BE2 Traitement et protection de l'information

URIEN, PRÉVOST

March 20, 2023

1 Channel coding laboratory

1.1 1. Block codes

1.1.1 1.1. From a generator matrix

We have the following generator matrix:

[273]:

```
%%file genMatrix.m
G = [[1 0 0 1 1 1 0 1 1 1]
     [1 1 1 0 0 0 1 1 1 0]
     [1 1 0 1 1 1 1 0 0 1]];
```

```
Created file '/Users/thomasprevost/github/ProtectInfo/BE2/genMatrix.m'.
```

[274]:

```
genMatrix;
```

1. First, we generate all possible codewords from the generator matrix:

[275]:

```
%%file genCode.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% genCode.m
% Generates all possible codewords of a given generator matrix
% Input: G - generator matrix
% Output: C - codeword matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function C = genCode(G)
    [m,n] = size(G);
    C = zeros(2^n,n);
    fprintf('Generating code matrix of size %d x %d\n',2^n,n);
    for i = 0:2^n-1
        b = de2bi(i,n);
        C(i+1,:) = mod(b*G(1),2);
    end
end
```

```
Created file '/Users/thomasprevost/github/ProtectInfo/BE2/genCode.m'.
```

[276]:

```
C = genCode(G);
```

```
Generating code matrix of size 1024 x 10
```

2. Then, we calculate the minimum distance of the code:

[277]:
```
%%file minDist.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% minDist.m
% Calculates the minimum distance of a given code
% Input: C - codeword matrix
% Output: d - minimum distance
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function d = minDist(C)
    [m,n] = size(C);
    d = n;
    for i = 1:m
        for j = i+1:m
            if sum(rem(C(i,:)+C(j,:),2)) < d
                d = sum(rem(C(i,:)+C(j,:),2));
            end
        end
    end
end
```

```
Created file '/Users/thomasprevost/github/ProtectInfo/BE2/minDist.m'.
```

[278]:
```
d = minDist(C)
```

```
d =

     1
```

3. Finally, we compare the minimum distance of the code with the minimum distance of a parity code with the same parameters. To do so, we first generate the parity code, then we calculate its minimum distance and compare it with the minimum distance of the code.

[279]:
```
%%file parityCode.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% parityCode.m
% Generates a parity code given its parameters
% Input: n - number of bits
%        k - number of information bits
% Output: C - parity code matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [C, G] = parityCode(n,k)
    G = [eye(k) zeros(k,n-k)];
    C = [G rem(G*G',2)];
end
```

```
Created file '/Users/thomasprevost/github/ProtectInfo/BE2/parityCode.m'.
```

[280]:
```
%%file compareToParity.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% compareToParity.m
% Compares the minimum distance of a given code with the minimum distance
% of a parity code with the same parameters
% Input: G - generator matrix
% Output: d - minimum distance
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function d = compareToParity(G)
    [m,n] = size(G);
    C = genCode(G);
    d = minDist(C);
    [Cp,Gp] = parityCode(n,m);
    dp = minDist(Cp);
    fprintf('Minimum distance of the code: %d\n',d);
    fprintf('Minimum distance of the parity code: %d\n',dp);
end
```

```
Created file '/Users/thomasprevost/github/ProtectInfo/BE2/compareToParity.m'.
```

[281]:
```
compareToParity(G);
```

```
Generating code matrix of size 1024 x 10
Minimum distance of the code: 1
Minimum distance of the parity code: 4
```

1.1.2 1.2. Hamming code

1. First, we generate codes of the Hamming coding $C(15, 11)$:

[282]:
```
%%file hammingCode.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% hammingCode.m
% Generates codes of a hamming coding given its parameters
% Input: n - number of bits
%        k - number of information bits
% Output: H - hamming code matrix
%         G - generator matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [H,G] = hammingCode(n,k)
    A = zeros(n-k, k);
    for i = 1:n-k
        A(i,:) = de2bi(i,k);
    end
    G = [eye(k) A.'];
    H = [G rem(G*G',2)];
end
```

```
Created file '/Users/thomasprevost/github/ProtectInfo/BE2/hammingCode.m'.
```

[283]:

```
%%file genHammingCodes.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% genHammingCodes.m
% Generates all possible codes of a hamming coding given its parameters
% Input: n - number of bits
%        k - number of information bits
% Output: C - codeword matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function C = genHammingCodes(n,k)
    [~,G] = hammingCode(n,k);
    C = genCode(G);
end
```

```
Created file '/Users/thomasprevost/github/ProtectInfo/BE2/genHammingCodes.m'.
```

[284]:

```
[~,Gh] = hammingCode(15,11);
Ch = genHammingCodes(15,11);
```

```
Generating code matrix of size 32768 x 15
```

2. Then, we calculate the minimum distance of the Hamming code:

[285]:

```
dh = minDist(Gh)
```

```
dh =

     2
```

NB : the minimum distance of the Hamming code is given to be $d = 3$. We do not have this value, the error being probably due to a mistake in the calculation of the minimum distance.