Chapter 10
Historical Convolutional Codes as Tail-Biting Block Codes

10.1 Introduction

In the late 1950s, a branch of error-correcting codes known as convolutional codes [1, 6, 11, 14] was explored almost independently of block codes and each discipline had their champions. For convolutional codes, sequential decoding was the norm and most of the literature on the subject was concerned with the performance of practical decoders and different decoding algorithms [2]. There were few publications on the theoretical analysis of convolutional codes. In contrast, there was a great deal of theory about linear, binary block codes and not a great deal about decoders, except for hard decision decoding of block codes. Soft decision decoding of block codes was considered to be quite impractical, except for trivial, very short codes.

With Andrew Viterbi's invention [13] of the maximum likelihood decoder in 1967, featuring a trellis based decoder, an enormous impetus was given to convolutional codes and soft decision decoding. Interestingly, the algorithm itself, for solving the travelling saleman's problem [12], had been known since 1960. Consequently, interest in hard decision decoding of convolutional codes waned in favour of soft decision decoding. Correspondingly, block codes were suddenly out of fashion except for the ubiquitous Reed–Solomon codes.

For sequential decoder applications, the convolutional codes used were systematic codes with one or more feedforward polynomials, whereas for applications using a Viterbi decoder, the convolutional codes were optimised for largest, minimum Hamming distance between codewords, d_{free}, for a given memory (the highest degree of the generator polynomials defining the code). The result is always a non-systematic code. It should be noted that in the context of convolutional codes, the minimum Hamming distance between codewords is understood to be evaluated over the constraint length, the memory of the code. This is traditionally called d_{min}. This is rather confusing when comparing the minimum Hamming distance of block codes with that of convolutional codes. A true comparison should compare the d_{free} of a convolutional code to the d_{min} of a block code, for a given code rate.

M. Tomlinson et al., *Error-Correction Coding and Decoding*,
Signals and Communication Technology,
DOI 10.1007/978-3-319-51103-0_10

Table 10.1 Best rate $\frac{1}{2}$ convolutional codes designed for Viterbi decoding

Memory	Generator Polynomial $r_1(x)$	Generator Polynomial $r_2(x)$	d_{free}
2	$1+x+x^2$	$1+x^2$	5
3	$1+x+x^2+x^3$	$1+x+x^3$	6
4	$1+x+x^2+x^4$	$1+x^3+x^4$	7
5	$1+x+x^2+x^3+x^5$	$1+x^2+x^4+x^5$	8
6	$1+x+x^2+x^3+x^6$	$1+x^2+x^3+x^5+x^6$	10
7	$1+x+x^2+x^3+x^4+x^7$	$1+x^2+x^5+x^6+x^7$	10
8	$1+x+x^2+x^3+x^5+x^7+x^8$	$1+x^2+x^3+x^4+x^8$	12

Since the early 1960s, a lot of work has been carried out on block codes and convolutional codes for applications in deep space communications, primarily because providing a high signal-to-noise ratio is so expensive. Error-correcting codes allowed the signal to noise ratio to be reduced.

The first coding arrangement implemented for space [6, 9] was part of the payload of Pioneer 9 which was launched into space in 1968. The payload featured a systematic, convolutional code designed by Lin and Lyne [7] with a d_{free} of 12 and memory of 20. The generator polynomial is

$$r(x) = 1+x+x^2+x^5+x^6+x^8+x^9+x^{12}+x^{13}+x^{14}+x^{16}+x^{17}+x^{18}+x^{19}+x^{20}.$$

This convolutional code was used with soft decision, sequential decoding featuring the Fano algorithm [2] to realise a coding gain of 3 dB. Interestingly, it was initially planned as a communications experiment and not envisaged to be used operationally to send telemetry data to Earth. However, its superior performance over the standard operational communications system which featured uncoded transmission meant that it was always used instead of the standard system.

In 1969, the Mariner'69 spacecraft was launched with a first order Reed–Muller (32, 6, 16) code [8] equivalent to the extended (32, 6, 16) cyclic code. A maximum likelihood correlation decoder was used. The coding gain was 2.2 dB [9].

By the mid 1970s, the standard for soft decision decoding on the AWGN channel notably applications for satellite communications and space communications was to use convolutional codes with Viterbi decoding, featuring the memory 7 code listed in Table 10.1. The generator polynomials are $r_1(x) = 1+x+x^2+x^3+x^6$ and $r_2(x) = 1+x^2+x^3+x^5+x^6$ convolutional code, best known, in octal representation, as the (171, 133) code. The best half rate convolutional codes designed to be used with Viterbi decoding [1, 6] are tabulated in Table 10.1.

The (171, 133) code with Viterbi soft decision decoding featured a coding gain of 5.1 dB at 10^{-5} bit error rate which was around 2 dB better than its nearest rival featuring a high memory convolutional code and hard decision, sequential decoding. The (171, 133) convolutional code is one of the recommended NASA Planetary Standard Codes [3].

However, more coding gain was achieved by concatenating the (171, 133) convolutional code with a (255, 233) Reed–Solomon (RS) code which is able to correct 16 symbol errors, each symbol being 8 bits. Quite a long interleaver needs to be used between the Viterbi decoder output and the RS decoder in order to break up the occasional error bursts which are output from the Viterbi decoder. Interleaver lengths vary from 4080 bits to 16320 bits and with the longest interleaver the coding gain of the concatenated arrangement is 7.25 dB, ($\frac{E_b}{N_0} = 2.35$ dB at 10^{-5} bit error rate), and it is a CCSDS [3] standard for space communications.

10.2 Convolutional Codes and Circulant Block Codes

It is straightforward to show that a double-circulant code is a half rate, tail-biting, feedforward convolutional code. Consider the Pioneer 9, half rate, convolutional code invented by Lin and Lyne [7] with generator polynomial

$$r(x) = 1+x+x^2+x^5+x^6+x^8+x^9+x^{12}+x^{13}+x^{14}+x^{16}+x^{17}+x^{18}+x^{19}+x^{20}$$

For a semi-infinite data sequence defined by $d(x)$, the corresponding codeword, $c(x)$, of the convolutional code consists of

$$c(x) = d(x)\|d(x)r(x) \tag{10.1}$$

where $\|$ represents interlacing of the data polynomial representing the data sequence and the parity polynomial representing the sequence of parity bits.

The same generator polynomial can be used to define a block code of length $2n$, a $(2n, n)$ double-circulant code with a codeword consisting of

$$c(x) = d(x)\|d(x)r(x) \text{ modulo } (1+x^n) \tag{10.2}$$

(Double-circulant codewords usually consist of one circulant followed by the second but it is clear that an equivalent code is obtained by interlacing the two circulants instead.)

While comparing Eq. (10.1) with (10.2) as $n \to \infty$, it can be seen that the same codewords will be obtained. For finite n, it is apparent that the tail of the convolution of $d(x)$ and $r(x)$ will wrap around adding to the beginning as in a tail-biting convolutional code. It is also clear that if n is sufficiently long, only the Hamming weight of long convolutions, will be affected by the wrap around and these long convolution results will be of high Hamming weight anyway leading to the conclusion that if n is sufficiently long the d_{min} of the circulant code will be the same as the d_{free} of the convolutional code. Indeed, the low weight spectral terms of the two codes will be identical, as is borne out by codeword enumeration using the methods described in Chap. 5.

For the Pioneer 9 code, having a d_{free} of 12, a double-circulant code with d_{min} also equal to 12 can be obtained with n as low as 34, producing a (68, 34, 12) code. It is noteworthy that this is not a very long code, particularly by modern standards.

Codewords of the double-circulant code are given by

$$c(x) = d(x)|d(x)(1 + x + x^2 + x^5 + x^6 + x^8 + x^9 + x^{12} + x^{13} + x^{14} + x^{16} + x^{17} + x^{18} + x^{19} + x^{20}) \text{ modulo } (1 + x^{34}) \quad (10.3)$$

As a double-circulant block code, this code can be soft decision decoded, with near maximum likelihood decoding using an extended Dorsch decoder, described in Chap. 15. The results for the AWGN channel are shown plotted in Fig. 10.1. Also plotted in Fig. 10.1 are the results obtained with the same convolutional code realised as a (120, 60, 12) double-circulant code which features less wrap around effects compared to the (68, 34, 12) code.

Using the original sequential decoding with 8 level quantisation of the soft decisions realised a coding gain of 3 dB at a BER of 5×10^{-4}. Using the modified Dorsch decoder with this code can realise a coding gain of over 5 dB at a BER of 5×10^{-4} and over 6 dB at a BER of 10^{-6} as is evident from Fig. 10.1. Moreover, there is no need for termination bits with the tail-biting arrangement. However, it should be noted that the state of the art, modified Dorsch decoder with soft decision decoding

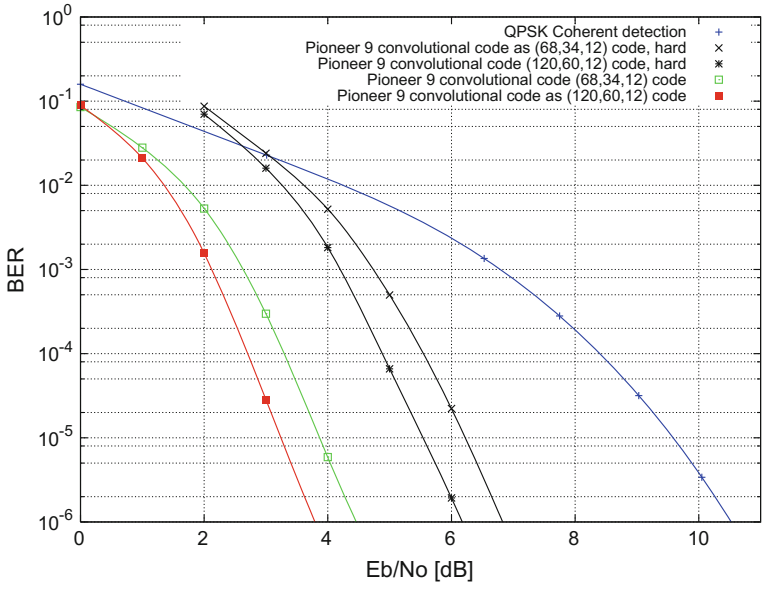

Fig. 10.1 BER performance of the Pioneer 9 convolutional code encoded as a (68, 34, 12) or (120, 60, 12) double-circulant code with soft and hard decision, extended Dorsch decoding in comparison to uncoded QPSK

needs to evaluate up to 500,000 codewords per received vector for the (68, 34, 12) double-circulant code realisation and up to 1,000,000 codewords per received vector for the (120, 60, 12) double-circulant code version in order to achieve near maximum likelihood decoding. Figure 10.1 also shows the hard decision decoding performance realised with the modified, hard decision Dorsch decoder, also described in Chap. 15. The (120, 60, 12) double-circulant code version, has a degradation of 2.3 dB at 10^{-4} BER compared to soft decision decoding, but still achieves a coding gain of 3.3 dB at 10^{-4} BER. Similarly, the (68, 34, 12) double-circulant code version, has a degradation of 2.2 dB at 10^{-4} BER compared to soft decision decoding, but still achieves a coding gain of 2.3 dB at 10^{-4} BER.

The conclusion to be drawn from Fig. 10.1 is that the Pioneer 9 coding system was limited not by the design of the code but by the design of the decoder. However to be fair, the cost of a Dorsch decoder would have been considered beyond reach back in 1967.

It is interesting to discuss the differences in performance between the (68, 34, 12) and (120, 60, 12) double-circulant code versions of the Pioneer 9 convolutional code. Both have a d_{min} of 12. However the number of weight 12 codewords, the multiplicities of weight 12 codewords of the codes' weight distributions, is higher for the (68, 34, 12) double-circulant code version due to the wrap around of the second circulant which is only of length 34. The tails of the circulants of codewords of higher weight than 12 do suffer some cancellation with the beginning of the circulants. In fact, exhaustive weight spectrum analysis, (see Chaps. 5 and 13 for description of the different methods that can be used), shows that the multiplicity of weight 12 codewords is 714 for the (68, 34, 12) code and only 183 for the (120, 60, 12) code.

Moreover, the covering radius of the (68, 34, 12) code has been evaluated and found to be 10 indicating that this code is well packed, whereas the covering radius of the (120, 60, 12) code is much higher at 16 indicating that the code is not so well packed. Indeed the code rate of the (120, 60, 12) code can be increased without degrading the minimum Hamming distance because with a covering radius of 16 at least one more information bit may be added to the code.

With maximum likelihood, hard decision decoding, which the modified Dorsch decoder is able to achieve, up to 10 hard decision errors can be corrected with the (68, 34, 12) code in comparison with up to 16 hard decision errors correctable by the (120, 60, 12) code. Note that these are considerably higher numbers of correctable errors in both cases than suggested by the d_{free} of the code (only five hard decision errors are guaranteed to be correctable). This is a recurrent theme for maximum likelihood, hard decision decoding of codes, as discussed in Chap. 3, compared to bounded distance decoding.

It is also interesting to compare the performance of other convolutional codes that have been designed for space applications and were originally intended to be used with sequential decoding. Of course now we have available the far more powerful (and more signal processing intensive) modified Dorsch decoder, which can be used with any linear code.

Massey and Costello [6, 10] constructed a rate $\frac{1}{2}$, memory 31 non-systematic code which was more powerful than any systematic code with the same memory

and had the useful property that the information bits could be obtained from the two convolutionally encoded parity streams just by adding them together, modulo 2. The necessary condition for this property is that the two generator polynomials differ only in a single coefficient. The two generator polynomials, $r_0(x)$ and $r_1(x)$ may be described by the exponents of the non-zero coefficients:

$$r_0(x) \leftarrow \{0, 1, 2, 4, 5, 7, 8, 9, 11, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 31\}$$
$$r_1(x) \leftarrow \{0, 2, 4, 5, 7, 8, 9, 11, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 31\}$$

As can be seen the two generator polynomials differ only in the coefficient of x. This code has a d_{free} of 23 and can be realised as a double-circulant (180, 90, 23) code from the tail-biting version of the same convolutional code. This convolutional code has exceptional performance and in double-circulant form it, of course, may be decoded using the extended Dorsch decoder. The performance of the code in (180, 90, 23) form, for the soft decision and hard decision AWGN channel, is shown in Fig. 10.2. For comparison purposes, the performances of the Pioneer 9 codes are also shown in Fig. 10.2. Shorter double-circulant code constructions are possible from this convolutional code in tail-biting form, without compromising the d_{min} of the double-circulant code. The shortest version is the (166, 83, 23) double-circulant code.

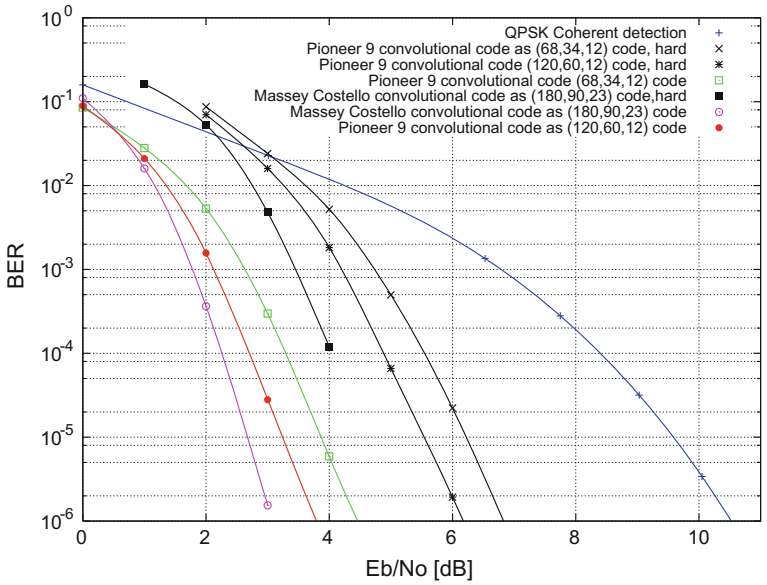

Fig. 10.2 BER performance of the Massey Costello convolutional code in (180, 90, 23) double-circulant code form for the AWGN channel, using soft and hard decisions, with extended Dorsch decoding

By truncating the generator polynomials, $r_0(x)$ and $r_1(x)$ above, a reduced memory convolutional code with memory 23 and d_{free} of 17 can be obtained as discussed by Massey and Lin [6, 10] which still has the non-systematic, quick decoding property. The generator polynomials are given by the exponents of the non-zero coefficients:

$$\hat{r}_0(x) \leftarrow \{0, 1, 2, 4, 5, 7, 8, 9, 11, 13, 14, 16, 17, 18, 19, 21, 22, 23\}$$
$$\hat{r}_1(x) \leftarrow \{0, 2, 4, 5, 7, 8, 9, 11, 13, 14, 16, 17, 18, 19, 21, 22, 23\}$$

A (160, 80, 17) double-circulant code can be obtained from the tail-biting version of this convolutional code. In fact, many double-circulant codes with high d_{min} can be obtained from tail-biting versions of convolutional codes.

It is straightforward to write a program in C++ which searches for the generator polynomials that produce the convolutional codes with the highest values of d_{free}. The only other constraint is that the the generator polynomials need to be relatively prime to each other, that is, the GCD of the generator polynomials needs to be 1 in order to avoid a catastrophic code [6]. However, it is also necessary in selecting the generator polynomials that the wrap around effects of the circulants are taken into account otherwise the d_{min} of the double-circulant code is not as high as the d_{free} of the convolutional code from which it is derived. Indeed to construct a good code in this way with high d_{free} and high d_{min}, it has to be constructed as a tail-biting convolutional code right from the start. One example of a good tail-biting convolutional code that has been found in this way has generator polynomials, $r_0(x)$ and $r_1(x)$ given by the exponents of the non-zero coefficients:

$$r_0(x) \leftarrow \{0, 2, 5, 8, 9, 10, 12, 13, 14, 15, 27\}$$
$$r_1(x) \leftarrow \{0, 1, 2, 3, 4, 5, 7, 8, 11, 12, 16, 18, 20, 23, 27\}$$

This code has a memory of 27 and a d_{free} of 26. It may be realised in double-circulant form as a (180, 90, 26) double-circulant code and weight spectrum analysis shows that this code has the same d_{min} of 26 as the best-known code with the same code parameters [4]. The two polynomials $r_0(x)$ and $r_1(x)$ factorise into polynomials with the following exponents of the non-zero coefficients:

$$r_0(x) \leftarrow \{0, 3, 5\}\{0, 2, 3, 5, 6, 7, 8, 10, 13, 14, 16, 17, 18, 20, 22\}$$
$$r_1(x) \leftarrow \{0, 3, 5, 6, 8\}\{0, 1, 3, 4, 5, 6, 8\}\{0, 2, 4, 7, 11\}$$

It can be seen that neither polynomial has a common factor and so the GCD is 1. Correspondingly, the convolutional code is not a catastrophic code.

As well as constructing double-circulant codes from convolutional codes, double-circulant codes may be used to construct good convolutional codes. The idea of generating convolutional codes from good block codes is not that new. Massey et al. in 1973 generated a convolutional code for space communications from a (89, 44, 18) quadratic residue cyclic code [5, 6]. As described in Chap. 9, prime numbers which

are congruent to ± 3 modulo 8 may be used to generate double-circulant codes using the quadratic residues to construct one circulant, the other circulant being the identity circulant; the length of the circulants are equal to the prime number.

Particularly, good double-circulant codes are obtained in this way as discussed in Chap. 9. For example, the prime number 67 can be used to generate a (134, 67, 23) double-circulant code with the circulants defined by the two polynomials with the following exponents of the non-zero coefficients:

$$
\begin{aligned}
r_0(x) &\leftarrow \{0\} \\
r_1(x) &\leftarrow \{0, 1, 4, 6, 9, 10, 14, 15, 16, 17, 19, 21, 22, 23, 24, 25, 26, 29, 33, 35, 36, \\
&\qquad 37, 39, 40, 47, 49, 54, 55, 56, 59, 60, 62, 64, 65\}
\end{aligned}
$$

Using these two polynomials as the generator polynomials for a $\frac{1}{2}$ rate convolutional code, a systematic convolutional code having a d_{free} of 30 is obtained. Interestingly, deriving another double-circulant code from the tail-biting version of this convolutional code only produces good results when the circulants are exactly equal to 67, thereby reproducing the original code. For longer circulants, the d_{min} is degraded unless the circulants are much longer. It is found that the circulants have to be as long as 110 to produce a (220, 110, 30) double-circulant code having a d_{min} equal to that of the original convolutional code. Moreover, this is a good code because the code has the same parameters as the corresponding best-known code [4].

A double-circulant code may also be used to derive a non-systematic convolutional code with much smaller memory and a d_{free} equal to the d_{min} of the double-circulant code by selecting a codeword of the double-circulant code which features low-degree polynomials in each circulant. It is necessary to check that these polynomials are relatively prime otherwise a catastrophic convolutional code is produced. In this event a new codeword is selected. The code produced is a non-systematic convolutional code with memory equal to the highest degree of the two circulant polynomials. For example, a memory 41 non-systematic convolutional code can be derived from a memory 65, systematic convolutional code based on the (134, 67, 23) double-circulant code with the following exponents of the non-zero coefficients:

$$
\begin{aligned}
r_0(x) &\leftarrow \{0\} \\
r_1(x) &\leftarrow \{0, 1, 4, 6, 9, 10, 14, 15, 16, 17, 19, 21, 22, 23, 24, 25, 26, 29, 33, 35 \\
&\qquad 36, 37, 39, 40, 47, 49, 54, 55, 56, 59, 60, 62, 64, 65\}
\end{aligned}
$$

Codeword analysis of the double-circulant code is carried out to find the low memory generator polynomials. The following two generator polynomials were obtained from the two circulant polynomials making up a weight 23 codeword of the (134, 67, 23) code:

$$
\begin{aligned}
r_0(x) &\leftarrow \{0, 1, 2, 4, 5, 10, 12, 32, 34, 36, 39, 41\} \\
r_1(x) &\leftarrow \{0, 2, 4, 13, 19, 24, 25, 26, 33, 35, 37\}
\end{aligned}
$$

In another example, the outstanding (200, 100, 32) extended cyclic quadratic residue code may be put in double-circulant form using the following exponents of the non-zero coefficients:

$$r_0(x) \leftarrow \{0\}$$
$$r_1(x) \leftarrow \{0, 1, 2, 5, 6, 8, 9, 10, 11, 15, 16, 17, 18, 19, 20, 26, 27, 28, 31, 34, 35, 37, 38, 39, 42, 44, 45, 50, 51, 52, 53, 57, 58, 59, 64, 66, 67, 70, 73, 75, 76, 77, 80, 82, 85, 86, 89, 92, 93, 97, 98\}$$

Enumeration of the codewords shows that there is a weight 32 codeword that defines the generator polynomials of a memory 78, non-systematic convolutional code. The codeword consists of two circulant polynomials, the highest degree of which is 78. The generator polynomials have the following exponents of the non-zero coefficients:

$$r_0(x) \leftarrow \{0, 2, 3, 8, 25, 27, 37, 44, 50, 52, 55, 57, 65, 66, 67, 69, 74, 75, 78\}$$
$$r_1(x) \leftarrow \{0, 8, 14, 38, 49, 51, 52, 53, 62, 69, 71, 72, 73\}$$

The non-systematic convolutional code that is produced has a d_{free} of 32 equal to the d_{min} of the double-circulant code. Usually, it is hard to verify high values of d_{free} for convolutional codes, but in this particular case, as the convolutional code has been derived from the (200, 100, 32) extended quadratic residue, double-circulant code which is self-dual and also fixed by the large projective special linear group $PSL_2(199)$ the d_{min} of this code has been proven to be 32 as described in Chap. 9. Thus, the non-systematic convolutional code that is produced has to have a d_{free} of 32.

10.3 Summary

Convolutional codes have been explored from a historical and modern perspective. Their performance, as traditionally used, has been compared to the performance realised using maximum likelihood decoding featuring an extended Dorsch decoder with the convolutional codes implemented as tail-biting block codes. It has been shown that the convolutional codes designed for space applications and sequential decoding over 40 years ago were very good codes, comparable to the best codes known today. The performance realised back then was limited by the sequential decoder as shown by the presented results. An additional 2 dB of coding gain could have been realised using the modern, extended Dorsch decoder instead of the sequential decoder. However back then, this decoder had yet to be discovered and was probably too expensive for the technology available at the time.

It has also been shown that convolutional codes may be used as the basis for designing double-circulant block codes and vice versa. In particular, high, guaranteed values of d_{free} may be obtained by basing convolutional codes on outstanding double-circulant codes. A memory 78, non-systematic, half rate convolutional code with a d_{free} of 32 was presented based on the (200, 100, 32) extended quadratic residue, double-circulant code.

References

1. Clark, G.C., Cain, J.B.: Error-Correction Coding for Digital Communications. Plenum Publishing Corporation, New York (1981). ISBN 0 306 40615 2
2. Fano, R.: A heuristic discussion of probabilistic decoding. IEEE Trans. Inf. Theory **IT–9**, 64–74 (1963)
3. Gaborit, P., Otmani, A.: Tm synchronization and channel coding. CCSDS 131.0-B-1 BLUE BOOK (2003)
4. Grassl, M.: Code tables: bounds on the parameters of various types of codes (2007). http://www.codetables.de
5. Massey, J.L., Costello Jr., D.J., Justesen, J.: Polynomial weights and code constructions. IEEE Trans. Inf. Theory **IT–19**, 101–110 (1973)
6. Lin, S., Costello Jr., D.J.: Error Control Coding: Fundamentals and Applications, 2nd edn. Pearson Education Inc., Upper Saddle River (2004)
7. Lin, S., Lyne, H.: Some results on binary convolutional code generators. IEEE Trans. Inf. Theory **IT–13**, 134–139 (1967)
8. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-Holland, Amsterdam (1977)
9. Massey, J.: Deep-space communications and coding: a marriage made in heaven. In: Hagenauer, J. (ed.) Advanced Methods for Satellite and Deep Space Communications. Lecture Notes in Control and Information Sciences 182. Springer, Heidelberg (1992)
10. Massey, J.L., Costello Jr., D.J.: Nonsystematic convolutional codes for sequential decoding in space applications. IEEE Trans. Commun. **COM–19**, 806–813 (1971)
11. Peterson, W.: Error-Correcting Codes. MIT Press, Cambridge (1961)
12. Pollack, M., Wiebenson, W.: Solutions of the shortest route problem - a review. Oper. Res. **8**, 224–230 (1960)
13. Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans. Inf. Theory **IT–13**, 260–269 (1967)
14. Wozencraft, J.: Sequential decoding for reliable communications. Technical report No. 325 Research Laboratory of Electronics, MIT (1957)