Computation of weight enumerators of binary linear codes using the package “Coding Theory”

Igor Gachkov
Karlstad University, Department of Engineering Sciences, Physics and Mathematics 65188 Karlstad Sweden

Igor.Gachkov@kau.se

1. Abstract
Using the package "Coding Theory in Mathematica" illustrates Sidelnikovs theorem: “Some weight distributions of linear codes are asymptotically normal”. We give a simple description of the Wagner code with parameters (23,14,5), and prove that the code is quasi-perfect. We use Mathematica to obtain, using linear programming, a weight distribution of the Wagner code. This method of calculating the weight polynomial allows us to prove the optimum of the Wagner code among the family of codes correcting two errors with nine checking symbols.

2. Introduction

Information coding is of interest to engineers working with data transmission and related fields. Coding Theory relies heavily on abstract algebra and number theory, and is therefore usually studied for the first time by postgraduate students.

In this paper we will give an illustration of Sidelnikovs theorem: "Some weight distributions of linear codes are asymptotically normal ". We will also give a simple description of the Wagner code [Wagner 1966] with parameters (23,14,5) and prove that the code is quasi-perfect [Gachkov 1999]. The calculations are based on the discussion "Coding Theory in Mathematica" [Gachkov 2003]. The Mathematica package "Coding Theory" was developed for a course " Error-Correcting codes with Mathematica ", for students at advanced undergraduate level. We will show here how we can use this package for research in the area of Error-Correcting codes.
2.1 The basics of Coding Theory

A binary block code $C$ of length $n$ is any non-empty subset of $\mathbb{Z}^n$. A linear code $C$ of length $n$ is any linear subspace of $\mathbb{Z}^n$. We represent the code dimension of $C$ as $k = \dim C$, where $\dim C$ is the dimension of the linear subspace. The elements of $C$ are called code words. We define also the Hamming distance $d(v, w)$ between two code words $v$ and $w$ as the number of coordinates where $v$ and $w$ differ, the weight $w(v)$ of code word $v$ is $d(v, 0)$, where $0$ is $(0, 0, \ldots, 0)$ the null vector. The code distance of code $C$ is the minimum Hamming distance between all possible pairs of code words and can be given by $d = \text{Min}\{d(v,w); v,w \in C\}$. The error correcting capability is calculated by $t = \text{Floor}\{(d-1)/2\}$. For instance, a code with code distance 5 can correct two errors.

As early as 1966, Terry J. Wagner [Wagner 1966] suggested an algorithm for finding quasi-perfect codes, which was implemented on a computer. This linear binary code is known today as "Wagner code". The results were summarized in a table of quasi-perfect double error-correcting codes. The distinction between perfect and quasi-perfect codes lies in the following: a perfect-errors-correcting code can correct all errors of weight $t$ or less, and no errors of weight greater than $t$, whereas a quasi-perfect $t$-errors-correcting code can correct all errors of weight $t$ or less, some errors of weight $t+1$, and no errors of weight greater than $t+1$. A quasi-perfect $t$ errors-correcting code $C$ with parameters $[n, k, d=2t+1]$ (length, dimension, code distance) has a parity-check matrix $H$ ($n$ by $(n-k)$ matrix) for which the following are true:

1. Every $2t$ distinct columns from $H$ are linearly independent.
2. The set of all linear combinations of $t+1$ (or fewer) columns from $H$ is equal to the set of all possible columns.

Perfect and quasi-perfect codes are of great interest because they are the optimal choice for the binary symmetric communication channel. Successful application in computers is proved by a list of codes obtained in 1974 by A. A. Hashim and A.G.Constantinides. The results of code searching used to be put in tables of optimal codes. The first lower bound appeared in 1977 in the book of MacWilliams and Sloane ([Sloane 1977], pp.684-691). A table of binary constant weight codes of length $n \leq 28$ with explicit constructions for most of the 600 codes was presented in the encyclopedic work of Brouwer, Shearer, Sloane, and Smith [BSS 1990]. Today Neil J. A. Sloane presents his table of constant weight codes [Sloane 2004] online and performs continual updates.

However, no opportunity for checking or correcting the results existed, due to a lack of appropriate software. Great developments and improvements of computational techniques in recent years have increased opportunities for finding new codes and correcting results. Some time later, the author used Mathematica in the theory of error-correcting codes. This resulted in a simple description of already known codes, with illustrations of some new theorems, based on the package described in "Coding Theory in Mathematica" [Gachkov 2003].
3. Some weight distributions are asymptotically normal

3.1 Definitions

Let $K[n, k, d]$ be a binary code (the parameters are length, dimension and code distance). We construct the double - error correcting BCH code of length 31 with weight distribution $(A_0, A_1, \ldots, A_n)$ where $A_i$ is the number of code words of hamming weight $i$.

**Definition:**

If $K$ is a block code with exactly $A[i]$ code words of weight $i$, the polynomial $A_K(x) = \sum_{i=0}^{n} A_i * x^i$

is called the weight polynomial.

If we want to find the weight polynomial of our code we need to check all $2^{31} = 2097152$ binary vectors of length 31.

We get a substantial simplification if we use the dual code: the dual code $K_1$ consists of the words which are orthogonal to every word in $K$. We could then use the MacWilliams Identity:

$$A_{K_1}(x) = \frac{(1+x)^n}{2^k} * A_K\left(\frac{1-x}{1+x}\right),$$

where $A_{K_1}(x)$ 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.

3.2 Implementation

We begin with:
The generator polynomial \( g(x) \) generates the cyclic code of dimension 21. Now we calculate the parameters of cyclic code \( K \).

\[
H = \text{CyclicCode}[g, n, x][[1]]
\]

\[
n = \text{Length}[H[[1]]]
\]

\[
k = \text{DimensionCode}[H]
\]

\[
\begin{array}{cccccccccccccccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

31

21

First of all we calculate weight enumerator of dual code \( K \)

\[
\text{wpdc} = \text{WeightPolynomial}[	ext{DualCode}[H], x]
\]

\[
186 \, x^{20} + 527 \, x^{16} + 310 \, x^{12} + 1
\]
and using MacWilliams identity we can calculate the Weight polynomial of the BCH code of length 31 and the Weight distribution of the code.

\[
WP = \text{McWilliamsIdentity}[wpdc, n, x]
\]

(* Weight polynomial of the BCH code of length 31*)

\[
x^{31} + 186 x^{26} + 806 x^{25} + 2635 x^{24} + 7905 x^{23} + 18910 x^{22} + 41602 x^{21} + 85560 x^{20} + 142600 x^{19} + 195300 x^{18} + 251100 x^{17} + 301971 x^{16} + 301971 x^{15} + 251100 x^{14} + 195300 x^{13} + 142600 x^{12} + 85560 x^{11} + 41602 x^{10} + 18910 x^{9} + 7905 x^{8} + 2635 x^{7} + 806 x^{6} + 186 x^{5} + 1
\]

\[
WP\text{Coeff} = \text{CoefficientList}[WP, x]
\]

(* Weight distribution of the code *)

\[
\{1, 0, 0, 0, 0, 186, 806, 2635, 7905, 18910, 41602, 85560, 142600, 195300, 251100, 301971, 301971, 251100, 195300, 142600, 85560, 41602, 18910, 7905, 2635, 806, 186, 0, 0, 0, 0, 1\}
\]

### 3.3 Statistical Analysis

Let \( a = (a_0, a_1, \ldots, a_n) \) where \( a_j = A_j / 2^k \) and \( \sum_j a_j = 1 \).
ListPlot[WPCoeff/2^k, AxesLabel -> 
{"j", "A[j]/2^k"}]
(* The numbers a0, a1, ..., an are plotted by the command ListPlot *)
lp1 = ListPlot[WPCoeff/2^k,
PlotJoined -> True, AxesLabel -> 
{"j", "A[j]/2^k"}]
Null

We recognize a curve similar to the (Gaussian) normal distribution curve.
From Mathematica we now load the package Statistics'Normal distribution' and calculate the mean value m and the dispersion dsp.

<< Statistics`NormalDistribution`
Finally we show the weight distribution of the code K and normal distribution with parameters \( m=16.5 \) and \( \text{dsp}=2.78386 \) in the same coordinate system.
3.4 Theorem

The graphics above illustrate the following theorem.

**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.

4. The Minimal property of some binary codes

Let's take a look at a class of binary codes, which contains all-ones vectors, with parameters $[n, n-k, d]$ where $n$ is a length of code, $k$ is a number of control bits and $d$ is the code distance. In this case weight polynomial $wt(x)$ of the code is a symmetric polynomial and has the following structure

$$1 + a_1 x^d + a_2 x^{d+1} + a_3 x^{d+2} + ... + a_{d} x^{n-d-2} + a_{d+1} x^{n-d-1} + a_{d+2} x^{n-d} + x^n$$

One of the classical examples is BCH code with parameters $[15,7,5]$. 
4.1 Code Generation

The package "Coding Theory" [Gachkov 2003] allows us to give a full description of the cyclic BCH code with length 15 which correcting 2 errors. We choose \( r(x) \) as follows for the generator polynomial.

\[
r = \text{GeneratorPolynomials}[15, x][[19]]
\]

\[
x^8 + x^7 + x^6 + x^4 + 1
\]

The generator and parity check matrix can be obtained by \texttt{ShowCyclicCode}.

\[
\text{ShowCyclicCode}[r, 15, x]
\]

\[
\begin{aligned}
\text{GENERATOR POLYNOMIAL} & \quad g = x^8 + x^7 + x^6 + x^4 + 1 \\
\text{GENERATOR MATRIX GM} & = \\
& \begin{pmatrix}
000101111000000 \\
010001011100000 \\
001000101110000 \\
000100010111000 \\
000010001011100 \\
000001000101110 \\
000000100010111 \\
\end{pmatrix}
\end{aligned}
\]

\[
\text{PARITY CHECK POLYNOMIAL} & \quad h = x^7 + x^6 + x^4 + 1 \\
\text{PARITY CHECK MATRIX HM} & = \\
& \begin{pmatrix}
000000011010001 \\
000000110100010 \\
000001101000100 \\
000011010001000 \\
001101000100000 \\
011010001000000 \\
110100010000000 \\
\end{pmatrix}
\end{aligned}
\]

In the last step the weight enumerators \( \text{wt}(x) \) for BCH 15 code and for dual code are calculated.
The fact that wt(x) is symmetric proves that BCH 15 code belongs to the class of interest.

### 4.2 Minimal Property: proof

We make the following conjecture for BCH 15 code.

**BCH 15 code contains the minimal possible number of code words with minimal hamming weight (wt = 5,6)**

The proof is based on the following *Mathematica* calculations. Let's form a weight enumerator for code K with undefined coefficients.

\[
\begin{align*}
\text{n} &= 15; \text{k} = 8; \\
\text{f} &= 1 + \text{Sum}\left[\text{If}[i < n/2, a[i-4] \times x \times i, a[-4+n-i] \times x \times i], \{i, 5, n-5\}] \times x \times n \right] \\
&= x^{15} + a(1) \times x^{10} + a(2) \times x^{9} + a(3) \times x^{8} + a(3) \times x^{7} + a(2) \times x^{6} + a(1) \times x^{5} + 1
\end{align*}
\]

By using the MacWilliams identity we get the weight distribution of the dual code

\[
\text{WL} = \text{Simplify}[\text{CoefficientList][} \text{Expand}[\text{McWilliamsIdentity}[\text{f}, n, x]], x] / . \\
\text{Sum}[a[i], \{i, 1, \text{Floor}[(n/2-4)], 0, 1] \times 2^\times((n-k-1)) - 1]
\]

\[
\begin{align*}
&\left\{ \frac{1}{64} \times (a(1) + a(2) + a(3) + 1), 0, \frac{1}{64} \times (5 a(1) - 3 a(2) - 7 a(3) + 105), \\
&0, \frac{1}{64} \times (-35 a(1) - 3 a(2) - 7 a(3) + 65)), \\
&0, \frac{5}{64} \times (13 a(1) + 5 a(2) - 7 a(3) + 1001), \\
&0, \frac{5}{64} \times (9 a(1) + 9 a(2) - 7 a(3) - 1287), \\
&0, \frac{1}{64} \times (-a(1) + 39 a(2) - 21 a(3) + 3003), \\
&0, \frac{1}{64} \times (15 a(1) - 17 a(2) + 7 a(3) + 65)), \\
&0, \frac{1}{64} \times (-5 a(1) + 3 a(2) - a(3) + 15) \right\}
\end{align*}
\]
The fact that the coordinates in the weight distribution are non-negative, together with \(a(1)+a(2)+a(3)+1 = 64\) allows us to formulate a linear programming problem to determine the upper and lower bounds for \(a(i)\).

```
Do[aa =
  ConstrainedMax[a[1], Join[Table[WL[[2 i + 1]] >= 0, {i, 1, Floor[(n/2)]}],
    Table[a[i] <= 2^(n-k) - 2, {i, 1, Floor[(n/2 - 4)]]},
    {Sum[a[i], {i, 1, Floor[(n/2 - 4)]]} ==
      2^((n-k) - 1) - 1}],
  Table[a[i], {i, 1, Floor[(n/2 - 4)]]]);
bb = ConstrainedMin[a[1],
  Join[Table[WL[[2 i + 1]] >= 0, {i, 1, Floor[(n/2)]}],
    Table[a[i] <= 2^(n-k) - 2, {i, 1, Floor[(n/2 - 4)]]},
    {Sum[a[i], {i, 1, Floor[(n/2 - 4)]]} ==
      2^((n-k) - 1) - 1}],
  Table[a[i], {i, 1, Floor[(n/2 - 4)]]]];
Print[bb[[2, 1]] // N, " ", aa[[2, 1]] // N],
{1, 1, n/2 - 4}]
```

\[
a(1) \rightarrow 14. \quad a(1) \rightarrow 25.5
\]

\[
a(2) \rightarrow 30. \quad a(2) \rightarrow 43.75
\]

\[
a(3) \rightarrow 0. \quad a(3) \rightarrow 15.
\]

Now we test the linear programming problem for minimising \(a(1)+a(2)\).

```
bb = ConstrainedMin[a[1] + a[2],
  Join[Table[WL[[2 i + 1]] >= 0, {i, 1, Floor[(n/2)]}],
    Table[a[i] <= 2^(n-k) - 2, {i, 1, Floor[(n/2 - 4)]]},
    {Sum[a[i], {i, 1, Floor[(n/2 - 4)]]} ==
      2^((n-k) - 1) - 1}],
  Table[a[i], {i, 1, Floor[(n/2 - 4)]]]]]
```

\[
\{48, \{a(1) \rightarrow 18, a(2) \rightarrow 30, a(3) \rightarrow 15\}\}
\]

We get the only possible solution, which gives the BCH 15 code parameters.

## 5. Wagner code

By using this method we can find whether or not the Wagner code belongs to the same class as described above, and if this code has a minimal property. A simple (computer) description of the Wagner code is presented by the author in [Gachkov 1999]. In particular the parity check matrix and
weight polynomial of the Wagner code were described. We get the parity check matrix of the Wagner code as follows

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

The weight enumerator is:

\[
x^{23} + 84x^{18} + 252x^{17} + 445x^{16} + 890x^{15} + 1620x^{14} + 2268x^{13} + 2632x^{12} + 2632x^{11} + 2268x^{10} + 1620x^9 + 890x^8 + 445x^7 + 252x^6 + 84x^5 + 1
\]

\[
x^{23} + 84x^{18} + 252x^{17} + 445x^{16} + 890x^{15} + 1620x^{14} + 2268x^{13} + 2632x^{12} + 2632x^{11} + 2268x^{10} + 1620x^9 + 890x^8 + 445x^7 + 252x^6 + 84x^5 + 1
\]

Once again the symmetry of \( wt(x) \) demonstrate that the Wagner code belongs to our class.

### 5.1 Minimal property for Wagner code

We make the following conjecture for Wagner code

The Wagner code has the minimal possible number of code words of minimal weight.

Again, we prove this using Mathematica. By analogy with the previous proof we can build a weight enumerator \( wt(x) \) of degree 23 (code distance is 5)
n = 23; k = 9;
f = 1 + \sum[\text{If}[i < n/2, a[i-4] \times x^i, a[-4+n-i] \times x^i],
   \{i, 5, n-5\}] + x^n

x^{23} + a(1) x^{18} + a(2) x^{17} + a(3) x^{16} + a(4) x^{15} +
a(5) x^{14} + a(6) x^{13} + a(7) x^{12} + a(7) x^{11} + a(6) x^{10} +
a(5) x^{9} + a(4) x^{8} + a(3) x^{7} + a(2) x^{6} + a(1) x^{5} + 1

The weight distribution of the dual code is

\text{Short}[\text{WL} = \text{Simplify}[
   \text{CoefficientList}[\text{Expand}[\text{McWilliamsIdentity}[f, n, x]], x] /. \n   \text{Sum}[a[i], \{i, 1, \text{Floor}[(n/2 - 4)]\}] \rightarrow 2^\left((n - k - 1) - 1\right), 3]

\left\{ \frac{a(1) + a(2) + a(3) + a(4) + a(5) + a(6) + a(7) + 1}{8192}, 0,
   \frac{73 a(1) + 49 a(2) + 29 a(3) + 13 a(4) + a(5) - 7 a(6) - 11 a(7) + 253}{8192},
   \frac{\text{\langle\text{18}\rangle}, 0,}{8192},
   \frac{-13 a(1) + 11 a(2) - 9 a(3) + 7 a(4) - 5 a(5) + 3 a(6) - a(7) + 23}{8192}\right\}

The coordinates of the weight distribution are non-negative, so we can formulate a linear programming problem
Do[aa = ConstrainedMax[a[1],
   Join[Table[WL[[2 i + 1]] >= 0, {i, 1, Floor[(n/2)]]],
   Table[a[i] <= 2^(n-k) - 2, {i, 1, Floor[(n/2 - 4)]]],
   {Sum[a[i], {i, 1, Floor[(n/2 - 4)]]} ==
    2^((n-k) - 1) - 1],
   Table[a[i], {i, 1, Floor[(n/2 - 4)]]}]];
bb = ConstrainedMin[a[1],
   Join[Table[WL[[2 i + 1]] >= 0, {i, 1, Floor[(n/2)]]],
   Table[a[i] <= 2^(n-k) - 2, {i, 1, Floor[(n/2 - 4)]]],
   {Sum[a[i], {i, 1, Floor[(n/2 - 4)]]} ==
    2^((n-k) - 1) - 1],
   Table[a[i], {i, 1, Floor[(n/2 - 4)]]}]];
Print[bb[[2, 1]] // N, " ", aa[[2, 1]] // N],
{1, 1, n/2 - 4}]

\[
a(1) \rightarrow 46.6667 \quad a(1) \rightarrow 121.333 \\
a(2) \rightarrow 252. \quad a(2) \rightarrow 413.778 \\
a(3) \rightarrow 242.333 \quad a(3) \rightarrow 474.6 \\
a(4) \rightarrow 612.311 \quad a(4) \rightarrow 914.889 \\
a(5) \rightarrow 1486.67 \quad a(5) \rightarrow 1956. \\
a(6) \rightarrow 2268. \quad a(6) \rightarrow 2902.67 \\
a(7) \rightarrow 2184. \quad a(7) \rightarrow 2632.
\]

That gives us exact bounds of the variables a (1), a (2)…a (7).
We now minimise a(1) + a(2).

ConstrainedMin[a[1] + a[2],
   Join[Table[WL[[2 i + 1]] >= 0, {i, 1, Floor[(n/2)]]],
   Table[a[i] <= 2^(n-k) - 2, {i, 1, Floor[(n/2 - 4)]]],
   {Sum[a[i], {i, 1, Floor[(n/2 - 4)]]} ==
    2^((n-k) - 1) - 1],
   Table[a[i], {i, 1, Floor[(n/2 - 4)]]}]]]

\{
\{336, \{a(1) \rightarrow 84, a(2) \rightarrow 252, a(3) \rightarrow 445, \\
a(4) \rightarrow 890, a(5) \rightarrow 1620, a(6) \rightarrow 2268, a(7) \rightarrow 2632\} \}

This gives a wt(x) of the Wagner code.

6. Conclusion
Today it is hardly possible to imagine engineers working with data transmission and related fields who do not have a basic knowledge of information coding/decoding. In past years courses in Coding Theory have been restricted to postgraduate students. This is due to the complexity of the mathematical methods used in most codes, such as various results from abstract algebra and number theory. With the introduction of computers and computer algebra it is possible to offer method-oriented courses, where the learning process is supported by a rich variety of methods for manipulation of algebraical structures, and the simplicity of being able to vary parameters in specific constructs. Methods can be fairly well illustrated with applications, and thus strengthen the understanding of the mathematical ideas used. In this paper we have outlined how, with help of our Mathematica package, we recently found a solution of a research problem. This result indicates how computer algebra can be used to create new knowledge, hardly achievable without computers. By using our package we have been able to give a simple description of the Wagner code with parameters [23,14,5], and to calculate some properties for binary codes.

7. References


