AN-code 與 Hamming Code BER/BLER 分析與模擬

為了測試與了解 AN-Code 的可靠度，我利用 matlab 的 simulink 建立利用 PBSK hard decision 通訊系統。

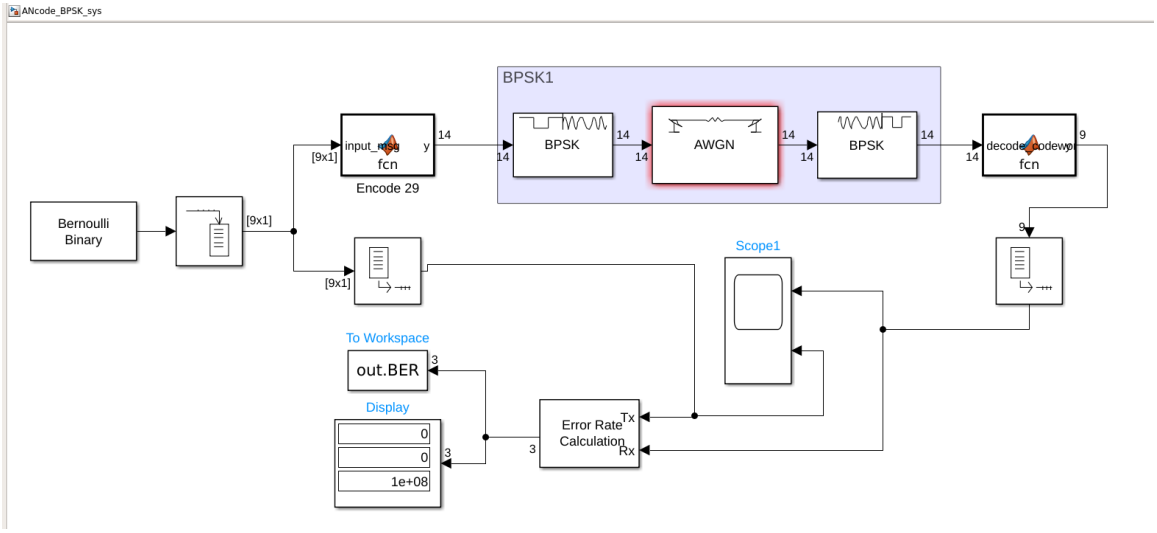

Figure 1 System Diagram

我模擬了三種情況

1. Uncode
2. HammingCode(7,4)
3. A=29 ANCODE

在三種情況下，模擬測試環境設定為

- 傳輸最大 bit 數:1e8
- 測試錯誤數:100

也就是說，在同 Eb/N0 模擬環境下，模擬系統會持續傳輸直到傳輸 1e8 個 bit 資料為止或者當錯誤的 bit 數累積到 100 個就進行下一個 Eb/N0 的模擬。

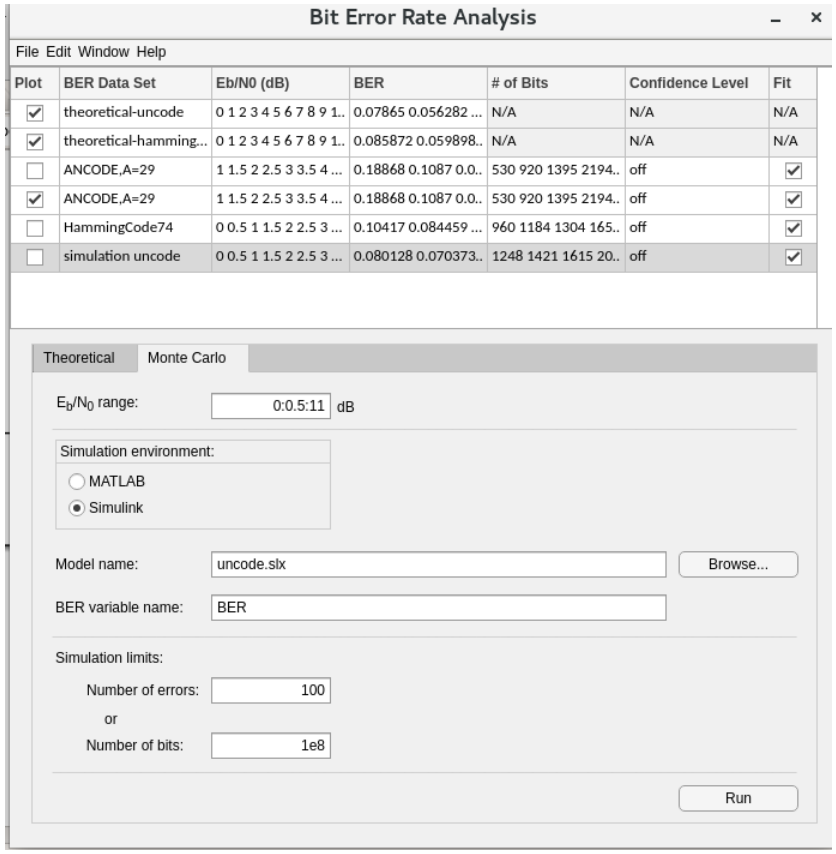

Figure 2 BERTOOL Setting

A=29 AN-Code

A=29 時，其能更正的錯誤範圍為$\frac{29-1}{2} = 14$ bit

其能傳輸的最大訊息為 $\frac{2^{14}}{29} = 564_{10} \leq 2^9$

碼長為 $\log_2(512 \times 29) = 13.85 \leq 14$

每傳輸 9 bit 能夠被 encode 成 14bit AN code

Figure3 為 matlab 的模擬結果

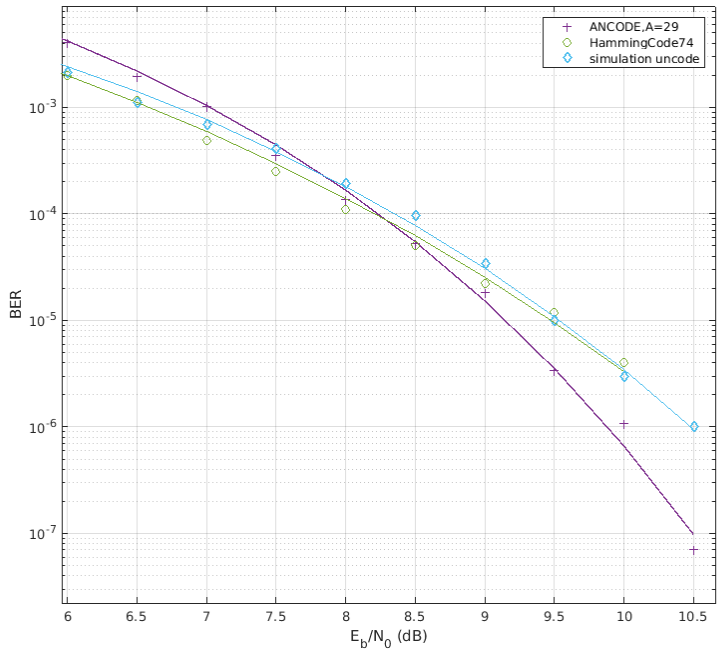

Figure 3 BER SNR

可以看見在 Eb/N0=8.5 時，AN code 的 BER 比 hammingCode 還要低，說明了在雜訊越低的情況下，AN code 的抗雜訊能力越好。

BLER (Block Error Rate) 分析

以 A=29，一個 block 碼長為 14bit 的情況下，分析 BLER 的方式如下:

假設 BER/SNR=0.1/10(db)，也就是在 SNR 為 10db 時，BER 為 0.1 的情況下，BLER 為

$$1-(1-0.1)^{14} \cong 0.77$$

根據這個算法，在依據 BER/SNR 的結果可以做出 BLER/BER 的結果。

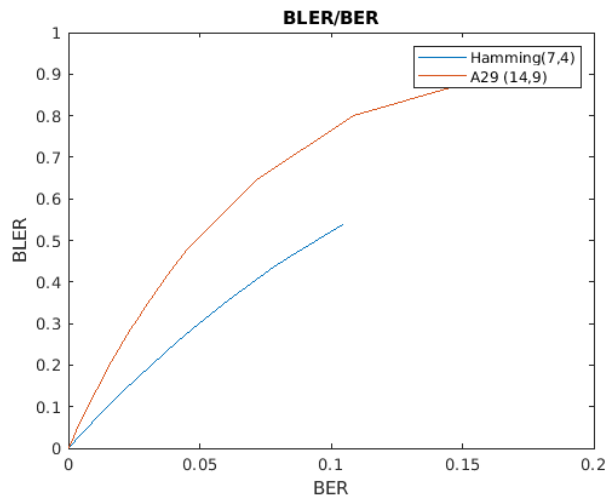

Figure 4 BLER/BER

由 Figure4 可以發現不論在任何 BER 下 AN code 的 BLER 錯誤率都比較高，我想這是由於 AN code 的一個 block 碼長為 14bit，要全部都對的情況會比 hamming code 的 7bit，機率還要低。

Figure 5 是將結果轉為 log scale 的圖形

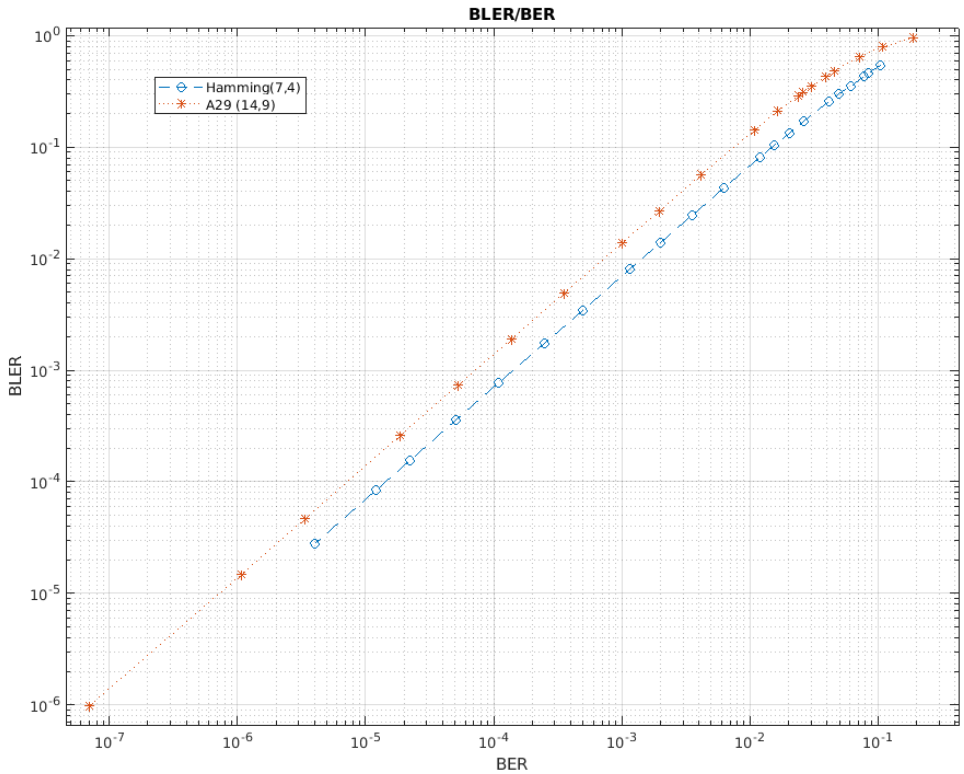

Figure 5 BLER/BER x log scale

結論

由 Figure5 可以看出不論 BER 在何種情況下，AN code 的 BLER 都比較差，倘若要證明或者展現 AN code 在神經網路的可靠度，那我考慮用以下方式說明:

- 在硬體架構中，comparator 前面加入 error，以此方式直接測試準確度下降多寡，來比體現有 encdoe 與 uncode 之間的差別。
- 推論出的 BLER 表現雖然比較差，但並不代表準確度就會因此下降，在 BLER 中的錯並不一定會直接影響判斷結果。

Appendix

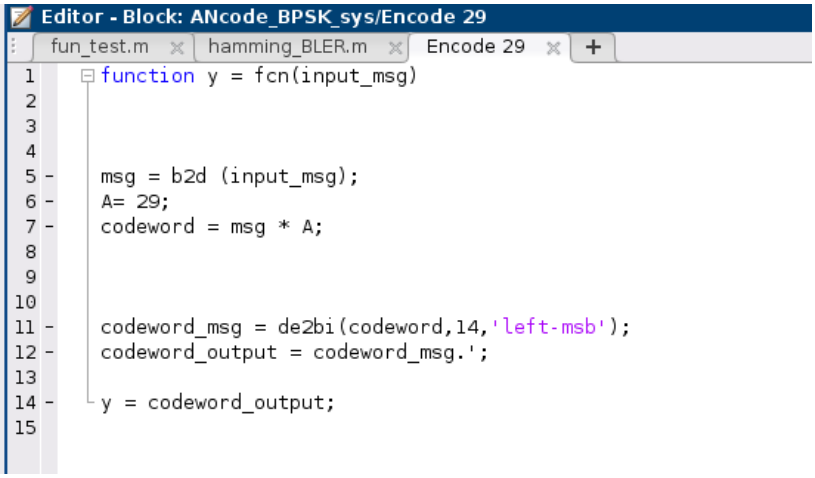

Figure 6 matlab Encode Code

```
function y = fcn(decode_codeword)
    decimal_codeword = b2d(decode_codeword);
    A=29;
%     decode_msg = decimal_codeword/A;
    residue = mod(decimal_codeword,A);
    quotient = floor(decimal_codeword/A);
        switch residue
            case  1
                 quotient = quotient;
            case  2
                 quotient = quotient;
            case  4
                 quotient = quotient;
            case  8
                 quotient = quotient;
            case  16
                 quotient = quotient;
            case  3
                 quotient = quotient- 1 ;
            case  6
                 quotient = quotient- 2 ;
            case  12
                 quotient = quotient- 4 ;
            case  24
                 quotient = quotient- 8 ;
            case  19
                 quotient = quotient- 17 ;
            case  9
                 quotient = quotient- 35 ;
            case  18
                 quotient = quotient- 70 ;
            case  7
                 quotient = quotient- 141 ;
            case  14
                 quotient = quotient- 282 ;
            case  28
                 quotient = quotient+1;
            case  27
                 quotient = quotient+1;
            case  25
                 quotient = quotient+1;
            case  21
                 quotient = quotient+1;
```

Figure 7 Matlab decode code (1)

Editor - Block: ANcode_BPSK_sys/MATLAB Function

fun_test.m | hamming_BLER.m | Encode 29 | MATLAB Function

```
39 -                 quotient = quotient+1;
40                 case  25
41 -                 quotient = quotient+1;
42                 case  21
43 -                 quotient = quotient+1;
44                 case  13
45 -                 quotient = quotient+1;
46                 case  26
47 -                 quotient = quotient+ 2 ;
48                 case  23
49 -                 quotient = quotient+ 3 ;
50                 case  17
51 -                 quotient = quotient+ 5 ;
52                 case  5
53 -                 quotient = quotient+ 9 ;
54                 case  10
55 -                 quotient = quotient+ 18 ;
56                 case  20
57 -                 quotient = quotient+ 36 ;
58                 case  11
59 -                 quotient = quotient+ 71 ;
60                 case  22
61 -                 quotient = quotient+ 142 ;
62                 case  15
63 -                 quotient = quotient+ 283 ;
64
65                 otherwise
66 -                 quotient = quotient;
67
68             end
69
70 -    quotient2 =quotient;
71 -        if(quotient2<0)
72 -            quotient=0;
73          end
74
75 -        if(quotient2>=(2^9))
76 -            quotient=0;
77          end
78
79 -        binary_decode = de2bi(quotient,9,'left-msb');
80 -        u=binary_decode.';
81 -    y = u;
```

Figure 8 Matlab decode code (2)