Coding Theory in Mathematica

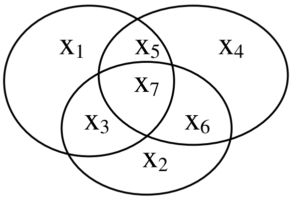
Igor Gachkov & Kenneth Hulth
School of Engineering
Jonkoping University. Sweden

Preface

Applications of the theories of Error-Corecting Codes have in recent years increased tremendously. Thus it is today hardly possible to imagine engineers working with data transmission and related fields without basic knowledges of coding/decoding of information.
The possibilities of quantifying information with electronical equipment are developing rapidly, supplying the specialists working in communication theory with more sophisticated methods for cicuit realisation of concrete algorithms in Coding Theory. During preceding years courses in Coding Theory have been concidered only for students on postgraduate level. This is due to the complexity of the mathematical methods used in most codes, such as results from abstract algebra including linear spaces over Galois Fields.
With the introduction of computers and computer algebra the methods can be fairly well illustrated. The autors have developed a course, 'Coding Theory in Mathematica' , using the wide range of capabilities of MATHEMATICA. The course was given during the autumn 1994 at Jönköping University, Sweden, on undergraduate level with a minimum of prerequisites. The hands on sessions were based on a package of application programmes/algorithms , developed to illustrate the mathematical constructions, used in coding theory to encode and decode information.
We will in our seminar present some of our hands-on material, which are used to construct Block Codes with means of algebraical methods. We first present the basic concepts of coding theory, such as hamming distance, generator and parity check matrices,binary linear codes and group codes. We will then use some basic results from matrix algebra and linear spaces to construct code words from the information we want to send. Due to noise in the channel of transmission the recieved words may differ from the code words which were sent, and the very aim of coding theory is to detect and correct possible errors.
In the linear codes (the code words having group structure) the encoding is accomplished by multiplying the information word (seen as a vector) with a generator matrix, whereas the decoding process starts with multiplication of the recieved word with the parity check matrix.
Within the cyclic codes it is preferable to work with generator- and parity check polynomials instead of matrices, and the code words here form a polynomial ring. With algebraic extensions of the field Z_p by irreducible (over Z_p) polynomials, the final step is taken into the BCH-codes, which have a rich algebraical structure.
The application programmes in the package support the learning processes by illustrating the algorithms in the mathematical constructions. The rich variety of manipulations in the algebraical structures, the possibility to vary the parameters in specific cases and the simplicity to construct concrete codes with MATHEMATICA should strengthen the understanding of the mathematical ideas used in coding theory.

Introduction to Coding Theory

As an introduction to the theory of Error-Correcting Codes we study the well-known Venn-Diagram with the three overlapping circles:



We place 7 pieces x_i, where x_i is marked with the number (1) on one side and (0) on the other, on the respective i:th area of the figure. This could be performed in such a way, that the sum of the numbers in each circle is even, i.e for each of the circles applies $\Sigma x_i = 0$ (mod 2). We will see that there are a total of 16 such possibilities, and if we write these $x = x_1x_2x_3...x_7$ we have created a code with 16 codewords. If we transmit such a codeword through a communication channel, it may happen, due to noise in the channel, that one (or several) of the x_i changes from 0 to 1 or vise versa. In our figure the error is easily found by checking the sum (mod 2) of the numbers in each of the circles. If we change x_i for exactly one i the conditions $\Sigma x_i = 0$ (mod 2) will be fullfilled. We thus see, that there do exist possibilities to correct errors in the transmission of information.

We now turn to the figure again. The condition $\sum x_i = 0$ for each of the tree circles immediately gives (if we perform the operations modulo 2, i.e. within the field Z_2)

$$\begin{array}{rcl} x_4 + x_5 + x_6 + x_7 = 0 & & x_7 = x_1 + x_2 + x_4 \\ x_2 + x_3 + x_6 + x_7 = 0 & \Leftrightarrow & x_6 = x_1 + x_3 + x_4 \\ x_1 + x_3 + x_5 + x_7 = 0 & & x_5 = x_2 + x_3 + x_4 \end{array} \qquad (1)$$

It follows that in order to place the 7 pieces correctly, we simply place $x_1 ... x_4$ freely and then compute x_5, x_6 and x_7 in accordance with the equations. This means, that out of a total of 2^7 combinations of $x_1....x_7$ there will be 2^4 combinations satisfying our condition, i.e. from 2^7 *words* there will be 2^4 *codewords* .

In order to find the codewords we could also proceed as follows: The word $v= v_1v_2...v_7$ would be a codeword iff (1) is satisfied, which in matrixform means iff v satisfy

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} v^{tr} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \Leftrightarrow \quad H\, v^{tr} = \mathbf{0}$$

In order to check whether a word w recieved from the communication channel is a code word , we only compute H w^{tr} = s^{tr} , where s is called the syndrom, if s^{tr} = **0**, then v is a code word. If exactly one error occurs in the transmission, then exactly one x_i would be false, and we write (as vectors) w = v + e_i , where v is a code word and e_i has a single nonzero bit, in the position i .
We compute H v^{tr} = H ($v^{tr}+e_i^{tr}$) = (linearity !) = H v^{tr} + H e_i^{tr} = 0 + H e_i^{tr} = the i:th column in H. One of these columns has to be in accordance with s^{tr} and so we correct the corresponding $v_{i.}$ It can easily be proved, that we would always find the error, provided we have exactly one false bit. In general, if H is a binary matrix, the linear code with the *parity check matrix* H consists of all vectors v, satisfying H v^{tr} = **0**. Usually H is an (n-k) x n matrix H = [A I_{n-k}] , with I_{n-k} the unit matrix (n-k) x (n-k) . With the information word u = $u_1 \ldots u_k$ we write the codeword v = $v_1 \ldots v_k v_{k+1} \ldots v_n$, where $v_i = u_i$ $1 \leq i \leq k$ and where $v_{k+1} \ldots v_n$ are the check symbols. We then have H v^{tr} = **0** $\Leftrightarrow$ **x = u**G , where G = [I_k A^{tr}] . G is called the generator matrix of the code.

We have in our introductory example endeed demonstrated the *Hamming Code* K[7,4,3] , where the parameter 7 indicates the length and 4 the dimension (=number of information bits) of the code. The parameter 3 gives the *hamming distance* d of the code: the hamming distance d(x,y) between the words x and y equals the number of positions i, where $x_i \neq y_i$; d = min d(x,y) = minimum distance between any two codewords. d is easily found to be equal to the minimum *hamming weight* wt(v) of any codeword v, where wt(v) is the number of nonzero v_i .
The Hamming Code K[n,k,d] is characterized by (let m be the number of check bits, $m \geq 2$)
n = 2^m-1, k = 2^m-1-m , d=3 and is a <u>perfect</u> single-error-correcting code, meaning that the code has exactly the information needed for correcting one error. In this case, every possible recieved word is either a code word or has a hamming distance equal 1 to exactly on code word.

The package "Coding Theory"

The package " Coding Theory" is a file written in MATHEMATICA Version 2.0 and will be read into MATHEMATICA with the commands.

In[1] : = <<CodingT1.m

In[2] : = <<CodingT2.m

The package consists of two parts: one part with illustrative explanations, and one for scientific purposes.
The illustrative part (commands starting with Show...) is considered to visualize the theoretical aspects of encoding / decoding , construct shift-register circuits etc.

The command ? Show* gives the following list of commands

In[3] : = ?Show*

Show	ShowHammingCode
ShowBinaryGaloisField	ShowMeggittDecoder
ShowBurstRectangularCodeVector	ShowNonsystematicEncoderCyclicCode
ShowCorrectBurstRectangularCode	ShowReconstructInformRectangularCode
ShowCyclicCode	ShowRectangularCode
ShowDecHammingCode	ShowStandardArrays
ShowDecRectangularCode	ShowSystematicEncode
ShowErrorTrappingDecoderBCHCode	ShowSystematicEncoderCyclicCode
ShowTablePlus	ShowTableMult

The complete information about a command is received by using the command ? *Name* .

In[4] : = ? ShowHammingCode
ShowHammingCode[m,inf] shows the method of encoding an information word inf into a hamming code word of length 2^m - 1 .

We now use MATHEMATICA to construct Hamming codes. In the Example we first chose m = 3 which gives $n=2^3 - 1 = 7$, i.e. the code K[7,4,3].

In[5]:=
inf={1,1,1,0};
ShowHammingCode[3,inf]

```
PARITY
CHECK
MATRIX = 0 0 0 1 1 1 1
         0 1 1 0 0 1 1
         1 0 1 0 1 0 1

GENERATOR
MATRIX = 1 1 0 1 0 0 1
         0 1 0 1 0 1 0          (* Remember H  v =0   ⇔   v = u G *)
         1 0 0 1 1 0 0
         1 1 1 0 0 0 0

inf = {1,1,1,0}
v = inf * G
HAMMING CODE VECTOR v = {0,0,0,1,1,1,1}
```

We proceed by sending the code word v and let an error appear in the 5:th position. We thus decode the recieved word w = 0001011.

In[6]:=

```
v={0,0,0,1,1,1,1};
e={0,0,0,0,1,0,0};
w=Mod[v + e,2];
ShowDecHammingCode[3,w]
```

```
PARITY
CHECK
MATRIX = 0 0 0 1 1 1 1  SYNDROME = 1
         0 1 1 0 0 1 1             0        (*   Syndrome s^tr = H w^tr  *)
         1 0 1 0 1 0 1             1

POSITION OF ERROR = 5

RECEIVED WORD = {0,0,0,1,0,1,1}
DECODED WORD =  {0,0,0,1,1,1,1}
inf ={1,1,1,0}                  (*  We have used nonsystematic encoding and must
                                    perform additional computetion to find  the
                                    information word. In the systematic encoding
                                    we get the information word immediately as will be
                                    shown later*)
```

We now compute the parameters of the code.

In[7]:=

```
H = HammingMatrix[3];
MatrixForm[H]
Length[H[[1]]]
DimensionCode[H]
DistanceCode[H]
```

Out[7] =

```
0 0 0 1 1 1 1
0 1 1 0 0 1 1
1 0 1 0 1 0 1
```

Out[8] =

```
7
```

Out[9] =

```
4
```

Out[10] =

```
3
```

Fundamentals from algebra.

We need for the construction of the linear block-codes some prerequisites from algebra, such as structures of groups, cosets, rings and finite fields. The linear block codes are seen as subspaces of linear spaces over a finite field. Of fundamental importance are the following algebraical concepts:

* Irreducible polynomial : f(x) is irreducible over the field F, if f(x) cannot be factored as a product of two polynomials of degrees smaller than that of f(x).

* Primitive element : α is a primitive element in the field F provided that every nonzero element in F is equal to some power of α.

* Minimal polynom: If β is an element of some extension of the field F, the minimal polynomial of β (with respect to F) is the lowest-degree monic polynomial M(x) over F with $M(\beta) = 0$. The minimal polynomial M(x) of $\beta \in GF(p^m)$ can be computed as

$$M(x)=\left(x-\beta\right)\left(x-\beta^{p}\right)\left(x-\beta^{p^2}\right)\ldots\left(x-\beta^{p^i}\right),\text{where}\,\beta=\beta^{p^{i+1}}.$$

* The Galois field $GF(p^m)$ is defined as an algebraic extension of the field Z_p (p prime number) by an irreducible polynomial of degree m. If g(x) is a polynomial of degree m and irreducible over Z_p we construct $GF(p^m) = Z_p / g(x)$ with p^m elements $\sum_{i=0}^{m-1} a_i \alpha^i,\quad a_i \in Z_p$.

  The element α satisfies $g(\alpha) = 0$.

In MATHEMATICA we now demonstrate some algebraical constructions and computations :

In[11]:=
ShowtablePlus[5]

+	0 1 2 3 4
0	0 1 2 3 4
1	1 2 3 4 0
2	2 3 4 0 1
3	3 4 0 1 2
4	4 0 1 2 3

In[12]:=
ShowTablemult[5]

*	0 1 2 3 4
0	0 0 0 0 0
1	0 1 2 3 4
2	0 2 4 1 3
3	0 3 1 4 2
4	0 4 3 2 1

We recognize the field structure of Z_5, as every element is seen to have a multiplicative invers.

In[13]:=
BIP = BinaryIrreduciblePolynomials[3,x]   (*All irreducible polynomials of degree 3*)
Out[13] =
$\{1 + x^2 + x^3,\ 1 + x + x^3\}$

In[14]:=
irrpol = BIP[[2]];
ShowBinaryGaloisField[irrpol, x , b , b]

(* GF(8) is recieved by extending the field Z_2 by the irreducible polynomial $g(x) = 1 + x + x^3$. Observe that b is a primitive element in this extension field. *)

Galois Field GF(8)
with irreducible polynomial
$1 + x + x^3$

Log	Vector	pr.el.	polynomial	min polynomial
-oo	000	0	0	x
0	100	1	1	$1 + x$
1	010	b	b	$1 + x + x^3$
2	001	b^2	b^2	$1 + x + x^3$
3	110	b^3	$1 + b$	$1 + x^2 + x^3$
4	011	b^4	$b + b^2$	$1 + x + x^3$
5	111	b^5	$1 + b + b^2$	$1 + x^2 + x^3$
6	101	b^6	$1 + b^2$	$1 + x^2 + x^3$

We illustrate the multiplication of the elements $1 + b + b^2$ and $1 + b^2$ in GF(8):

In[15] :=
PolynomialMod[
PolynomialMod[(1 + b + b^2)*(1 + b^2), irrpol /. x - > b], 2]
Out[15] =
$b + b^2$

Cyclic codes

The linear code K is called cyclic, if $v = v_0v_1\ldots v_{n-1} \in K \Leftrightarrow v' = v_{n-1}v_1\ldots v_{n-2} \in K$. If we then represent the vector $v = v_0v_1\ldots v_{n-1}$ with the polynomial $v(x) = \sum_{i=0}^{n-1} v_i x^i$, the cyclical shift of v to $v^`$ corresponds to a multiplication of v(x) with x. If g(x) is a polynomial with lowest degree in K, it can be shown, that g(x) is uniquely determined and generates K (i.e. every other codeword c(x) in K could be written c(x)=q(x) g(x)) and is thus called the *generator polynomial* of K.

g(x) is a divisor of $x^n - 1$, has degree (n-k) and we have the following connection between the generator matrix and the generator polynomial:

$$G = \begin{pmatrix} g(x) \\ xg(x) \\ . \\ x^{k-1} g(x) \end{pmatrix}$$

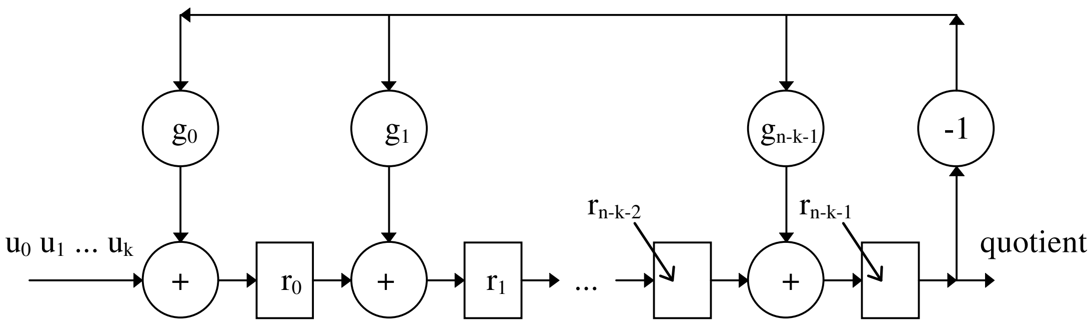
In a cyclical code we now construct the code word v(x) from the information word u in following steps:

* $u = u_0 \ldots u_k \Rightarrow u(x) = u_k x^{n-1} + u_{k-1} x^{n-2} + \ldots + u_0 x^{n-k}$

* construct r(x) as the remainder in u(x) = q(x) g(x) + r(x)

* let the code word be v(x) = u(x) - r(x) (= q(x) g(x))

These algebraical operations could be performed by the *shift register circuit:*



In MATHEMATICA we illustrate the procedure with the Hamming code K[7,4,3] :
We first recieve the generator polynomials to the code by the command:

In[16] : = gp=GeneratorPolynomials[7,x] (* This command gives all possible g(x) of length 7 *)

Out[16] =
$\{1, 1 + x, 1 + x + x^3, 1 + x^2 + x^3, 1 + x + x^2 + x^4,$
$1 + x^2 + x^3 + x^4, 1 + x + x^2 + x^3 + x^4 + x^5 + x^6, 1 + x^7\}$

In[17] : = g=gp[[3]] (* We choose the third polynomial as g(x) i.e. $g(x) = 1 + x + x^3$ *)

Out[17] =
$1 + x + x^3$

In[18] : =
ShowCyclicCode[g,7,x]

GENERATOR
POLYNOMIAL $g = 1 + x + x^3$

GENERATOR MATRIX GM =

$$\begin{matrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$

PARITY
CHECK POLYNOMIAL h = $1 + x + x^2 + x^4$ (* $h(x)\ g(x) = x^n - 1$ *)

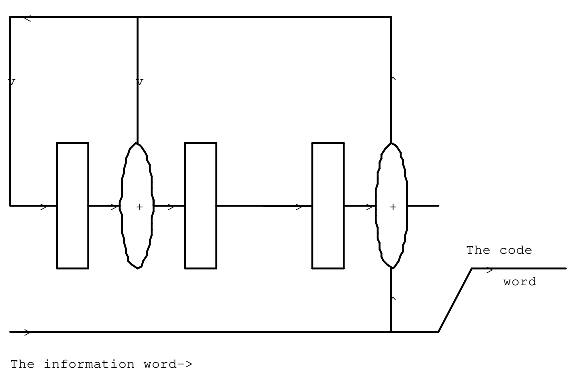
PARITY
CHECK MATRIX HM =

$$\begin{matrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{matrix}$$

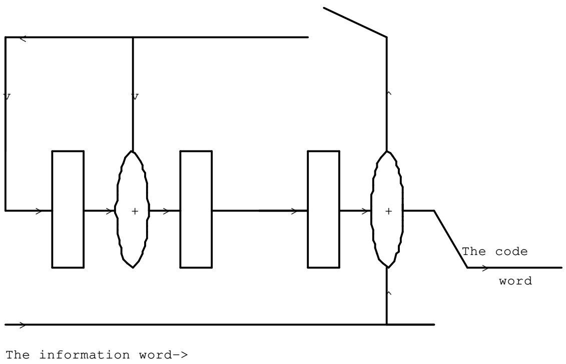
From MATHEMATICA we then get the shift-register (systematic encoding)

In[19] : =
ShowSystematicEncoderCyclicCode[g,7,x]

1. The first 4 shifts.
(the length of information word =4)



2. The last 3 shifts



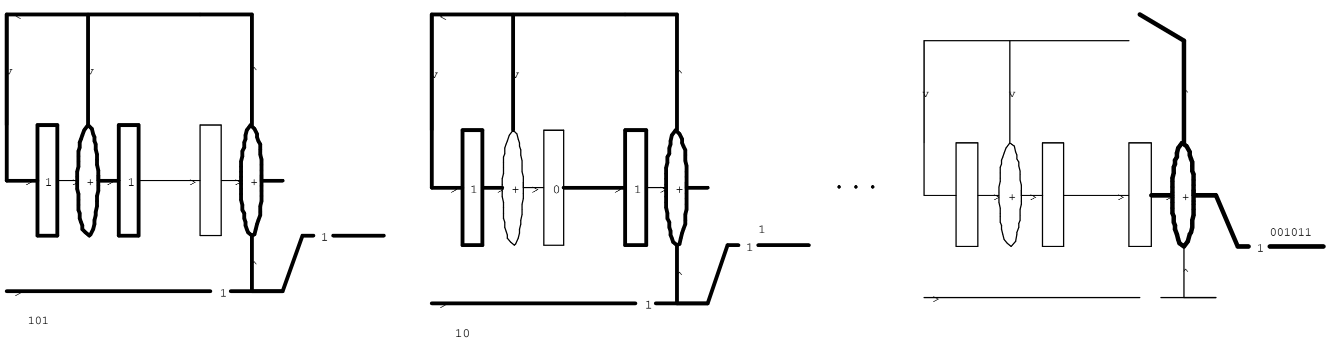
With the information word u = 1011 we get the code word as follows:

In[20] : = u={1,0,1,1};
U=x^6+x^5+x^3;
r=PolynomialMod[PolynomialMod[U,g],2];
V=PolynomialMod[U-r,2]
v=CoefficientList[V,x]
Out[20] =
$1 + x^3 + x^5 + x^6$
Out[21] =
{1, 0, 0, 1, 0, 1, 1}

Using the package we get :

In[22] : =
SystematicEncodeCyclicCode[g,u,7,x]

THE INFORMATION POLYNOMIAL = $x^3 + x^5 + x^6$



THE CODE POLYNOMIAL = $1 + x^3 + x^5 + x^6$
THE CODE WORD = 1001011

From the information word u=1011 we have thus got the code word
$v(x) = 1 + x^3 + x^5 + x^6 = 1001011$ wich will be sent through the communication channel.
Due to noise in the channel, the reciever might recieve $w(x) = v(x) + e(x)$, where $e(x)$ is an error vector. We illustrate this in our example by letting $e(x) = x^4$, i.e $w(x) = 1 + x^3 + \mathbf{x^4} + x^5 + x^6$.
Our next aim is to decode the received word w = 1001111.

In general, when we recieve a word w from the communication channel we use the concept of syndromes to reconstruct the information word. For a general linear code this requires the following steps:

* compute the syndrome $s^{tr} = H\, w^{tr}$

* divide the received words into cosets and fix a coset leader e with smallest hamming weight to each coset

* decode as $v = w - e$

Within the cyclic codes we construct the syndrome polynomial $s(x)$ from $w(x) = q(x)\, g(x) + s(x)$
As $\deg s(x) < \deg g(x)$, we find the syndromes $s(x)$ as all possible polynomials with degree $< (n-k)$. For every $s(x)$ we chose an error vector $e(x)$ with smallest hamming weight and decode as $v(x) = w(x) - e(x)$. For the Hamming code K[7,4,3] with generator polynomial $g(x) = 1+x+x^3$ we construct the syndromes to the error vectors x^i $0 \le i \le 6$ and get

e(x)	s(x)
1	1
x	x
x^2	x^2
x^3	$x+1$
x^4	x^2+x
x^5	x^2+x+1
x^6	x^2+1

Observe that we have got all the polynomials $s(x)$ with degree $s(x) < (n-k) = 3$ due to the fact that the Hamming codes are perfect single-error-correcting.

We use MATHEMATICA to describe the methods of decoding. We consider (as in the example above) the Hamming code K[7,4,3] , were we recieved the word $w(x) = 1 + x^3 + \mathbf{x^4} + x^5 + x^6$.
The following steps illustrate how the information word could be reconstructed:

In[23] : =
```
V=1+x^3+x^5+x^6;
e=x^4;
W=V+e;
s=PolynomialMod[PolynomialMod[W,g],2]
```
Out[23] =
$x + x^2$

We have got the syndrome polynomial and from the syndrome list above we recognize the error polynomial $e(x) = x^4$

An alternative way to decode the recieved word $w(x) = 1 + x^3 + \mathbf{x^4} + x^5 + x^6$ is given in next steps:

In[24] : =
w=CoefficientList[W,x]
Out[24] =
{1, 0, 0, 1, 1, 1, 1}

In[25] : =
H=CyclicCode[g,7,x][[1]];
MatrixForm[H]
Out[25] =

$$\begin{matrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{matrix}$$

In[26] : =
MatrixForm[Syndrome[H,w]]
Out[26] =

$$\begin{matrix} 1 \\ 1 \\ 1 \end{matrix}$$

We see that this column is identical with the 5:th column in H , indicating that we have an error in the 5:th position.

Meggitt - decoding

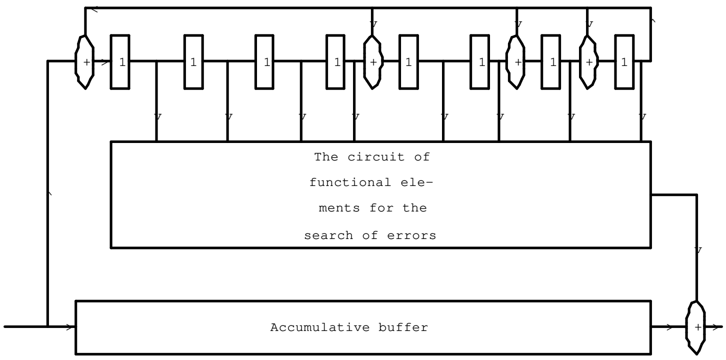
We consider now the code K[15,7,5] with the generator polynomial $g(x) = 1+x^4+x^6+x^7+x^8$. This is a cyclic BCH-code, which corrects 2 errors. The syndrome list consists of all polynomials s(x) with deg s(x) < 8 (the degree of f(x)) . This means, we have 2^8=256 syndromes, and the error vector has hamming weight 0,1 or 2.
The decoding would start with the construction of the syndrome list and the corresponding error vectors. From this list we select the error vector corresponding to the syndrome of the recieved word w(x).
A substantial simplification is provided by the Method of Meggitt, which uses the cyclic character of the code:
w(x) has the syndrome s(x) $\Rightarrow$ $w^{(1)}(x) = x\, w(x)$ has the syndrome $s^{(1)}(x)$ (= the remainder in x s(x)/g(x)).
Using this theorem, we would only compare the syndrome of w(x), recieved after n shifts in the divide-by-g(x)-circuit, with the syndromes having a highest exponent, i.e. $x^{(n-k-1)}$, corresponding to the rightmost bit . If these are equal, we correct the highest bit of the word w(x). If not equal, we perform a cyclical shift of w(x) and compare again. After (n-1) shifts we would have corrected all bits and a final shift gives us the code word v(x).

We illustrate the method of Meggitt decoding in MATHEMATICA with the code K[15,7,5] .

In[27] : =
g=1+x^4+x^6+x^7+x^8;
ShowMeggittDecoder[g,15,x]

(* As usual we choose g(x) after recieving all possible generator polynomials by the command generatorPolynomials[15,x] *)

1 - the shift register



The syndrome list encompasses all combinations with x^{14} of length 2 (the code corrects two errors)

Error Polynomial	Syndrome Polynomial	Binary Vector
x^{14}	$x^7 + x^6 + x^5 + x^3$	00010111
$1 + x^{14}$	$x^7 + x^6 + x^5 + x^3 + 1$	10010111
$x + x^{14}$	$x^7 + x^6 + x^5 + x^3 + x$	01010111
. . .	. . .	. . .
$x^{13} + x^{14}$	$x^7 + x^4 + x^3 + x^2$	00111001

In[28] :=
g=1+x^4+x^6+x^7+x^8;
v={1,0,0,0,1,0,1,1,1,0,0,0,0,0,0};
e={0,0,0,0,1,0,0,0,0,0,0,0,1,0,0};
w=Mod[v+e,2];

(* We let the error vector indicate two errors, in the 5.th and in the 13.th position *)

MeggittDecode[g,w,15,x] (* This command shows the decoding process *)

The code distance = 5.

THE SYNDROME LIST = 0 0 0 1 0 1 1 1

1 0 0 1 0 1 1 1

0 1 0 1 0 1 1 1

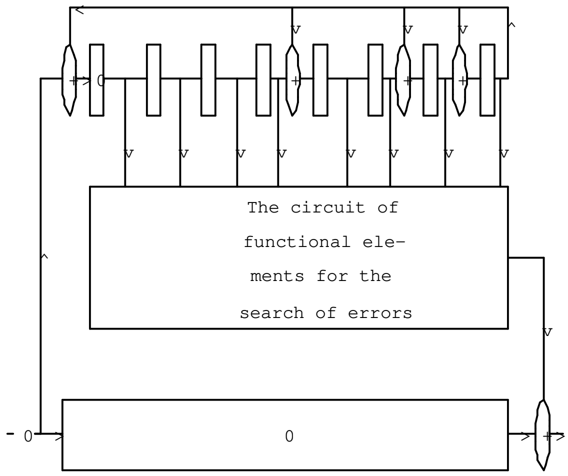
. . . . . .

. . . . . .

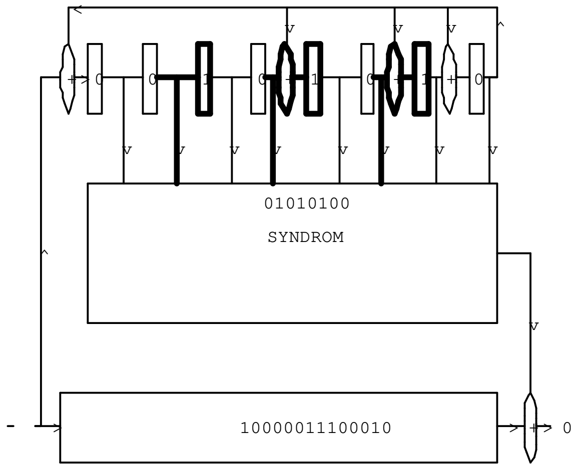
0 1 0 0 1 0 1 1

0 0 1 1 1 0 0 1

1.1

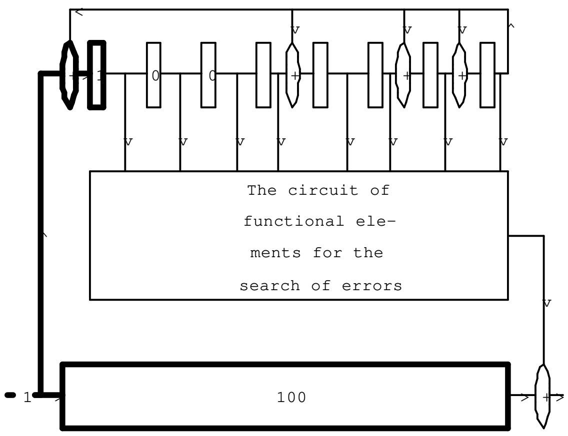


2.1

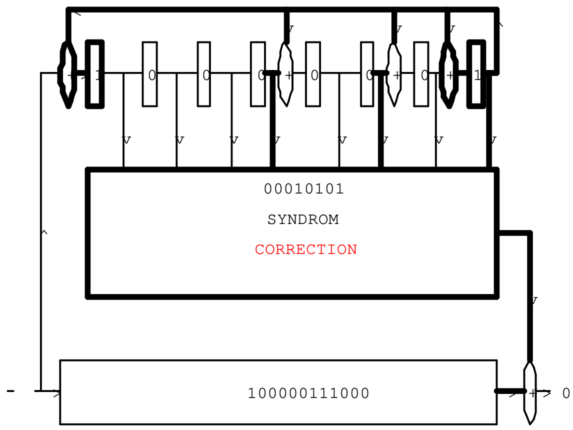


. . . . . .

1.3

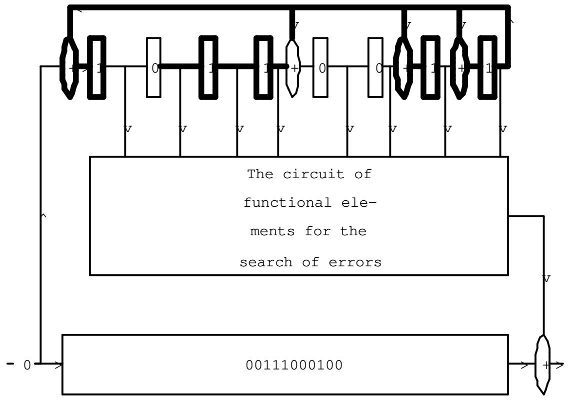


2.3

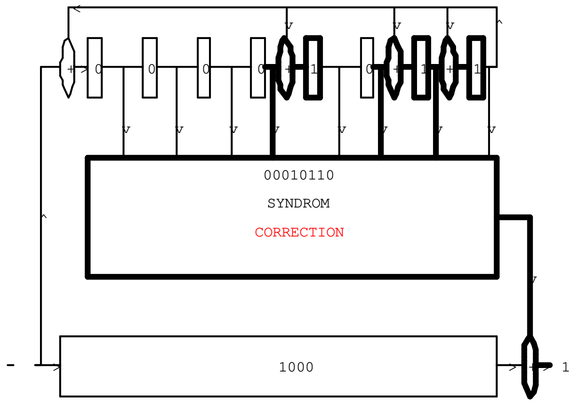


. . . . . .

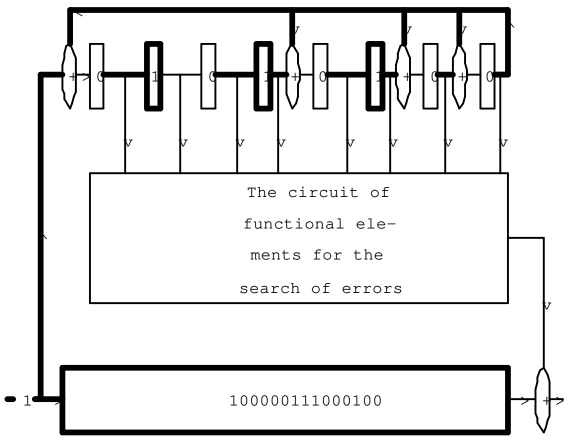
1.11



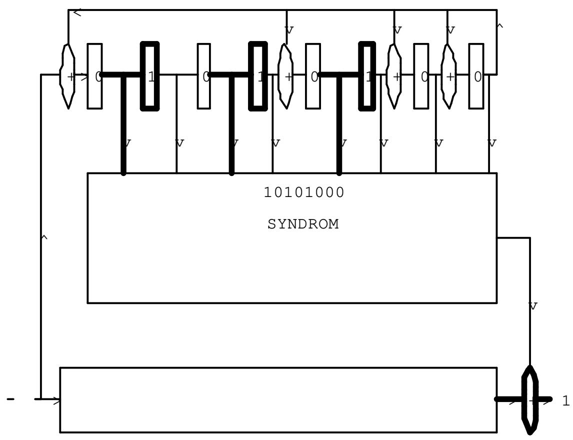
2.11



1.15



2.15



The received word = 100000111000100
The decoded word = 100010111000000

Some weight distributions are asymptotically normal

Let K[n,k,d] be a binary code. We construct the double - error correcting BCH code of length 31 with weight distribution (A_0 , A_1 , ... , A_n) where A_i is the number of codewords of hamming weight i.

In[29] :=
n=31
Out[29] =
31

In[30] :=
gp=GeneratorPolynomials[n,x];
Short[gp]
Out[30] =
{1, 1 + x, 1 + x^2 + x^5 , <<123>>, 1 + <<30>>, 1 + x^{31} }

In[31] :=
g=gp[[26]]
Out[31] =
$1 + x^3 + x^5 + x^6 + x^8 + x^9 + x^{10}$ (* The generator polynomial g(x) generates the cyclic code of dimension 21 . *)

In[32] :=
H=CyclicCode[g,n,x][[1]];
n=Length[H[[1]]]
k=DimensionCode[H]
Out[32] =
31
Out[33] =
21

<u>*Definition:*</u> If K is a block code with exactly A_i code words of weight i, the polynomial

$A(x) = \sum_{i=0}^{n} A_i x^i$ is called the weight polynomial.

If we want to find the weight polynomial of our code we need to check all 2^{21} = 2097152 binary vectors of length 31.We get a substantial simplification if we use the *dual code* : The dual code $K^{\perp}$ consists of the words ortogonal to every word in K. We could then use the Mac Williams Identity:

$A_{K^{\perp}}(x) = \frac{(1+x)^n}{2^k} A_K\left(\frac{1-x}{1+x}\right)$ where $A_{K^{\perp}}(x)$ is the weight polynomial of the dual code.

We therefore first construct the weight polynomial of the dual code . The dual code consists of 2^{31-21} =1024 code words, making the construction of the corresponding weight polynomial fairly easy.

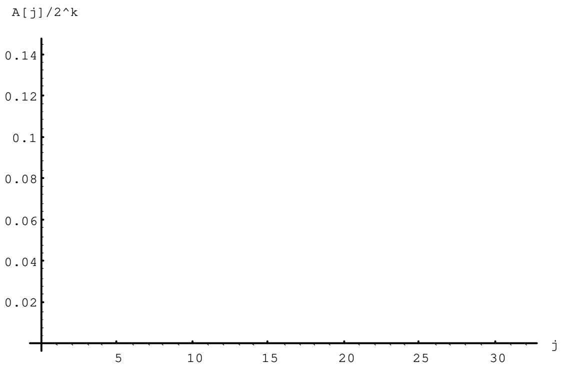
In[34] :=
wpdc=WeightPolynomial[DualCode[H],x]
Out[34] =
$1 + 310 x^{12} + 527 x^{16} + 186 x^{20}$

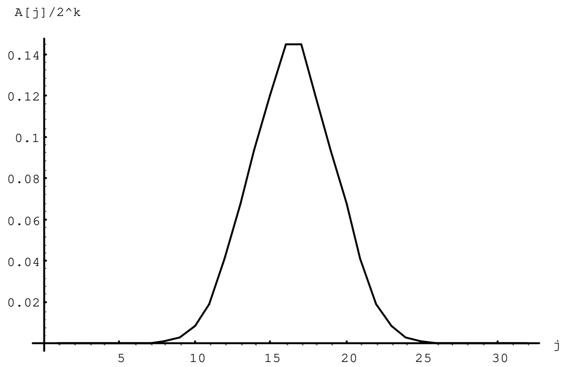
In[35] :=
WP=McWilliamsIdentity[wpdc,n,x] (* Weight polynomial of the BCH code of length 31*)
Out[35] =
$1 + 186 x^5 + 806 x^6 + 2635 x^7 + 7905 x^8 + 18910 x^9 + 41602 x^{10} + 85560 x^{11} + 142600 x^{12} + 195300 x^{13} + 251100 x^{14} + 301971 x^{15} + 301971 x^{16} + 251100 x^{17} + 195300 x^{18} + 142600 x^{19} + 85560 x^{20} + 41602 x^{21} + 18910 x^{22} + 7905 x^{23} + 2635 x^{24} + 806 x^{25} + 186 x^{26} + x^{31}$

In[36] :=
WPCoeff=CoefficientList[WP,x]   (* Weight distribution of the code *)
Out[36] =
{1, 0, 0, 0, 0, 186, 806, 2635, 7905, 18910, 41602, 85560, 142600, 195300, 251100, 301971, 301971, 251100, 195300, 142600, 85560, 41602, 18910, 7905, 2635, 806, 186, 0, 0, 0, 0, 1}

Let $a = (a_0, a_1, \ldots a_n)$ where $a_j = A_j / 2^k$ and $\sum a_j = 1$.

In[37] :=
ListPlot[WPCoeff/2^k,AxesLabel->{"j","A[j]/2^k"}]   (* The numbers a_0 , a_1 , ... a_n are plotted
lp1=ListPlot[WPCoeff/2^k,   by the command ListPlot *)
PlotJoined->True,AxesLabel->{"j","A[j]/2^k"}]





We recognize a curve similar to the (gaussian) normal distribution curve.

From MATHEMATICA we now load the package Statistic, Normal distribution and calculate the mean value m and the dispersion dsp.

In[38] :=
Normaldi.m

In[39] :=
m=N[Sum[(WPCoeff[[i]]*i)/(2^k),{i,1,n}]]
Out[39] =
16.5

In[40] :=
dsp=Sqrt[Sum[((m-i)^2)*(WPCoeff[[i]]/2^k),{i,1,n}]]
Out[40] =
2.78386

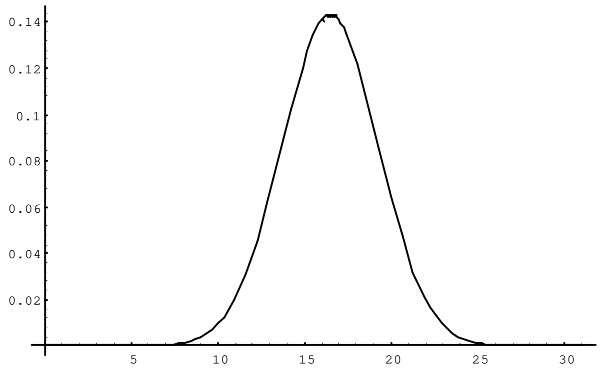
In[41] :=
ndb=NormalDistribution[m,dsp]
Out[41] =
NormalDistribution[16.5, 2.78386]

In[42] :=
nf=PDF[ndb,x]
Out[42] =

$$\frac{0.359213}{E^{0.0645171\,(-16.5 + x^2)}\ \mathrm{Sqrt}[2\ \mathrm{Pi}]}$$

(* This is the normal distribution function*)

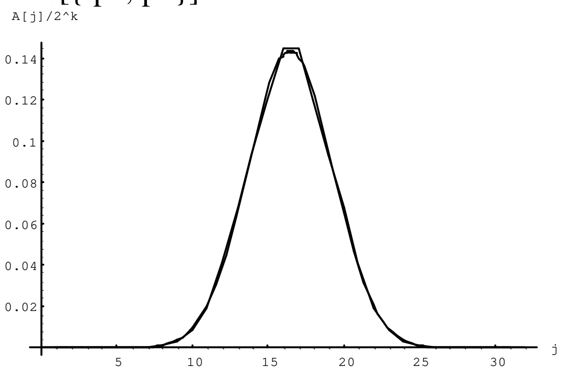
In[43] :=
lp2=Plot[nf,{x,0,n}]



(*This is the normal distribution *)

We finally we show the weight distribution and normal distribution in the same coordinate system.

In[41] :=
Show[{lp1,lp2}]



The picture confirms the THEOREM: Let K be an binary code , and let $d' > 3$ be the minimal distance of the dual code K. Then the weight distribution of the code K is asymptotically normal.