2 = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}\end{aligned}$$

\Rightarrow no error

$$\begin{aligned}\vec{z}_3 &= (\vec{x}_3 \cdot H) \mod 2 = ((11011011) \cdot H) \mod 2 \\ &= \begin{pmatrix} 3 & 2 & 4 & 6 \end{pmatrix} \mod 2 = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}\end{aligned}$$

\Rightarrow multiple errors

$$\begin{aligned}\vec{z}_4 &= (\vec{x}_4 \cdot H) \mod 2 = ((11010101) \cdot H) \mod 2 \\ &= \begin{pmatrix} 2 & 3 & 3 & 5 \end{pmatrix} \mod 2 = \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}\end{aligned}$$

\Rightarrow error, try correction

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:

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

Appendix

Python code for encoding and decoding

```

1  # %%
2  from typing import Tuple, Dict
3  from functools import reduce
4
5  Matrix = Tuple[Tuple[int, ...], ...]
6
7
8  def transpose(matrix: Matrix) -> Matrix:
9      """Return the transpose of a matrix."""
10     return tuple(zip(*matrix))
11
12
13  # %%
14  G_non_sys: Matrix = (
15      (1, 1, 1, 0, 0, 0, 0, 1),
16      (1, 0, 0, 1, 1, 0, 0, 1),
17      (0, 1, 0, 1, 0, 1, 0, 1),
18      (1, 1, 0, 1, 0, 0, 1, 0),
19  )
20  H_non_sys: Matrix = (
21      (1, 0, 1, 0, 1, 0, 1, 0),
22      (0, 1, 1, 0, 0, 1, 1, 0),
23      (0, 0, 0, 1, 1, 1, 1, 0),
24      (1, 1, 1, 1, 1, 1, 1, 1),
25  )
26  G_sys: Matrix = (
27      (1, 0, 0, 0, 0, 1, 1, 1),
28      (0, 1, 0, 0, 1, 0, 1, 1),
29      (0, 0, 1, 0, 1, 1, 0, 1),
30      (0, 0, 0, 1, 1, 1, 1, 0),
31  )

```

```

32 H_sys: Matrix = (
33     (0, 1, 1, 1, 1, 0, 0, 0),
34     (1, 0, 1, 1, 0, 1, 0, 0),
35     (1, 1, 0, 1, 0, 0, 1, 0),
36     (1, 1, 1, 1, 1, 1, 1, 1),
37 )
38
39 # %%
40 Vector = Tuple[int, ...]
41
42
43 def encode(a: Vector, G: Matrix) -> Vector:
44     """Encode a message vector using a generator matrix."""
45     return tuple(sum(a * b for a, b in zip(a, col)) % 2 for col in transpose(G))
46
47
48 # %%
49 words: Dict[str, Vector] = {
50     "1": (0, 1, 0, 0),
51     "2": (1, 0, 0, 1),
52     "3": (0, 0, 1, 1),
53     "4": (1, 1, 0, 1),
54 }
55
56 print("Task 3, encode with non-systematic matrix")
57 for word, vector in words.items():
58     encoded = encode(vector, G_non_sys)
59     print(f"{word}: {encoded}, Parity: {encoded[-1]} -> {'correct' if sum(
60         encoded) % 2 == 0 else 'incorrect'}")
61
62 # %%
63
64 # %%
65 words_2: Dict[str, Vector] = {
66     "1": (1, 1, 0, 0, 1, 1, 0, 1),
67     "2": (1, 0, 0, 1, 1, 0, 0, 1),
68     "3": (1, 1, 0, 1, 1, 0, 1, 1),
69     "4": (1, 1, 0, 1, 0, 1, 0, 1),
70 }
71

```

```

72
73 # %%
74 def get_syndrome(x: Vector, H: Matrix) -> Vector:
75     """Check if a vector has the correct parity."""
76     return tuple(sum(a * b for a, b in zip(x, col)) % 2 for col in H)
77
78
79 print("Task 4, check with non-systematic matrix")
80 for word, vector in words_2.items():
81     *syndrome, overall_parity = get_syndrome(vector, H_sys)
82     syndrome_number = sum(2**i * bit for i, bit in enumerate(syndrome))
83     syndrome_bits = sum(syndrome)
84
85     print(
86         f"Word {word}: Syndrome with parity: {syndrome + [overall_parity]} ({
87             syndrome_bits}), overall_parity: (p4:{vector[-1]}) -> {'correct' if
88             overall_parity == 0 else 'incorrect'}"
89     )
90     if syndrome_bits == 0 and overall_parity == 0:
91         print("\tno error")
92     elif syndrome_bits == 0 and overall_parity == 1:
93         print("\tError in p4")
94     elif syndrome_bits >= 1 and overall_parity == 1:
95         print("\terror, try to correct it")
96         vector = list(vector)
97         # Search for the column in H that matches the syndrome
98         error_position_mask = reduce(
99             # and all the columns, resulting in a tuple of 1s and 0s where 1s
100             # indicate the error positions
101             lambda total, new: tuple(a & b for a, b in zip(total, new)),
102             # for each row in H, return the row if the syndrome bit is 1,
103             # otherwise return the row with all bits flipped
104             # when anding the rows, the result will be a tuple of 1s and 0s where
105             # 1s indicate the positions the syndrome matches the column
106             (
107                 mask if s == 1 else tuple(val ^ 1 for val in mask)
108                 for s, mask in zip(syndrome + [overall_parity], H_sys)
109             ),
110             # get the positions of the 1s in the mask

```



```
107     error_positions = tuple(i for i, v in enumerate(error_position_mask) if v
108                             == 1)
109     # if there are no error positions or more than 1, the error is
110     # uncorrectable
111     if len(error_positions) == 0:
112         print("\tsyndrome does not match any column in H, multiple errors")
113     elif len(error_positions) != 1:
114         print("\tsyndrome matches multiple columns in H, multiple errors")
115     # if there is exactly one error position, flip the bit at that position
116     else:
117         print(f"\t error position: {error_positions[0]}")
118         vector[error_positions[0]] = vector[error_positions[0]] ^ 1
119         print(f"\tcorrected vector: {vector}")
120         print(f"\tcorrected syndrome: {get_syndrome(vector, H_sys)}")
121     else:
122         print("\tMultiple errors")
123     print()
```