

31 Chapter: Introduction to Algebraic Coding Theory

The chapter assumes familiarity with the notation and material in Chapter 31 of [Gallian].

A vector in **GAP** is entered by listing the components within square brackets. For example the vector $v = (1, 3, 7)$ is entered as:

```
gap> v:= [1,3,7];
[ 1, 3, 7 ]
```

A matrix can be entered as a list of row vectors. For example, the matrix

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

is entered into **GAP** by typing

```
gap> M:= [ [1,2,3], [4,5,6], [7,8,9] ];
[ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ]
```

If you prefer to exhibit the matrix M as a 3 by 3 array use the command **PrintArray**:

```
gap> PrintArray(M);
[ [ 1, 2, 3 ],
  [ 4, 5, 6 ],
  [ 7, 8, 9 ] ]
```

The notation $M[i][j]$ denotes the entry in the i th row and j th column of M . Similarly, the notation $M[i]$ denotes the i th row of M .

```
gap> M[2][3];
6
gap> M[1];
[ 1, 2, 3 ]
```

To multiply v and M type:

```
gap> v*M;
[ 62, 73, 84 ]
gap> M*v;
[ 28, 61, 94 ]
```

That is, $vM = (62, 73, 84)$ and $Mv = (28, 61, 94)$. Notice that **GAP** automatically treats v as a row vector in the multiplication vM but as a column vector in the multiplication Mv .

Read Example 9 of [Gallian, Chapter 31]. The software **GAP** easily performs the computations needed to decode received codes. First set up the parity check matrix given in [Gallian, Chapter 31, Example 9]:

```

gap> Elements(Integers mod 2);
[ 0*Z(2), Z(2)^0 ]
gap> z:= 0*Z(2);;
gap> a:= Z(2)^0;;
gap> H:= [ [a,a,z],
> [a, z, a],
> [a, a, a],
> [z, a, a],
> [a, z, z],
> [z, a, z],
> [z, z, a]];;
gap> PrintArray(H);
[ [ Z(2)^0, Z(2)^0, 0*Z(2) ],
  [ Z(2)^0, 0*Z(2), Z(2)^0 ],
  [ Z(2)^0, Z(2)^0, Z(2)^0 ],
  [ 0*Z(2), Z(2)^0, Z(2)^0 ],
  [ Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0 ] ]

```

Next enter in the received the word $v = 0000110$, compute vH and note vH is the first row of H :

```

gap> v:=[z,z,z,z,a,a,z];
[ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0, Z(2)^0, 0*Z(2) ]
gap> PrintArray(v*H);
[ Z(2)^0, Z(2)^0, 0*Z(2) ]
gap> v*H = H[1];
true

```

Similarly, if the received word is $w = 1011111$, we can use GAP to compute wH :

```

gap> w:=[a,z,a,a,a,a,a];
[ Z(2)^0, 0*Z(2), Z(2)^0, Z(2)^0, Z(2)^0, Z(2)^0, Z(2)^0 ]
gap> PrintArray(w*H);
[ Z(2)^0, 0*Z(2), Z(2)^0 ]
gap> w*H = H[2];
true

```

Thus, since vH equals the first row of H we decode $v = (0, 0, 0, 0, 1, 1, 0)$ as $(1, 0, 0, 0, 1, 1, 0)$. Since wH equals the second row of H we decode $w = (1, 0, 1, 1, 1, 1, 1)$ as $(1, 1, 1, 1, 1, 1, 1)$.

Read Example 11 of [Gallian, Chapter 31]. We will now use GAP to do syndrome decoding. First enter in the parity check matrix in [Gallian, Chapter 31, Example 11]. Here \mathbf{a} and \mathbf{z} are defined as in the example above:

```

gap> H:= [[a,a,z],
> [a,z,a],

```

```

> [z,a,a],
> [a,z,z],
> [z,a,z],
> [z,z,a]]];
gap> PrintArray(H);
[ [ Z(2)^0, Z(2)^0, 0*Z(2) ],
  [ Z(2)^0, 0*Z(2), Z(2)^0 ],
  [ 0*Z(2), Z(2)^0, Z(2)^0 ],
  [ Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0 ] ]

```

Now set up the matrix CL whose rows are the coset leaders:

```

gap> CL:= [[z,z,z,z,z,z],
> [a,z,z,z,z,z],
> [z,a,z,z,z,z],
> [z,z,a,z,z,z],
> [z,z,z,a,z,z],
> [z,z,z,z,a,z],
> [z,z,z,z,z,a],
> [a,z,z,z,z,a]]];
gap> PrintArray(CL);
[ [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ],
  [ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ] ]

```

The matrix CL*H will be the matrix whose rows are the syndromes.

```

gap> Synd:=CL*H;
gap> PrintArray(Synd);
[ [ 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ Z(2)^0, Z(2)^0, 0*Z(2) ],
  [ Z(2)^0, 0*Z(2), Z(2)^0 ],
  [ 0*Z(2), Z(2)^0, Z(2)^0 ],
  [ Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0 ],
  [ Z(2)^0, Z(2)^0, Z(2)^0 ] ]

```

To decode the word $v = 101001$ compute vH . Since vH equals the 5th row of **Synd** we decode v as v minus the 5th row of **CL**:

```

gap> v:=[a,z,a,z,z,a];;
gap> PrintArray(v*H);
[ Z(2)^0, 0*Z(2), 0*Z(2) ]
gap> v*H = Synd[5];
true
gap> decodev:= v - CL[5];;
gap> PrintArray(decodev);
[ Z(2)^0, 0*Z(2), Z(2)^0, Z(2)^0, 0*Z(2), Z(2)^0 ]

```

That is, we decode the word 101001 as 101101. Similarly, the following output shows we should decode the word $w = 011001$ as 111000:

```

gap> w:=[z,a,a,z,z,a];;
gap> PrintArray(w*H);
[ Z(2)^0, Z(2)^0, Z(2)^0 ]
gap> w*H = Synd[8];
true
gap> decodew:= w - CL[8];;
gap> PrintArray(decodew);
[ Z(2)^0, Z(2)^0, Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2) ]

```

Exercises

31.1 Find the parity check matrix of the binary linear code whose generator matrix is

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

[Gallian, Chapter 31, Exercise 13]

31.2 For the (7,4) binary linear code in Exercise 31.1, use GAP and the parity-check matrix method to decode each of the following received words

0001111, 0101011, 0111101, 0101110

.

31.3 Find the parity check matrix of the binary linear code whose generator matrix is

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

31.4 For the (6,3) binary linear code in Exercise 31.3, use GAP and the parity-check matrix method to decode each of the following received words

001001, 011000, 000110, 100001.

[Gallian, Chapter 31, Exercise 17]

31.5 Redo Exercise 31.2 using GAP and the syndrome decoding method.

31.6 Redo Exercise 31.4 using GAP and the syndrome decoding method.