Communication System Engineering: Report of TP1

Xinjian OUYANG

January 2021

1 Hamming coding

1.1 Code rate and codeword symbols

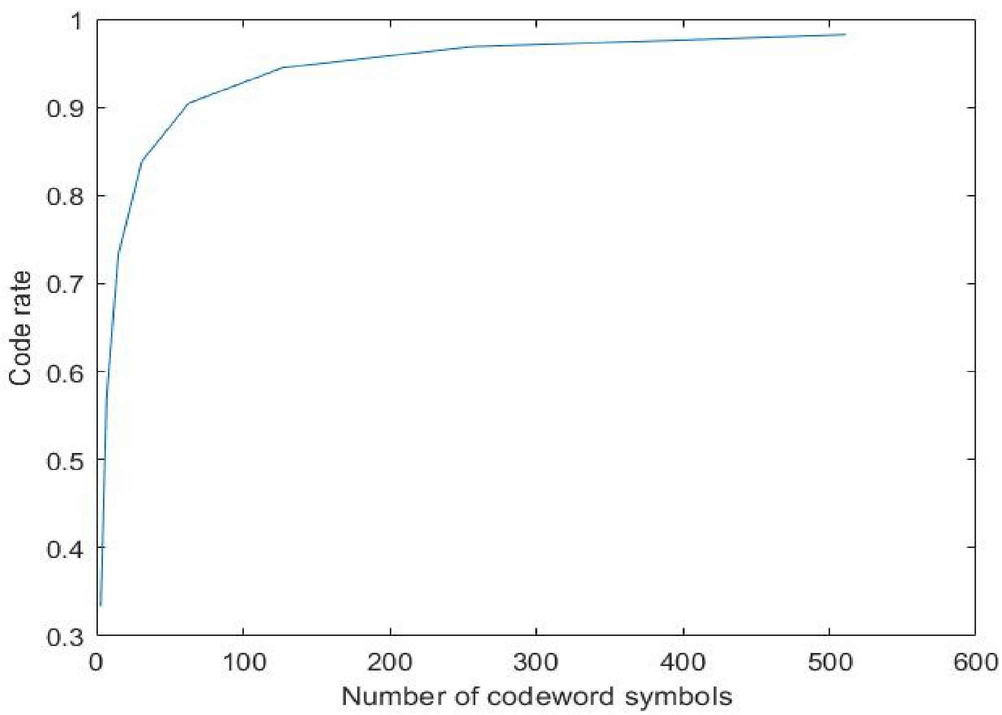

Figure 1: The code rate with respect to the number of codework symsbols

As the Figure shows, **the code rate converges to 1.0 as the number of codeword symsbols goes to infinity**.

For an (n, k) Hamming code, $n = 2^m - 1$ and $k = 2^m - m - 1$, so $n - k = m$ redundant bits are introduced by the code.

$$r = \frac{k}{n} = \frac{n-m}{n} \tag{1}$$

So as n goes to infinity, the code rate r converges to 1.0, which is consistent with the result in Figure 1.

1.2 Generator matrix G and Control matrix H

Through Hamming(7,4), we got $n = 1, k = 4, m = n - k = 3$, where the integer m is the parameter of Hamming code.

Using the Matlab code "[H,G,n,k] = hammgen(3)", we got **the generator matrix G and the control matrix(the parity check matrix) H**:

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}, \tag{2}$$

1.3 Codeword

With the input message $i = [1010]$, the codeword $u = mod(i * G, 2)$,we have

$$u = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}, \tag{3}$$

1.4 Function Hamcode

The hamcode function is as following:

```
function codef = hamcode(msgf,Hf)
    Gf = gen2par (Hf);
    codef = mod(msgf*Gf,2);
end
```

Figure 2: Hamcode function

With $msg = [1010]$, H of the equation(2), we use *encode* function to check the function hamcode as the Figure 3 shows.

Then we got **the code** = [0011010] **equals to code-check** = [0011010]**. Thus the hamcode function works**.

```
%% 1.4 hamcode
code = hamcode(msg,H) ;
%check using encode
code_check = encode(msg,n,k,'hamming/binary');
if  isequal(code,code_check)
    disp("Hamcode Success!The result of hamcode is consistent with that of encode.")
else
    disp("Failed!")
end
```

Figure 3: Hamcode checking

1.5 Function Hamdecode

With a randomly generated R(the received code), we use *decode* function to check the function hamdecode as the Figure 4 shows.

Then we got **the result msgr** = [1100] **equals to msgr-check** = [1100]**. Thus the hamdecode function works**.

```
%generate a received code using randi
R = randi([0,1],[1,n]);
msgr = hamdecode(R,H);

%decode check
msgr_check = decode(R,n,k,'hamming/binary');

if  isequal(msgr,msgr_check)
    disp("Hamdecode Success!The result of hamdecode is consistent with that of decode.")
else
    disp("Failed!")
end
```

Figure 4: Hamdecode checking

The hamdecode function is as the figure 5 shows:

```
function msgr = hamdecode(Rf,Hf)
    %calculation of syndrome
    syndrome = rem(Rf * Hf',2);
    % error location & Convert to decimal.
    syndrome_de = bi2de(fliplr(syndrome));
    %truth table
    trt = syndtable(Hf);
    % Correction vector & error location
    corrvect = trt(1+syndrome_de,:);

    % Now compute the corrected codeword.
    codec = rem(corrvect+Rf,2);

    % corrected message
    n = size(Hf,2);
    k = size(Hf,2) - size(Hf,1);
    I = eye(k);
    Gf = gen2par(Hf);
    if isequal(Gf(:, 1:k) ,I)
        msgr = codec(:, 1:k);
    else isequal(Gf(:, n-k+1:n), I)
        msgr = codec(:, n-k+1:n);
end
```

Figure 5: Hamdecode function

1.6 BER and SNR

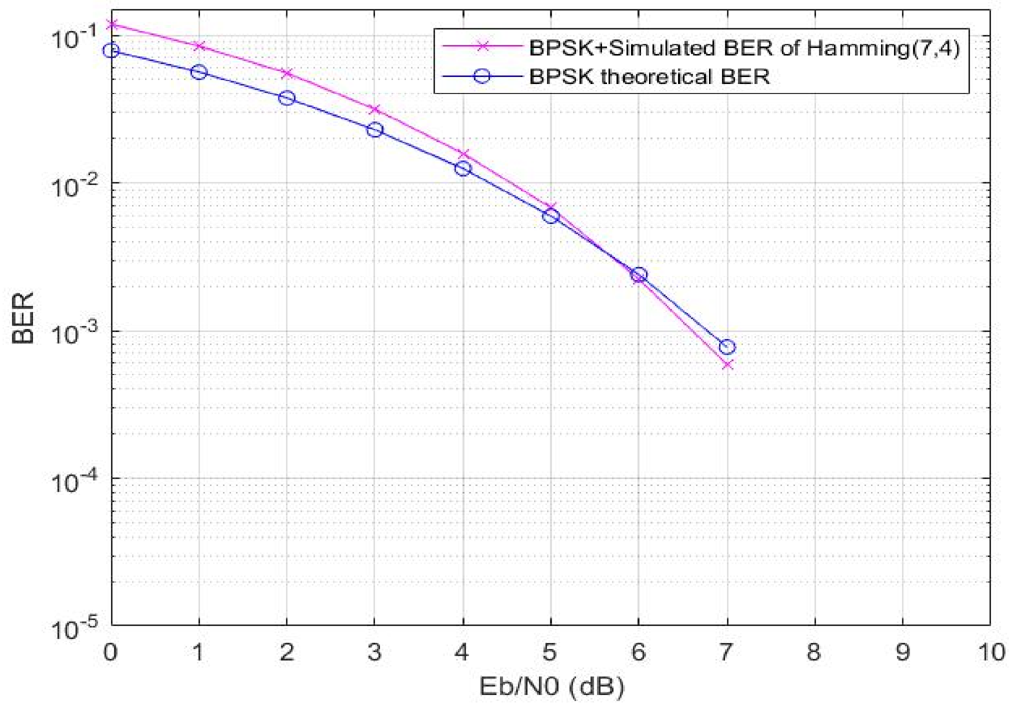

Figure 6: The BER with respect to the SNR

As $Eb/N0$(SNR) increases, the BER(Bit Error Rate) decreases. And the simulated BER is well consistent with the theoretical one.

2 Matlab code

Matlab Code Source