Convolutional code

In telecommunication, a **convolutional code** is a type of error-correcting code that generates parity symbols via the sliding application of a boolean polynomial function to a data stream. The sliding application represents the 'convolution' of the encoder over the data, which gives rise to the term 'convolutional coding'. The sliding nature of the convolutional codes facilitates trellis decoding using a time-invariant trellis. Time invariant trellis decoding allows convolutional codes to be maximum-likelihood soft-decision decoded with reasonable complexity.

The ability to perform economical maximum likelihood soft decision decoding is one of the major benefits of convolutional codes. This is in contrast to classic block codes, which are generally represented by a time-variant trellis and therefore are typically hard-decision decoded. Convolutional codes are often characterized by the base code rate and the depth (or memory) of the encoder $[n, k, K]$. The base code rate is typically given as n/k, where n is the input data rate and k is the output symbol rate. The depth is often called the "constraint length" K, where the output is a function of the current input as well as the previous $K-1$ inputs. The depth may also be given as the number of memory elements v in the polynomial or the maximum possible number of states of the encoder (typically : 2^v).

Convolutional codes are often described as continuous. However, it may also be said that convolutional codes have arbitrary block length, rather than being continuous, since most real-world convolutional encoding is performed on blocks of data. Convolutionally encoded block codes typically employ termination. The arbitrary block length of convolutional codes can also be contrasted to classic block codes, which generally have fixed block lengths that are determined by algebraic properties.

The code rate of a convolutional code is commonly modified via symbol puncturing. For example, a convolutional code with a 'mother' code rate $n/k = 1/2$ may be punctured to a higher rate of, for example, $7/8$ simply by not transmitting a portion of code symbols. The performance of a punctured convolutional code generally scales well with the amount of parity transmitted. The ability to perform economical soft decision decoding on convolutional codes, as well as the block length and code rate flexibility of convolutional codes, makes them very popular for digital communications.

Contents

History

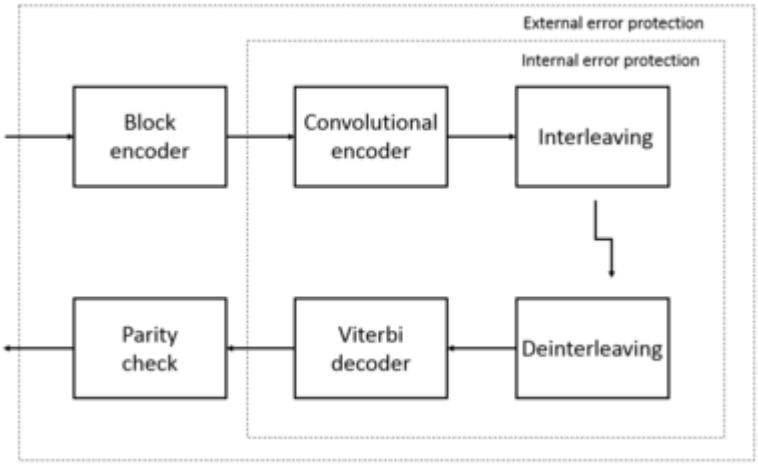
Convolutional codes were introduced in 1955 by Peter Elias. It was thought that convolutional codes could be decoded with arbitrary quality at the expense of computation and delay. In 1967 Andrew Viterbi determined that convolutional codes could be maximum-likelihood decoded with reasonable complexity using time invariant trellis based decoders — the Viterbi algorithm. Other trellis-based decoder algorithms were later developed, including the BCJR decoding algorithm.

Recursive systematic convolutional codes were invented by Claude Berrou around 1991. These codes proved especially useful for iterative processing including the processing of concatenated codes such as turbo codes[1].

Using the "convolutional" terminology, a classic convolutional code might be considered a Finite impulse response (FIR) filter, while a recursive convolutional code might be considered an Infinite impulse response (IIR) filter.

Where convolutional codes are used

Convolutional codes are used extensively to achieve reliable data transfer in numerous applications, such as digital video, radio, mobile communications (e.g., in GSM, GPRS, EDGE and 3G networks (until 3GPP Release 7)[3][4]) and satellite communications.[5] These codes are often implemented in concatenation with a hard-decision code, particularly Reed–Solomon. Prior to turbo codes such constructions were the most efficient, coming closest to the Shannon limit.



Stages of channel coding in GSM[2]. Block encoder and Parity check - error detection part. Convolutional encoder and Viterbi decoder - error correction part. Interleaving and Deinterleaving - code words separation increasing in time domain and to avoid bursty distortions.

Convolutional encoding

To convolutionally encode data, start with k memory registers, each holding one input bit. Unless otherwise specified, all memory registers start with a value of 0. The encoder has n modulo-2 adders (a modulo 2 adder can be implemented with a single Boolean XOR gate, where the logic is: $0+0 = 0$, $0+1 = 1$, $1+0 = 1$, $1+1 = 0$), and n generator polynomials — one for each adder (see figure below). An input bit m_1 is fed into the leftmost register. Using the generator polynomials and the existing values in the remaining registers, the encoder outputs n symbols. These symbols may be transmitted or

punctured depending on the desired code rate. Now bit shift all register values to the right (m_1 moves to m_0, m_0 moves to m_{-1}) and wait for the next input bit. If there are no remaining input bits, the encoder continues shifting until all registers have returned to the zero state (flush bit termination).

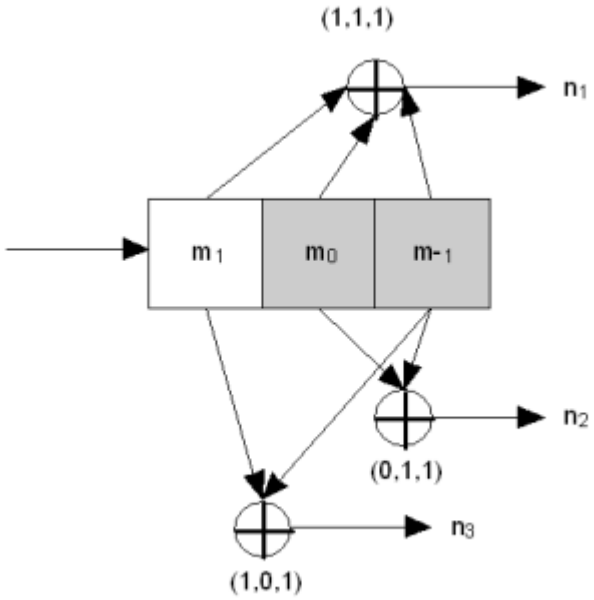
The figure below is a rate $1/3$ (m/n) encoder with constraint length (k) of 3. Generator polynomials are $G_1 = (1,1,1)$, $G_2 = (0,1,1)$, and $G_3 = (1,0,1)$. Therefore, output bits are calculated (modulo 2) as follows:

$$n_1 = m_1 + m_0 + m_{-1}$$
$$n_2 = m_0 + m_{-1}$$
$$n_3 = m_1 + m_{-1}.$$

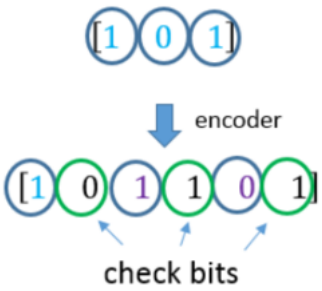
Convolutional codes can be systematic and non-systematic:

- systematic repeats the structure of the message before encoding
- non-systematic changes the initial structure

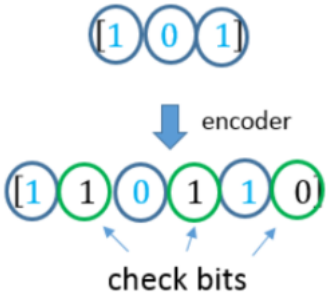
Non-systematic convolutional codes are more popular due to better noise immunity. It relates to the free distance of the convolutional code.[6]



Img.1. Rate 1/3 non-recursive, non-systematic convolutional encoder with constraint length 3



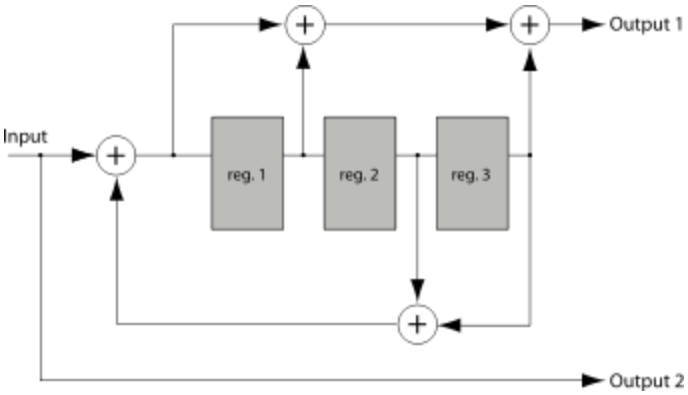
A short illustration of non-systematic convolutional code.



A short illustration of systematic convolutional code.

Recursive and non-recursive codes

The encoder on the picture above is a *non-recursive* encoder. Here's an example of a recursive one and as such it admits a feedback structure:



Img.2. Rate 1/2 8-state recursive systematic convolutional encoder. Used as constituent code in 3GPP 25.212 Turbo Code.

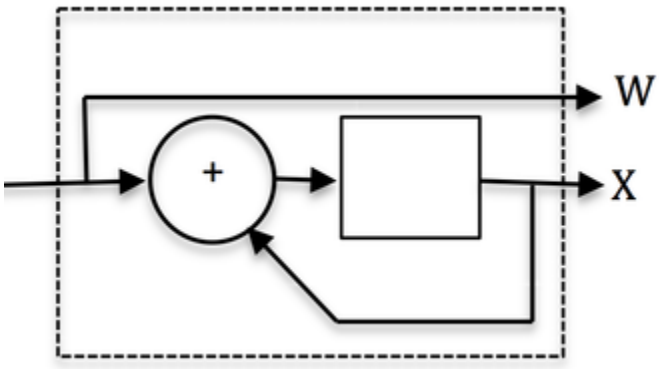
The example encoder is *systematic* because the input data is also used in the output symbols (Output 2). Codes with output symbols that do not include the input data are called *non-systematic*.

Recursive codes are typically systematic and, conversely, non-recursive codes are typically non-systematic. It isn't a strict requirement, but a common practice.

The example encoder in Img. 2. is an 8-state encoder because the 3 registers will create 8 possible encoder states (2^3). A corresponding decoder trellis will typically use 8 states as well.

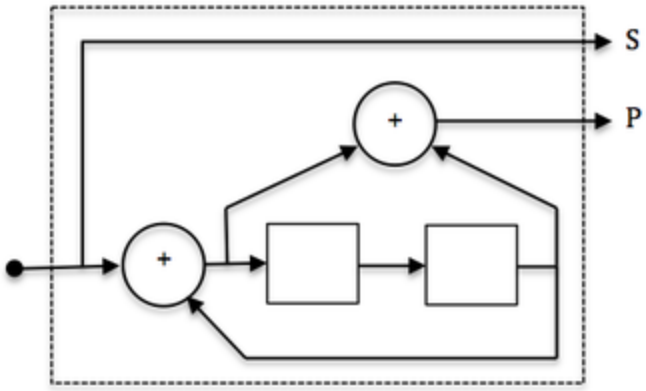
Recursive systematic convolutional (RSC) codes have become more popular due to their use in Turbo Codes. Recursive systematic codes are also referred to as pseudo-systematic codes.

Other RSC codes and example applications include:



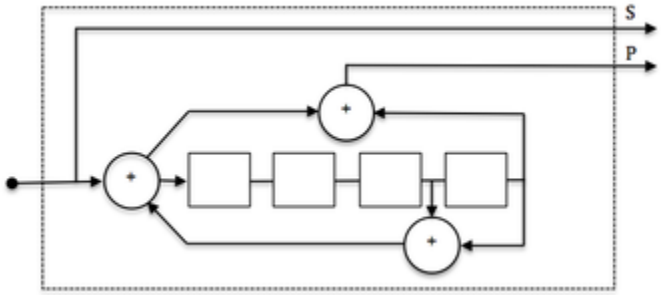
Img. 3. Two-state recursive systematic convolutional (RSC) code. Also called an 'accumulator.'

Useful for LDPC code implementation and as inner constituent code for serial concatenated convolutional codes (SCCC's).



Img. 4. Four-state recursive systematic convolutional (RSC) code.

Useful for SCCC's and multidimensional turbo codes.



Img. 5. Sixteen-state recursive systematic convolutional (RSC) code.

Useful as constituent code in low error rate turbo codes for applications such as satellite links. Also suitable as SCCC outer code.

Impulse response, transfer function, and constraint length

A convolutional encoder is called so because it performs a *convolution* of the input stream with the encoder's *impulse responses*:

$$y_i^j = \sum_{k=0}^{\infty} h_k^j x_{i-k} = (x * h^j)[i],$$

where x is an input sequence, y^j is a sequence from output j, h^j is an impulse response for output j and $*$ denotes convolution.

A convolutional encoder is a discrete linear time-invariant system. Every output of an encoder can be described by its own transfer function, which is closely related to the generator polynomial. An impulse response is connected with a transfer function through Z-transform.

Transfer functions for the first (non-recursive) encoder are:

- $H_1(z) = 1 + z^{-1} + z^{-2}$,
- $H_2(z) = z^{-1} + z^{-2}$,

- $H_3(z) = 1 + z^{-2}$.

Transfer functions for the second (recursive) encoder are:

- $H_1(z) = \frac{1 + z^{-1} + z^{-3}}{1 - z^{-2} - z^{-3}}$,
- $H_2(z) = 1$.

Define m by

$$m = \max_i \operatorname{polydeg}(H_i(1/z))$$

where, for any rational function $f(z) = P(z)/Q(z)$,

$$\operatorname{polydeg}(f) = \max(\deg(P), \deg(Q)) .$$

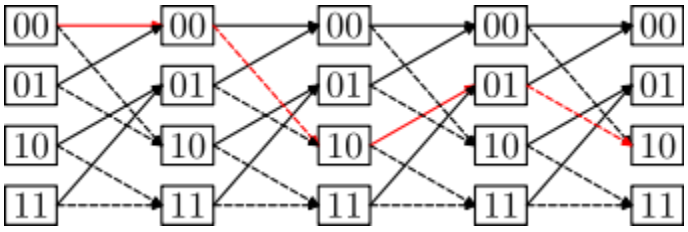
Then m is the maximum of the polynomial degrees of the $H_i(1/z)$, and the *constraint length* is defined as $K = m + 1$. For instance, in the first example the constraint length is 3, and in the second the constraint length is 4.

Trellis diagram

A convolutional encoder is a finite state machine. An encoder with *n* binary cells will have 2^n states.

Imagine that the encoder (shown on Img.1, above) has '1' in the left memory cell (m_0), and '0' in the right one (m_{-1}). (m_1 is not really a memory cell because it represents a current value). We will designate such a state as "10". According to an input bit the encoder at the next turn can convert either to the "01" state or the "11" state. One can see that not all transitions are possible for (e.g., a decoder can't convert from "10" state to "00" or even stay in "10" state).

All possible transitions can be shown as below:



Img.6. A trellis diagram for the encoder on Img.1. A path through the trellis is shown as a red line. The solid lines indicate transitions where a "0" is input and the dashed lines where a "1" is input.

An actual encoded sequence can be represented as a path on this graph. One valid path is shown in red as an example.

This diagram gives us an idea about *decoding*: if a received sequence doesn't fit this graph, then it was received with errors, and we must choose the nearest *correct* (fitting the graph) sequence. The real decoding algorithms exploit this idea.

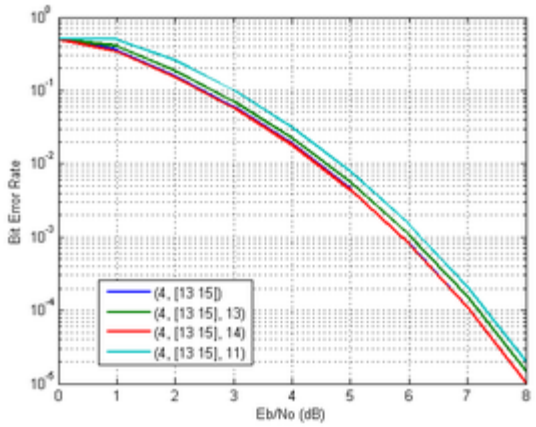
Free distance and error distribution

The **free distance**[7] (*d*) is the minimal Hamming distance between different encoded sequences. The *correcting capability* (*t*) of a convolutional code is the number of errors that can be corrected by the code. It can be calculated as

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor .$$

Since a convolutional code doesn't use blocks, processing instead a continuous bitstream, the value of *t* applies to a quantity of errors located relatively near to each other. That is, multiple groups of *t* errors can usually be fixed when they are relatively far apart.

Free distance can be interpreted as the minimal length of an erroneous "burst" at the output of a convolutional decoder. The fact that errors appear as "bursts" should be accounted for when designing a concatenated code with an inner convolutional code. The popular solution for this problem is to interleave data before convolutional encoding, so that the outer block (usually Reed–Solomon) code can correct most of the errors.

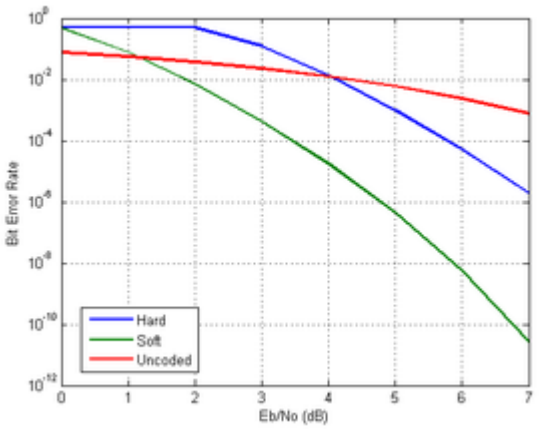


Theoretical bit-error rate curves of encoded QPSK (recursive and non-recursive, soft decision), additive white Gaussian noise channel. Curves are small distinguished due to approximately the same free distances and weights.

Decoding convolutional codes

Several algorithms exist for decoding convolutional codes. For relatively small values of *k*, the Viterbi algorithm is universally used as it provides maximum likelihood performance and is highly parallelizable. Viterbi decoders are thus easy to implement in VLSI hardware and in software on CPUs with SIMD instruction sets.

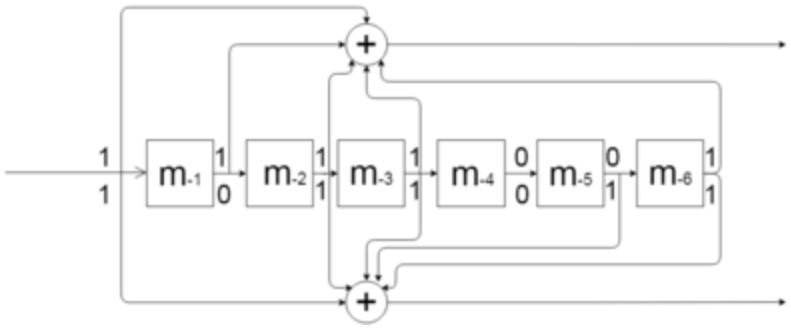
Longer constraint length codes are more practically decoded with any of several sequential decoding algorithms, of which the Fano algorithm is the best known. Unlike Viterbi decoding, sequential decoding is not maximum likelihood but its complexity increases only slightly with constraint length, allowing the use of strong, long-constraint-length codes. Such codes were used in the Pioneer program of the early 1970s to Jupiter and Saturn, but gave way to shorter, Viterbi-decoded codes, usually concatenated with large Reed–Solomon error correction codes that steepen the overall bit-error-rate curve and produce extremely low residual undetected error rates.



Theoretical bit-error rate curves for uncoded and coded QPSK, additive white Gaussian noise channel. Hard decision means that the decoder expects binary symbols (0's and 1's); Soft decision means that the decoder expects log-likelihood ratios[8][9].

Both Viterbi and sequential decoding algorithms return hard decisions: the bits that form the most likely codeword. An approximate confidence measure can be added to each bit by use of the Soft output Viterbi algorithm. Maximum a posteriori (MAP) soft decisions for each bit can be obtained by use of the BCJR algorithm.

Popular convolutional codes



Shift-register for the (7, [171, 133]) convolutional code polynomial. Branches: $h^1 = 171_o = [1111001]_b$, $h^2 = 133_o = [1011011]_b$. All of the math operations should be done by modulo 2.

use a *k* of 15 and a rate of 1/6; this code performs about 2 dB better than the simpler *k*=7 code at a cost of 256× in decoding complexity (compared to Voyager mission codes).

The convolutional code with a constraint length of 2 and a rate of 1/2 is used in GSM as an error correction technique.[11]

Punctured convolutional codes

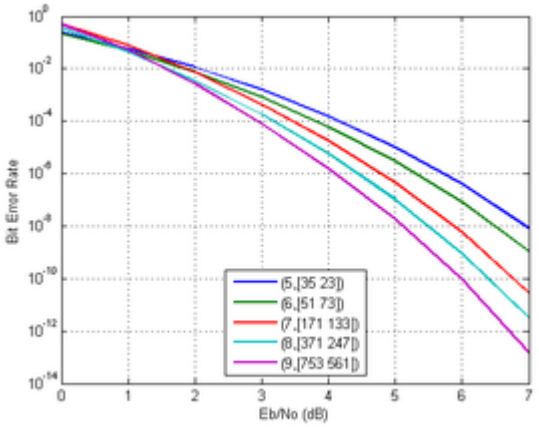
Convolutional code with any code rate can be designed based on polynomial selection[13]; however, in practice, a puncturing procedure is often used to achieve the required code rate. Puncturing is a technique used to make a *m*/*n* rate code from a "basic" low-rate (e.g., 1/*n*) code. It is achieved by deleting of some bits in the encoder output. Bits are deleted according to a *puncturing matrix*. The following puncturing matrices are the most frequently used:

Code rate	Puncturing matrix	Free distance (for NASA standard K=7 convolutional code)
1/2 (No perf.)	1 1	10
2/3	1 0 1 1	6
3/4	1 0 1 1 1 0	5
5/6	1 0 1 0 1 1 1 0 1 0	4
7/8	1 0 0 0 1 0 1 1 1 1 1 0 1 0	3

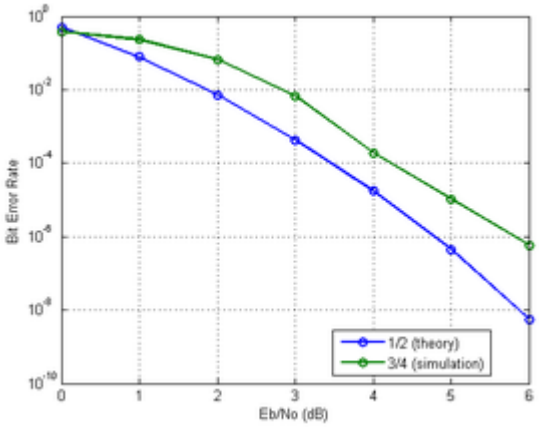
In fact, predefined convolutional codes structures obtained during scientific researches are used in the industry. This relates to the possibility to select catastrophic convolutional codes (causes larger number of errors).

An especially popular Viterbi-decoded convolutional code, used at least since the Voyager program has a constraint length *k* of 7 and a rate *r* of 1/2.[10]

Mars Pathfinder, Mars Exploration Rover and the Cassini probe to Saturn



Theoretical bit-error rate curves of encoded QPSK (soft decision), additive white Gaussian noise channel. Longer constraint lengths produce more powerful codes, but the complexity of the Viterbi algorithm increases exponentially with constraint lengths, limiting these more powerful codes to deep space missions where the extra performance is easily worth the increased decoder complexity.



Convolutional codes with 1/2 and 3/4 code rates (and constraint length 7, Soft decision, 4-QAM / QPSK / OQPSK).[12]

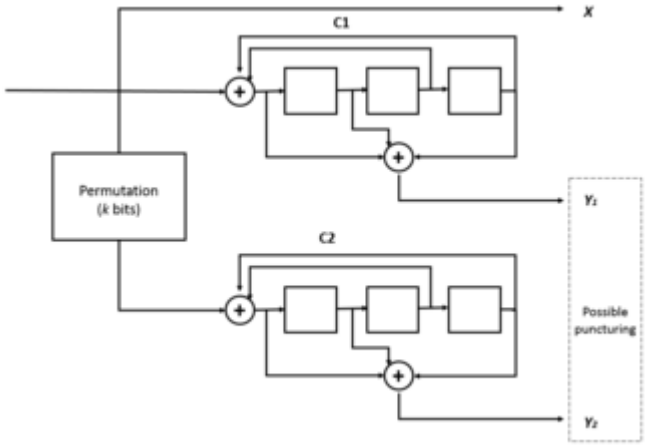
For example, if we want to make a code with rate 2/3 using the appropriate matrix from the above table, we should take a basic encoder output and transmit every first bit from the first branch and every bit from the second one. The specific order of transmission is defined by the respective communication standard.

Punctured convolutional codes are widely used in the satellite communications, for example, in INTELSAT systems and Digital Video Broadcasting.

Punctured convolutional codes are also called "perforated".

Turbo codes: replacing convolutional codes

Simple Viterbi-decoded convolutional codes are now giving way to turbo codes, a new class of iterated short convolutional codes that closely approach the theoretical limits imposed by Shannon's theorem with much less decoding complexity than the Viterbi algorithm on the long convolutional codes that would be required for the same performance. Concatenation with an outer algebraic code (e.g., Reed–Solomon) addresses the issue of error floors inherent to turbo code designs.



A turbo code with component codes 13, 15.[14] Turbo codes get their name because the decoder uses feedback, like a turbo engine. Permutation means the same as the interleaving. C1 and C2 are recursive convolutional codes. Recursive and non-recursive convolutional codes are not so much different in BER performance, however, recursive type of is implemented in Turbo convolutional codes due to better interleaving properties[15].

See also

- Quantum convolutional code

References

1. Benedetto, Sergio, and Guido Montorsi. "Role of recursive convolutional codes in turbo codes (https://ieeexplore.ieee.org/abstract/document/390945/)." Electronics Letters 31.11 (1995): 858-859.
2. Eberspächer J. et al. GSM-architecture, protocols and services. – John Wiley & Sons, 2008. - p.97
3. 3rd Generation Partnership Project (September 2012). "3GGP TS45.001: Technical Specification Group GSM/EDGE Radio Access Network; Physical layer on the radio path; General description". Retrieved 2013-07-20.
4. Halonen, Timo, Javier Romero, and Juan Melero, eds. GSM, GPRS and EDGE performance: evolution towards 3G/UMTS. John Wiley & Sons, 2004. - p. 430
5. Butman, S. A., L. J. Deutsch, and R. L. Miller. "Performance of concatenated codes for deep space missions." (http://tda.jpl.nasa.gov/progress_report/42-63/63H.PDF) The Telecommunications and Data Acquisition Progress Report 42-63, March–April 1981 (1981): 33-39.
6. Moon, Todd K. "Error correction coding." Mathematical Methods and Algorithms. Jhon Wiley and Son (2005). - p. 508

7. Moon, Todd K. "Error correction coding (https://leseprobe.buch.de/images-adb/7b/4f/7b4f94db-7c55-4836-9b61-2ff98cb242d9.pdf)." Mathematical Methods and Algorithms. Jhon Wiley and Son (2005).- p.508
8. LLR vs. Hard Decision Demodulation (https://www.mathworks.com/help/comm/examples/llr-vs-hard-decision-demodulation.html)
9. Estimate BER for Hard and Soft Decision Viterbi Decoding (https://www.mathworks.com/help/comm/ug/estimate-ber-for-hard-and-soft-decision-viterbi-decoding.html)
10. Butman, S. A., L. J. Deutsch, and R. L. Miller. "Performance of concatenated codes for deep space missions." (https://ipnpr.jpl.nasa.gov/progress_report/42-63/63H.PDF) The Telecommunications and Data Acquisition Progress Report 42-63, March–April 1981 (1981): 33-39.
11. Global system for mobile communications (GSM) (http://www.scholarpedia.org/article/Global_system_for_mobile_communications_(GSM))
12. Punctured Convolutional Coding (MathWorks) (https://ch.mathworks.com/help/comm/ug/punctured-convolutional-coding-1.html)
13. https://www.mathworks.com/help/comm/ref/poly2trellis.html
14. Turbo code (http://www.scholarpedia.org/article/Turbo_codes)
15. Benedetto, Sergio, and Guido Montorsi. "Role of recursive convolutional codes in turbo codes (https://ieeexplore.ieee.org/abstract/document/390945/)." Electronics Letters 31.11 (1995): 858-859.

External links

- The on-line textbook: Information Theory, Inference, and Learning Algorithms (http://www.inference.phy.cam.ac.uk/mackay/itila/), by David J.C. MacKay, discusses convolutional codes in Chapter 48.
- The Error Correcting Codes (ECC) Page (http://www.eccpage.com/)
- Matlab explanations (https://web.archive.org/web/20140522015414/http://www.mathworks.fr/fr/help/comm/convolutional-coding.html)
- Fundamentals of Convolutional Decoders for Better Digital Communications (http://www.ni.com/white-paper/14917/en/)
- Convolutional codes (MIT) (http://web.mit.edu/6.02/www/s2009/handouts/labs/lab5.shtml)
- Information Theory and Coding (TU Ilmenau) (http://www5.tu-ilmenau.de/nt/de/teachings/vorlesungen/itsc_master/folien/script.pdf), discusses convolutional codes on page 48.

Further reading

Publications

- Francis, Michael. "Viterbi Decoder Block Decoding-Trellis Termination and Tail Biting." Xilinx XAPP551 v2. 0, DD (2005): 1-21.
- Chen, Qingchun, Wai Ho Mow, and Pingzhi Fan. "Some new results on recursive convolutional codes and their applications." Information Theory Workshop, 2006. ITW'06 Chengdu. IEEE. IEEE, 2006.
- Fiebig, U-C., and Patrick Robertson. "Soft-decision and erasure decoding in fast frequency-hopping systems with convolutional, turbo, and Reed-Solomon codes." IEEE Transactions on Communications 47.11 (1999): 1646-1654.
- Bhaskar, Vidhyacharan, and Laurie L. Joiner. "Performance of punctured convolutional codes in asynchronous CDMA communications under perfect phase-tracking conditions." Computers

& Electrical Engineering 30.8 (2004): 573-592.
- Modestino, J., and Shou Mui. "Convolutional code performance in the Rician fading channel." IEEE Transactions on Communications 24.6 (1976): 592-606.
- Chen, Yuh-Long, and Che-Ho Wei. "Performance evaluation of convolutional codes with MPSK on Rician fading channels." IEE Proceedings F-Communications, Radar and Signal Processing. Vol. 134. No. 2. IET, 1987.
