

---

# Error Correcting Codes

## 3. Reed Solomon Codes

---

*Priti Shankar*

**Reed Solomon codes began as mathematical curiosities; today they are perhaps the most practical and widely used codes. This article describes the ingenious application of these codes to error correction in Compact Disc audio systems.**

### Introduction

The year 1983 witnessed the introduction of one of the most successful consumer products of all time – the Compact Disc (CD) digital audio system. It was engineering on the grand scale – the use of new material for the optical discs, a solid state laser to read digital information, optical servomechanisms for tracking, and electronics for error correction – all packed into a single unit. Most people who have had an opportunity to listen to this sound medium, acknowledge the vast improvement in sound quality over that available with previous systems. The improved sound quality is, in essence, obtained by accurate waveform coding and decoding of the audio signals. In addition, the coded audio information is protected against disc errors by the use of a Cross Interleaved Reed-Solomon Code (CIRC). Reed–Solomon codes were discovered by Irving Reed and Gus Solomon in 1960. While the codes were notable because of their elegant mathematical structure, their practical versatility became apparent only after the discovery in 1966, of an efficient decoding algorithm.

### Background

Reed–Solomon codes belong to the class of *block codes* introduced in the first two articles of this series. In a block code a block of



**Priti Shankar is with the Department of Computer Science and Automation at the Indian Institute of Science, Bangalore. Her interests are in Theoretical Computer Science.**

Previous articles of this series were:

1. How numbers protect themselves, October 1996.
2. The Hamming codes, January 1997.





*Irving Reed and Gus Solomon.*

The *Hamming distance* between two  $n$  symbol words is the number of corresponding positions in which they differ. The *minimum distance* of a code is the minimum of the Hamming distances between all pairs of distinct codewords.

$k$  information symbols is encoded into  $n$  code symbols. In the application we describe here, a symbol is encoded as an 8 bit byte. The  $n - k$  extra symbols are used for error correction. The *Hamming distance* between two  $n$  symbol words is the number of corresponding positions in which they differ. Thus if  $d$  symbol errors occur in a transmitted codeword, the Hamming distance between the codeword and the received word is  $d$ . The *minimum distance* of a code (a code being the collection of all codewords) is the minimum of the Hamming distances between all pairs of distinct codewords. We saw in a previous article that for a linear code, this was also the minimum *Hamming weight* of a codeword (that is, the number of non zero components of the word). In general, if up to  $t$  symbol errors in an arbitrary codeword have to be corrected, the minimum distance  $d_{min}$  must satisfy

$$d_{min} \geq 2t + 1.$$

In the application described in this article, the concept of *erasure* decoding is important. The  $i^{\text{th}}$  position in a block code is called an erasure position, if the symbol value at that position is unreliable. Let us assume for the time being that we are able to decide independently of the decoder, whether a position is reliable or not. The aim of erasure correction is to determine the correct symbol values at the given erasure positions. One can intuitively conclude that more errors can be corrected when the positions are already known, than when the positions are unknown. In general, for a code with a minimum distance  $d_{min}$ ,  $d_{min} - 1$  erasures can be corrected. A code can correct all combinations of  $t$  errors and  $e$  erasures if  $d_{min} \geq 2t + e + 1$ . Thus the strategy followed by a decoder which corrects both errors and erasures, depends on the choices for  $t$  and  $e$  for a given value of  $d_{min}$ . We could also use a code for simultaneous error correction and detection, as we saw in the previous article in this series. Thus a code with minimum distance  $d_{min}$  could detect  $d_{min} - 1$  errors, or simultaneously detect  $d$  errors and correct  $t$  errors if  $d_{min} \geq t + d + 1$ . For example, a code with minimum



distance 5 could be used as a single error correcting, triple error detecting code; it could be used for 4-erasure correction, or to simultaneously correct single errors and two erasures. We next describe the construction of the Reed–Solomon codes.

## Reed Solomon codes

Recall that the binary Hamming codes worked with bits. These were considered as elements of a finite field with two elements. Reed Solomon codes work with elements in a larger field. These elements can be represented as bytes.

From now on we concentrate our attention on big fields with  $2^m$  elements. The construction of these large fields is illustrated in *Box 1*.

### Box 1. Construction of Finite Fields

The basic building blocks for these large fields or extension fields, as they are called, are the prime fields.  $F_p$  is the field whose elements are the set  $\{0, 1, 2, \dots, p-1\}$  where  $p$  is prime and all arithmetic is performed modulo  $p$ . We illustrate with the construction of a field with  $p=2$  and with  $2^3=8$  elements. Let  $V_3(F_2)$  be the set of 3-tuples  $\mathbf{a} = (a_2, a_1, a_0)$  over  $F_2$ , with addition being defined component wise. Thus  $V_3(F_2)$  is the set  $\{000, 001, 010, 011, 100, 101, 110, 111\}$ , and, for example,  $(101) + (110) = (011)$ . We can make  $V_3(F_2)$  into a field as follows. Assume  $f(x)$  is of degree 3 over  $F_2$  and has no roots in  $F_2$ . For example  $f(x) = x^3 + x + 1$ . (No roots in  $F_2$  would mean  $f(1)$  and  $f(0)$  are both non zero modulo 2.) In such a case,  $f(x)$  is said to be *irreducible* over  $F_2$ . Associate with each tuple  $\mathbf{a} = (a_2, a_1, a_0)$  in  $V_3(F_2)$  the polynomial  $a_2 x^2 + a_1 x + a_0$ , and define the product of tuples  $\mathbf{a}$  and  $\mathbf{b}$  to be the tuple  $\mathbf{c}$  defined uniquely by the equation  $a(x)b(x) = c(x) \pmod{f(x)}$ . ( $c(x)$  is the remainder when the product  $a(x)b(x)$  is divided by  $f(x)$ .) For example  $(010)(101) = (001)$  as  $(x)(x^2 + 1) \equiv 1 \pmod{f(x)}$ . The field constructed above, has  $2^3$  elements. Of course, since there may be several irreducible polynomials of degree 3, there will be different ways to define the multiplication above. But all these fields are isomorphic and we can talk of *the* field with  $2^3$  elements called  $GF(2^3)$ . (We will refer to this field as  $F_8$ .) Every field has what is called a *primitive* element, powers of which generate all the non zero elements of the field. For example, the element  $\alpha$  corresponding to the residue class  $\{x\}$  or the tuple  $(010)$ , is a primitive element of this field, consecutive powers of  $\alpha$  generating the sequence of elements  $(010), (100), (011), (110), (111), (101), (001)$ .



We will use the fact that the non zero elements of every such field can be generated by powers of a single element called a *primitive* element of the field. We will use the symbol  $F_q$  to denote the field with  $q$  elements. While reading the following description, one can pretend that the symbols are real numbers and still get the basic ideas behind the encoding and decoding techniques.

Reed–Solomon codes have parameters  $(n, k)$  where  $n$  is at most  $2^m - 1$ ,  $m$  is the number of bits per symbol, and  $k = n - 2t$  where  $t$  is the symbol error correction capability of the code. Thus, only  $2t$  additional parity check symbols are required, to be able to correct  $t$  errors. For example, a  $(255, 223)$  Reed-Solomon code can correct 16 symbol errors.

Before we describe the construction of the code, let us illustrate its power with a small example. Suppose one wished to transmit messages in English uppercase. Then we would need at least 5 bits (which give a total of  $2^5 = 32$  combinations) to encode all 26 letters, with 6 combinations left over to use for blanks and other special symbols like punctuation marks etc. We could use a  $(2^5 - 1, 15)$  Reed Solomon code for our purpose. This code has symbols in  $F_{32}$  each requiring 5 bits for encoding. The code has a length of 31 symbols, with 15 message symbols and 16 check symbols, and an error correction capability of 8 symbols. Suppose we wanted to transmit the fifteen characters, (including the blank)

READ RESONANCE!

The encoder would add 16 more parity check symbols of its own to form a codeword. Assume for the sake of argument that the resultant codeword is

READ RESONANCE!QTBPJ!TL.,ZBFALK

Now even if the message is changed to



## ROAD REPAIRSCE!STOPJ!TL.,ZBFALK

where errors occur in both the message as well as the check symbols, the decoder would be able to correct all of these (as there are not more than 8 of them) and recover the original message!

The encoding technique for the Reed Solomon code defined below scrambles the message, so that the message symbols do not appear in specific locations. Thus it is not quite like the encoding in the example above. The scheme below is the original scheme proposed by Reed and Solomon.

Let  $\mathbf{u}=(u_0, u_1, \dots, u_{k-1})$  be the message symbols (all of which are in the big field) to be encoded, and define polynomial  $u(x)$  of degree  $k-1$  as

$$u(x) = u_0 + u_1 x + u_2 x^2 + \dots + u_{k-1} x^{k-1}.$$

Let  $\alpha$  be a primitive element of the field. Then the codeword corresponding to  $\mathbf{u}$  is taken to be the vector  $\mathbf{c}$  such that

$$\mathbf{c} = (u(1), u(\alpha), u(\alpha^2), \dots, u(\alpha^{n-1})).$$

Let  $\mathbf{c}$  be the vector

$$\mathbf{c} = (c_0, c_1, \dots, c_{n-1}), \text{ where } c_i = u(\alpha^i).$$

From a geometric viewpoint, we are computing, during the encoding process, points on the curve  $u(x)$  of degree  $k-1$ . We know that  $k$  points are sufficient to determine the curve  $u(x)$ . However we transmit  $n$  points  $c_0, c_1, \dots, c_{n-1}$  on the curve to establish a strong pattern, so that the decoder will be able to correct errors that occur during transmission.

It turns out that the polynomial  $c(x)$  given by

$$c(x) = c_0 + c_1 x + \dots + c_{n-1} x^{n-1}$$



has  $\alpha, \alpha^2, \dots, \alpha^{2t}$  as roots.

Thus another way of defining a Reed Solomon code is to say that every code polynomial has the above  $2t$  consecutive powers of  $\alpha$  as roots.

What is the parity check matrix for this code? Remember, we defined it as a  $(n - k) \times n$  matrix  $H$  which satisfied  $H\mathbf{c}^T = 0$  for all codewords  $\mathbf{c}$ . Here, codeword symbols are in the big field; so also are the entries of  $H$ . A little thought will convince the reader that  $H$  is simply the matrix

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \dots & (\alpha^2)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & \dots & (\alpha^{2t})^{n-1} \end{bmatrix}$$

as  $H\mathbf{c}^T$  is the all zero vector.

This is just a restatement of the fact that  $c(x)$  has the  $2t$  consecutive powers of  $\alpha$  as roots.

To find the minimum distance of the code we recall the following result from the first article of this series:

*If  $C$  is an  $(n, k)$  linear code over  $F_q$  with parity check matrix  $H$ , the minimum distance of the code is at least  $2t + 1$  if every subset of  $2t$  or fewer columns of  $H$  is linearly independent.*

A matrix formed by taking a subset of  $r \leq 2t$  columns of  $H$  has the form:

$$B = \begin{bmatrix} \beta_1 & \dots & \beta_r \\ \beta_1^2 & \dots & \beta_r^2 \\ \vdots & & \vdots \\ \beta_1^{2t} & \dots & \beta_r^{2t} \end{bmatrix}$$



where  $\beta_1, \beta_2, \dots, \beta_r$  are distinct elements of  $F_q$ . Consider the matrix  $B'$  formed from the first  $r$  rows of  $B$ :

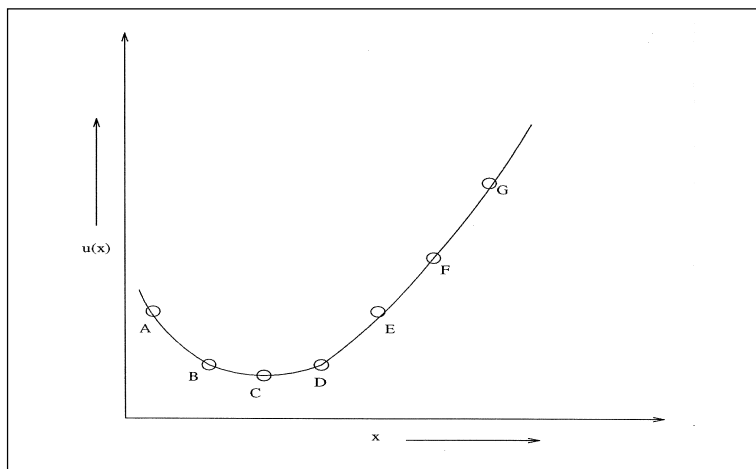
$$B' = \begin{bmatrix} \beta_1 & \dots & \beta_r \\ \vdots & & \vdots \\ \beta_1^r & \dots & \beta_r^r \end{bmatrix}.$$

We can show that  $B'$  is nonsingular since its determinant is

$$\begin{aligned} \det(B') &= \beta_1 \dots \beta_r \det \begin{bmatrix} 1 & \dots & 1 \\ \beta_1 & \dots & \beta_r \\ \vdots & & \vdots \\ \beta_1^{r-1} & \dots & \beta_r^{r-1} \end{bmatrix} \\ &= \beta_1 \dots \beta_r \prod_{i < j} (\beta_j - \beta_i) \neq 0. \end{aligned}$$

This shows that the minimum distance of the code is at least  $2t + 1$ . (The determinant of the last matrix is, in fact, a *Vandermonde determinant* mentioned in the article by V Balakrishnan in *Resonance* Vol. 1 No 8, 1996.)

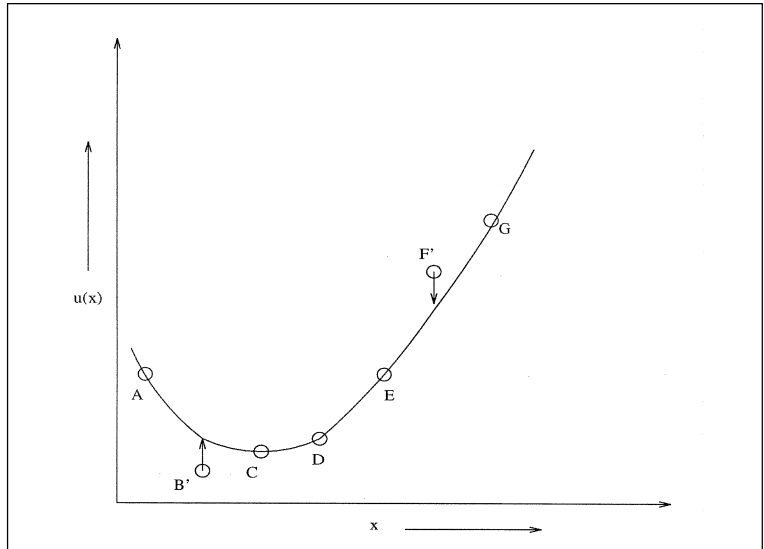
The decoding process uses a very clever algorithm devised by Elwyn Berlekamp in 1966. We saw above that encoding a message of length  $k$  symbols corresponds to computing  $n$  points on a curve of degree  $k-1$ . Let us look at a small example



*Figure 1 Sketch of a parabola with seven points on it.*



*Figure 2* Majority parabola with points B' and F' pulled back on to itself.



of a code of length seven symbols of which four are check symbols, i.e a (7,3) Reed Solomon code which is two error correcting. The three message symbols define a parabola. With reference to *Figure 1*, the encoder will compute the seven points A, B, C, D, E, F and G, and transmit them as the codeword. Assume that this codeword is corrupted by an error pattern of weight two. Say this changes points B and F to B' and F' respectively. The decoder is presented with a set of seven points, all of which do not lie on a parabola. However five of them will. This is a sufficiently strong pattern for the decoder to quickly compute the majority parabola (sketched in *Figure 2*). It is therefore able to pull back the errant points B' and F' on to the original parabola again. Of course this will not work if three or more errors occur.

We next describe the application of the Reed–Solomon code to correction of errors in the CD audio system.

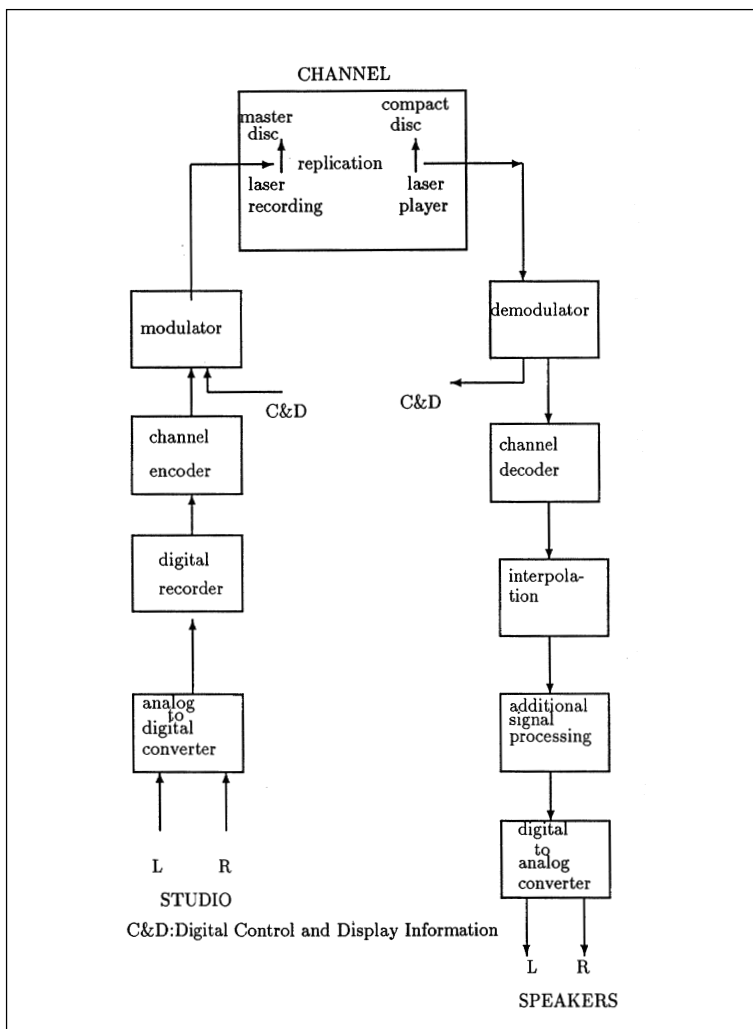
### Error Correction in Compact Discs

The Compact Disc digital audio system has the standard components of a communication system - a transmitting and a receiving end connected by a channel, and as is usual in a



communication system, many of the operations at the receiving end are the inverses of those at the transmitting end. *Figure 3* shows the various parts of the communication system.

The two audio signals that originate from the stereo input are converted from analog to digital and are recorded digitally on a magnetic tape. The channel encoder adds parity bytes (each byte consisting of eight bits), derived from two Reed Solomon Codes (to be described presently). After this, digital control and display information (containing music related data and a table



*Figure 3 The Compact Disc audio system represented as a transmission system <sup>1</sup>*

<sup>1</sup> This diagram is taken from JBH Peek, Communication Aspects of the Compact Disc Digital Audio System. *IEEE Communications Magazine*. Vol23. No 2. pp 7-15, February 1985.

of contents on the disc) is added to the encoded data. The output from the channel encoder goes through a modulator before it is conveyed to the master disc. Modulation caters for requirements like reducing inter symbol interference, and so on. Next, the CD standardized format is optically recorded on the surface of a glass disc which is coated with photoresist, called the *Master Disc*. The information is transferred in the form of *pits* to a transparent plastic disc via a nickel shell called a *stamper*. After receiving a reflective aluminium coating protected by lacquer, the CD is ready for playing.

There are several sources of channel errors. Firstly, there may be foreign particles or air bubbles in the plastic material, or inaccuracies in the pits due to stamper errors. These will cause errors when the information is optically read out. These commonly occur as random errors and the error patterns are of small lengths. On the other hand, *burst* errors caused by fingerprints and scratches on the surface of the disc, result in errors affecting several consecutive demodulated blocks, as the dimensions of the CD are minute - the track pitch is about 1.6 microns, and the average length of a pit is about one micron. (Each leading and trailing edge of a pit represents a 1; no change represents a 0.)

The CD standardized format is optically recorded on the surface of a glass disc which is coated with photoresist, called the *Master Disc*. The information is transferred in the form of *pits* to a transparent plastic disc via a nickel shell called a *stamper*.

Since the encoding and decoding strategies are inverses of one another in the case of the CD audio system, the special process of scrambling and adding check symbols can be explained with the help of the block diagram in *Figure 4* which represents the decoding strategy.

The CD decoder consists, effectively, of two decoders,  $DEC_1$  and  $DEC_2$  in series. Both decoders have a similar structure and are capable of correcting and detecting byte errors. Each uses four parity bytes for error correction. As we saw earlier, there is a trade-off between error detection and correction. In this case, an undetected erroneous audio sample results in an audible click, while if an erroneous audio sample is detected, inter-



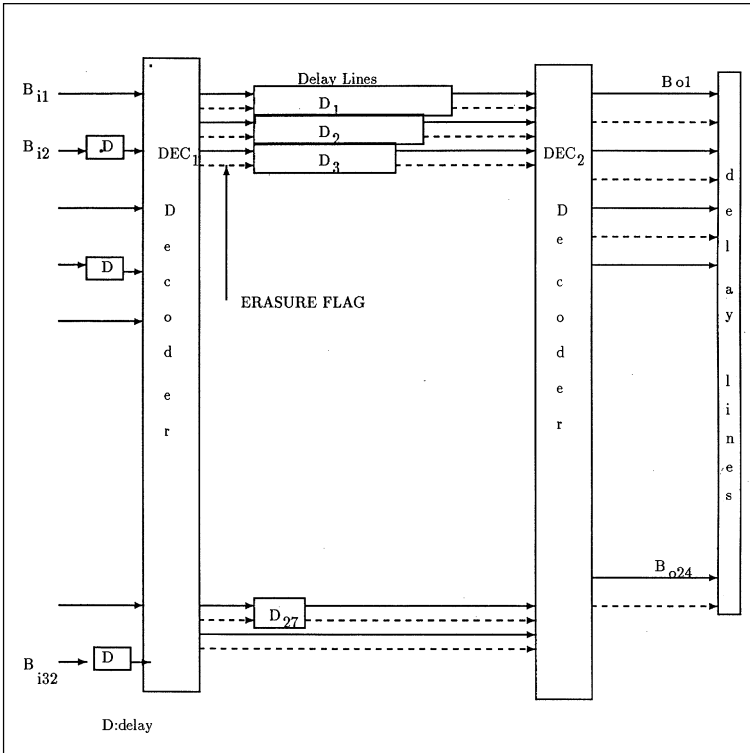


Figure 4 Schematic for the CD decoder.

polated sample values can be computed from neighbouring samples so that its effect is not heard. The decoders  $DEC_1$  and  $DEC_2$  are separated from each other and from the demodulator by deinterleaving delay lines. The effect of these lines is to distribute a single burst error on the disc among many codewords, so that there are fewer errors per codeword. The first set of deinterleaving lines and the first decoder are intended for the correction of most random single byte errors, and detection of larger burst errors. The second set of deinterleaving lines distribute the burst errors among the codewords entering  $DEC_2$ .  $DEC_1$  also provides indication of whether a symbol is reliable or not by means of erasure flags attached to all its outputs, to enable erasure decoding at  $DEC_2$ . The delay lines after  $DEC_2$  scramble uncorrectable but detected byte errors, which are considered as unreliable symbols, in such a way that their values can be interpolated between reliable neighbour samples.

The decoders are separated from each other and from the demodulator by deinterleaving delay lines which distribute a single burst error on the disc among many codewords, so that there are fewer errors per codeword.



The first decoder is used for the correction of most of the random single byte errors, and the detection of the larger burst errors.

We now describe the decoding strategy in more detail. The error correcting code used in the CD employs *two* Reed–Solomon codes  $C_1$  and  $C_2$  which are *interleaved*. For code  $C_1$  we have  $n_1 = 32$ ,  $k_1 = 28$ , and for  $C_2$  we have  $n_2 = 28$ ,  $k_2 = 24$ . The symbols for both codes are from  $F_{256}$ , that is they require 8 bits for representation. The overall rate of the interleaved code is  $(k_1/n_1)(k_2/n_2) = 3/4$ . Each code has a redundancy of four symbols, and a minimum distance of 5, and is hence capable of correcting either 2 symbol errors or four symbol erasures. Of course, one can also use the code to simultaneously correct one error *and* detect three errors, or correct a single error and two erasures.

The decoders  $DEC_1$  and  $DEC_2$  are the decoding circuits for the Reed–Solomon codes  $C_1$  and  $C_2$  respectively. The thirty two bytes or symbols  $B_{i_1} \dots B_{i_{32}}$  of a frame (corresponding to 12 audio samples encoded as 24 information symbols and eight check symbols), are applied in parallel to decoder  $DEC_1$ . Every even byte is delayed by one symbol (represented by the delay block marked  $D$ ), so that the even numbered symbols of one frame are cross interleaved with the odd numbered symbols of the next frame. Thus two consecutive bytes on the disc will end up on two *different*  $C_1$  codewords. This ensures that a small error spanning the boundary of two consecutive bytes on the disc will not cause two byte errors in the same  $C_1$  codeword. Though  $DEC_1$  can correct two byte errors, it employs a strategy whereby it corrects at most one byte error, and can therefore detect with certainty up to triple byte errors. (Thus  $t = 1$ ,  $d = 3$ .) There is a small probability of missing the detection of errors from 4 to 32 bytes. This has been estimated to be less than  $2^{-19}$ . The first decoder is used for the correction of most of the random single byte errors, and the detection of the larger burst errors.

If the decoder  $DEC_1$  detects no error, it will pass on the 28 information symbols (after stripping off the four parity check



symbols), to  $DEC_2$  with all the erasure indications off. (Each of the 28 outgoing symbols has an erasure flag attached to it which indicates if that symbol is unreliable). However, if  $DEC_1$  detects an uncorrectable error, it will set the erasure flags on all 28 bits indicating that they are all unreliable (as it is unable to give any information more specific than this).

The symbols arrive at  $DEC_2$  via the triangular network of delay lines  $D_1, \dots, D_{27}$ , the last symbol being fed in without any delay. There is a delay of four bytes between every successive pair of lines. This has the effect of spreading the symbols, (and therefore, the errors) in a single word output by  $DEC_1$  over 28 different  $DEC_2$  input words which are equidistantly spaced.

If no errors are present,  $DEC_2$  will receive words that are codewords for  $C_2$  and it will pass on 24 audio symbols unchanged to its output.  $DEC_2$  is a decoder for a code with minimum distance 5 so it can correct up to 4 erasures. If the increments in the delay lengths of the triangular network were one byte, it would be possible to correct a burst error encompassing four consecutive  $C_1$  codewords if four-erasure correction was used at the decoder  $DEC_2$ . In the actual CD system the increment equals 4 bytes, thus giving a maximum burst error correcting capability of 16 consecutive uncorrectable  $C_1$  words, or around 4000 bits, corresponding to a track length of about 2.5mm. In current CD players the possibility of correcting four erasures is not used, to allow for error detection as well. The strategy followed allows up to two-erasure correction, and up to single error correction.

If  $DEC_2$  detects an uncorrectable error, it indicates so by assigning erasure flags to all its 24 outputs. A technique involving interpolation is able to interpolate the values and restore the signal. The purpose of the delay lines at the output of  $DEC_2$  is to scramble uncorrectable but detected byte errors in such a way that these can be often interpolated between reliable neighbour samples. The maximum interpolatable burst length is around

The maximum interpolatable burst length is around 12300 bits, that is a track length of about 7.7 mm. One can deliberately scratch a CD without losing any music. One can actually drill holes in a CD and it will still play!



12300 bits, that is a track length of about 7.7 mm. One can deliberately scratch a CD without losing any music, and it is said that you can actually drill holes in a CD and it will still play!

## **Beyond Linear Codes**

Though linear codes have held sway in practical applications for the most part of coding history, recent results indicate that non linear codes may soon be carving out a niche for themselves. The main problem with linear codes is that the number of codewords for a given code length is restricted. One cannot just add a new vector which is at the appropriate distance from all other codewords—one has also to check the sum of this vector with all codewords in the code for the minimum distance property. Coding theorists have known for a long time that non linear codes of a given length have more codewords than their linear counterparts. Thus information can be sent more quickly or stored more compactly using non linear codes. However, these had so far remained mathematical curiosities as there didn't seem to be any efficient way to decode them.

Recent results have shown that many non linear codes are actually disguised versions of linear codes. To give an example, a code over  $F_2$  which is non linear, can actually be viewed as a linear code over the integers modulo 4. (Though 4 is not a prime and the numbers modulo 4 do not form a field, the theory can be fixed to define codes over such domains). New decoding schemes that take advantage of such connections can be devised. These may soon show up in a technique called code division multiple access (CDMA) which is proposed as a basis for digital cellular communication. CDMA provides a facility for many users to simultaneously broadcast signals over the same channel and assigns a special codeword to each user. More codewords for a given length allows for more users, and the distance between codewords keeps the users signals from interfering with one another.



We have seen in the three articles of this series, a practical application of Algebra. Coding theory was born out of very real engineering problems, solutions to which resulted in the discovery of codes with elegant mathematical structure. The structure present in these codes in turn, permitted the discovery of decoding algorithms, thereby allowing exploitation of the full error correction capability of the codes. As the previous paragraph suggests, the quest for better codes and newer applications is not yet over.

### Suggested Reading

- ◆ F J Macwilliams and N J A Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
- ◆ Robert J McEliece. *The Theory of Information and Coding*. *Encyclopedia of Mathematics and its Applications*. Addison Wesley, 1977.
- ◆ H Hoeve, J Timmermans, and L B Vries. Error correction and concealment in the Compact Disc system. *Philips Technical Review*. Vol 40. No.6, 1982.

Address for Correspondence

Priti Shankar

Department of Computer  
Science and Automation

Indian Institute of Science  
Bangalore 560 012, India

email:priti@csa.iisc.ernet.in

Fax:(080) 334 1683

