**ELEC 5507 Error Control Coding**

**Group Project**

**1) Introduction**

The main goal of error control coding is to encode original messages for channel transmission with errors, and the receiver can correct the error bits and recover the original messages.

In this project, the state-of-art forward error correcting (FEC) encoders and decoders will be simulated and analysed by using MATLAB.

In Section I, the BCH code will be designed and simulated over BSC and AWGN channels, and BCH encoder and decoder will be used to analyse the sound file (austinpowers.wav). After that, the coding gain of BER will be obtained for coded and uncoded systems.

In section II,

**2) Theoretical background**

**BCH Generator polynomial**

The generator polynomial g(X) of a t-error-correcting primitive BCH code of length

is given by

A description...

Where is the minimum polynomial of the field element .

**Encoding**

Since BCH code is included in cyclic code, all codewords can be generated by using generator polynomial g(x) in the same way that of a cyclic code.

A message of k bits is encoded into a length of n bits codeword. The n-k redundant bits are used for protection against errors. The codeword

v = c\*G = , where G is the generator matrix shown below.

The Generator polynomial genpol = [1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1].

To generate a systematic BCH code, c(X) needs to be multiplied with to obtain

A description...

And then dividing by g(X), we have , where b(X) is the remainder. So the code polynomial is given by v(X) = b(X) +.

**Decoding**

For BCH decoding, there are three methods:

* MATLAB defined “decode” function which is

‘msg = decode(code,n,k,’cyclic/fmt’,genpoly,trt)’, where ‘genpoly' is the BCH generator polynomial, and ‘trt’ is the syndrome decoding table associated with the method's parity-check matrix.

* By using the syndrome lookup table, the codeword can be decoded.

The syndrome is calculated from codeword \* H’, where H is the parity-check matrix. And every syndrome is corresponded to a most probable error pattern E out of all error patterns, i.e. the one with the minimum Hamming weight.

….

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | … |  | … |  |
|  |  | … |  | … |  |
| … | … | … | … | … | … |
|  |  | … |  | … |  |
| … | … | … | … | … | … |
|  |  | … |  | … |  |

So to construct the standard array, there are rows ( syndromes) and columns ( codewords), all error patterns with smallest hamming weight are used to calculate the associated possible received vectors. Every column of a standard array consists of one and only one codeword, and all the other vectors are sums of the codeword and the coset leaders.

A description...

The decoding rule: If the received vector r is found in the j-th column Dj, the r is decoded into the codeword vj.

* By using Berkelamp’s iterative algorithm, the syndromes are needed to be calculated first, . And then to find error location polynomial by using Berlekamp’s iterative algorithm

A description...

A description...

After that, the error location numbers can be determined by finding the roots of . And then the received codeword can be corrected.

**LDPC algorithm**

**3) Design & Implementation**

**Section One**: Implementation of BCH code, n=31, k=16, and t=3, m=5

**Q1**: The generator polynomial g(x) of 3-error-correcting primitive BCH code of length n is,

g(X) = LCM (p32 lec5)

= x5+x2+1

=x5+x4+x3+x2+1

=x5+x4+x2+x+1

so g(X) =

By using MATLAB’s ‘bchgenpoly’ function, the result is verified the same with above.

bchpol = bchgenpoly (n, k)

bchpol = [1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1]

**Q2**: For BCH code,

Here t=3, so, , dmin=7

**Q3**: As discussed in theoretical background, the size of standard array is. Here, the syndrome lookup table contains two columns, one is for syndromes, the other column is corresponded error patterns (coset leaders).

The MATLAB codes below are used to generate the syndrome decoding table using function ‘syndtable’. And the function ‘cyclgen’ is used to generate the parity-check matrix H and generator matrix G.

bchPol = [1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1];

pol = cyclpoly(31,16,'max');

for index = 1:16

    pol(index) = bchPol(index);

end

[H,G]=cyclgen(31,pol);

trt = syndtable (H)

The syndrome decoding table contains 32768 rows, and the sub-array of the first 5 rows are taken below. The full corresponding table is trt.\*txt file.

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

And the corresponded syndromes are shown below.

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

**Q4**: Corresponding to the syndrome decoding table in Q3,

**Q5**: As discussed in theoretical background, the message c(X) needs to be multiplied with , and then dividing by g(X) to obtain the remainder b(X). And the codeword polynomial is given by v(X) = b(X)+.

The MATLAB codes below shows the process of generating systematic BCH codewords. The function ‘gfdeconv’ is used to get the remainder b(X) of dividing by g(X).

function codeword = polBCHencoder(msg)

genPol = [1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1];

d = [msg zeros(1,15)];

[~, r] = gfdeconv(fliplr(d), fliplr(genPol));

r = [zeros(1,15 - length(r)) fliplr(r) zeros(1,16)];

codeword = r + [zeros(1,15) msg];

**Q6**: The BCH decoding part contains three methods:

* This decoding is done by using MATLAB’s function ‘decode’. The codes are shown below, and ‘trt' is the syndrome decoding table.

function decoded\_data = matlabBCHdecode(data\_to\_decode)

load trt

bchPol = [1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1];

decoded\_data = decode(data\_to\_decode,31,16,'cyclic/fmt', bchPol, trt);

* The syndrome can be obtained by multiplying received codeword r with transposed parity-check matrix H, received\_codeword \* H’, and the corresponded error pattern can be found in the decoding table, and then the corrected code = received\_codeword + error pattern.

function msg = syndLookupDecode(codeword)

load trt

H = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0

        0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0

        0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0

        0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1

        0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 1

        0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 1 1

        0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 1

        0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1

        0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 0 1 0

        0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 0 1

        0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 0

        0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1

        0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1

        0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1

        0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 1];

syndrome = bi2de(mod(codeword \* H',2),'left-msb');

errLocations = trt(1+syndrome,:);

correctedcode = mod(errLocations+codeword,2);

msg = correctedcode(:,16:end);

* Berkelamp’s iterative

**Q7**: Simulations and sound analysis

**Q8**: Simulations

**Section Two**

**5) Conclusion**