Turbo-codes: the ultimate error control codes?

by A. Burr

Turbo-codes have attracted a great deal of interest since their discovery in 1993. This paper reviews the reasons for this, in particular their attainment of the ultimate limits of the capacity of a communication channel. The paper describes the two fundamental concepts on which they are based: concatenated coding and iterative decoding. This latter is the real 'turbo-principle', which is the real secret of their remarkable performance. The paper also reviews the direction of research in this area since 1993, and shows that, far from bringing coding research to an end, turbo-codes have led to a renaissance. In particular, other applications of the 'turbo-principle' have emerged, and these are discussed, along with the practical applications of turbo-codes that have appeared, from mobile radio to deep-space exploration.

1 Introduction

Turbo-codes promise the attainment of the 'Holy Grail' of communication theory, sought for nearly half a century: to achieve the ultimate limits of capacity of a communication channel. It is not surprising, therefore, that they have very rapidly moved from the research laboratories to find practical application throughout the world and beyond it. Following their announcement in 1993, they have found a very wide range of applications, mainly in wireless communications, ranging from the third generation mobile systems to deep-space exploration.

To understand the reason for the interest they have aroused, we must first review the limits to communication system capacity discovered more than 50 years ago by Claude Shannon (who also introduced the whole topic of error-control coding[1]) and this is done in the next section. We shall then introduce the two concepts that are the basis of turbo-codes: concatenated coding and iterative decoding. This latter is the real secret of turbo-codes, and also the basis of their name.

But will the discovery of turbo-codes put an end to the story of error control coding? It might be expected that the discovery of the ultimate codes would make any further work unnecessary, but in fact the reverse seems to be the case: it has resulted in a renaissance of coding research, both in universities and in industry world-wide. This has already led to the discovery of a range of related codes of equivalent power, and to other applications of the 'turbo' principle, which we will also consider at the end of this article.

2 The limits to capacity

In 1948 Claude Shannon was working at Bell Laboratories in the USA on the fundamental information transmission capacity of a communication channel[2]. (In doing so he also rigorously quantified the concept of 'information', and

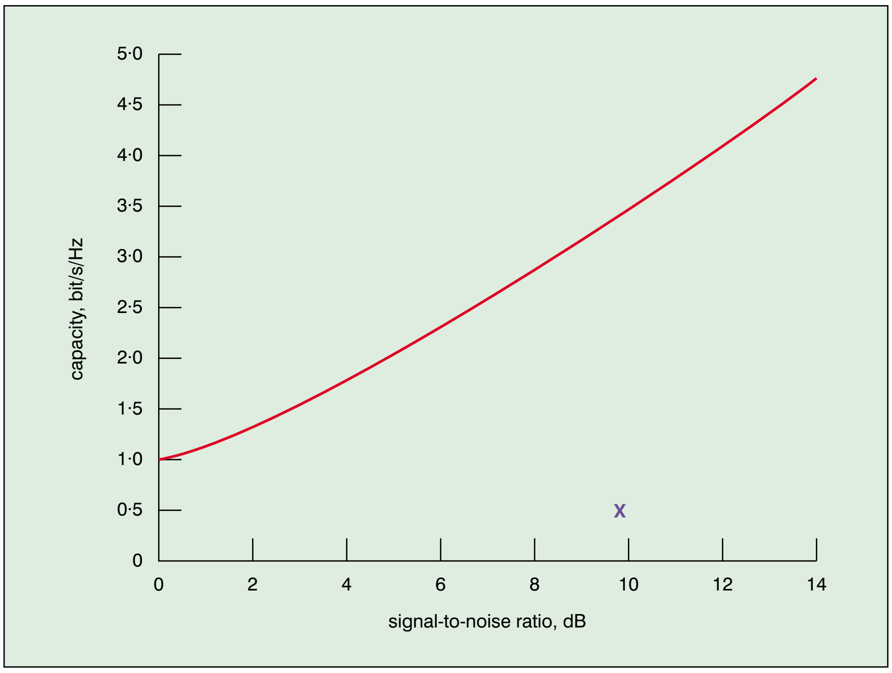

Fig. 1 Shannon bound on capacity per unit bandwidth, plotted against signal-to-noise ratio. 'x' indicates the capacity and SNR requirements of BPSK for a BER of 10^{-3}.

thus founded the discipline of information theory.) He showed that a communication channel is in principle able to transmit information with as few errors as we wish, even if the channel is subject to errors due to noise or interference, provided the capacity of the channel is not exceeded. This capacity depends on the signal-to-noise ratio (SNR)—the ratio of the signal power to noise power—as shown in Fig. 1.

Note that the capacity obtainable by conventional means is much less than this capacity limit. For example, the '**x**' mark on Fig. 1 shows the performance achieved on a radio system with a simple modulation scheme: binary phase-shift keying (BPSK). This is for a bit error ratio (BER) of 0·001, which is low enough for only a few services, such as speech, whereas the Shannon theory promises an arbitrarily low BER. Note that at the same SNR a capacity several times greater could be achieved; or equivalently that the same capacity could be achieved with a signal power many decibels lower. This highlighted the potential gains available and led to the quest for techniques that could achieve this capacity in practice.

Shannon did in fact also show in principle how to achieve capacity. The incoming data should be split into blocks containing as many bits as possible (say k bits). Each possible data block is then mapped to another block of n code symbols, called a *codeword*, which is transmitted over the channel. The set of codewords, and their mapping to data blocks, is called a *code*[3], or more specifically a *forward error correcting* (FEC) code. At the receiver there is a decoder, which must find the codeword that most closely resembles the word it receives, including the effects of noise and interference on the channel. The decoder is more likely to confuse codewords that resemble one another more closely: hence the power of the code to correct errors and overcome noise and interference depends on the degree of resemblance. This is characterised in terms of the minimum number of places in which any two codewords differ, called the *Hamming distance.*

Remarkably, Shannon showed that capacity could be achieved by a completely random code, that is a randomly chosen mapping set of codewords. The drawback is that this performance is approached only as k and n tend to infinity. Since the number of codewords then increases as 2^k, this makes the decoder's search for the closest codeword quite impractical, unless the code provides for a simpler search technique.

This motivated a quest which was to last for the next 45 years for practical codes and decoding techniques that could achieve Shannon's capacity bounds. Many good codes and decoders were found[3,4], but none that actually approached the limit. In fact it has been remarked that (in view of the fact that there must be infinitely many good random codes) 'all codes are good, except for the ones we can think of'[5]! It was also surmised that for practical purposes a capacity limit applied that was a few decibels lower than Shannon's, called the *cut-off rate bound.*

Hence there was a great deal of interest, not to mention scepticism, when results were announced in 1993[6] that significantly exceeded the cut-off rate bound, and approached within 0·7 dB of the Shannon bound. The work was by Berrou, Glavieux and Thitimajshima, a group previously unknown in the coding community, and was presented at the International Conference on Communications (ICC). In fact similar results had been submitted to ICC the previous year, but the paper had been rejected by the referees as too good to be true. However once published the results were very soon verified by many independent researchers.

Despite the suddenness with which they burst upon the scene, turbo-codes were in fact based on two previously known concepts, namely concatenated coding and iterative decoding, which we will now consider in more detail.

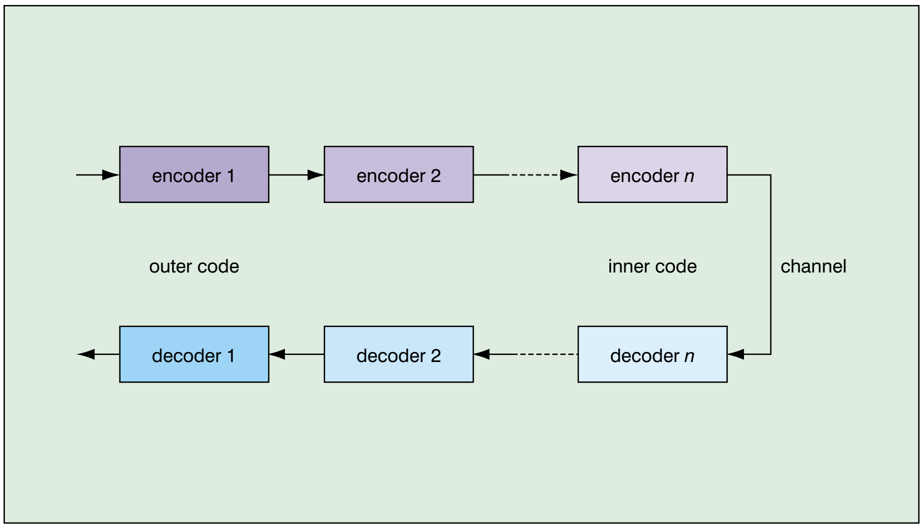

Fig. 2 Principle of concatenated codes

3 Concatenated codes

We have seen that the power of FEC codes increases with length k and approaches the Shannon bound only at very large k, but also that decoding complexity increases very rapidly with k. This suggests that it would be desirable to build a long, complex code out of much shorter *component codes*, which can be decoded much more easily. *Concatenation* provides a very straightforward means of achieving this (Fig. 2). The principle is to feed the output of one encoder (called the *outer* encoder) to the input of another encoder, and so on, as required. The final encoder before the channel is known as the *inner* encoder. The resulting composite code is clearly much more complex than any of the individual codes. However it can readily be decoded: we simply apply each of the component decoders in turn, from the inner to the outer.

This simple scheme suffers from a number of

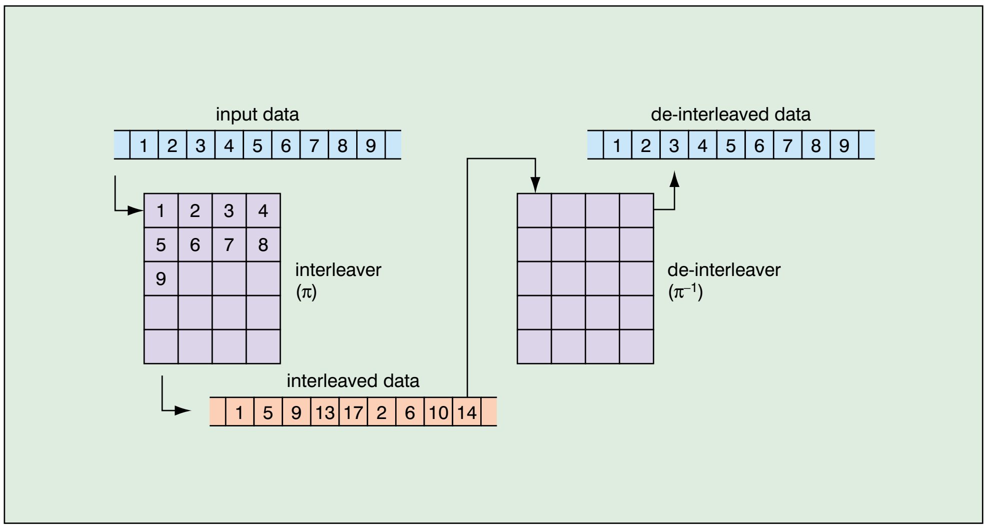

Fig. 3 Operation of interleaver and de-interleaver

drawbacks, the most significant of which is called *error propagation*. If a decoding error occurs in a codeword, it usually results in a number of data errors. When these are passed on to the next decoder they may overwhelm the ability of that code to correct the errors. The performance of the outer decoder might be improved if these errors were distributed between a number of separate codewords. This can be achieved using an *interleaver/ de-interleaver*. The simplest type of interleaver is illustrated in Fig. 3: much more complex versions will be encountered later.

This simple interleaver (sometimes known as a *rectangular* or *block* interleaver: we will use the former term) consists of a two-dimensional array, into which the data is read along its rows. Once the array is full, the data is read out by columns, thus permuting the order of the data. (Because it performs a permutation, an interleaver is commonly denoted by the Greek letter π, and its corresponding de-interleaver by π^{-1}.) The original order can then be restored by a corresponding de-interleaver: an array of the same dimensions in which the data is read in by columns and read out by rows.

This interleaver may be placed between the outer and inner encoders of a concatenated code that uses two component codes, and the de-interleaver between the inner and outer decoders in the receiver, as shown in Fig. 4. Then, provided the rows of the interleaver are at least as long as the outer codewords, and the columns at least as long as the inner data blocks, each data bit of an inner codeword falls into a different outer codeword. Hence, provided the outer code is able to correct at least one error, it can always cope with single decoding errors in the inner code.

Usually the block codes used in such a concatenated coding scheme are *systematic*: that is, the k data bits appear in the codeword, along with $n - k$ *parity* or *check* bits, which allow the data bits to be corrected if errors occur[3], making a codeword of length n. Now suppose the outer code has data length k_1 and code length n_1, while the inner code has data length k_2 and code length n_2, and the interleaver has dimension k_2 rows by n_1 columns. Then the parity and data bits may be arranged in an array as shown in Fig. 5. Part of this array (within the heavy line) is stored in the interleaver array: the rows contain codewords of the outer code. The parity of the inner code is then generated by the outer encoder as it encodes the data read out of the interleaver by columns. This includes the section of the array generated by encoding the parity of the outer code in the inner code, marked 'Checks on checks' in the figure. The columns of the array are thus codewords of the inner code. Observe that the composite code is much longer, and therefore potentially more powerful, than the component codes: it has data length $k_1 \times k_2$ and overall length $n_1 \times n_2$.

These codes have been well known for some time[7]: they are called *array* or *product codes* (because the concatenation is in the nature of a multiplicative process).

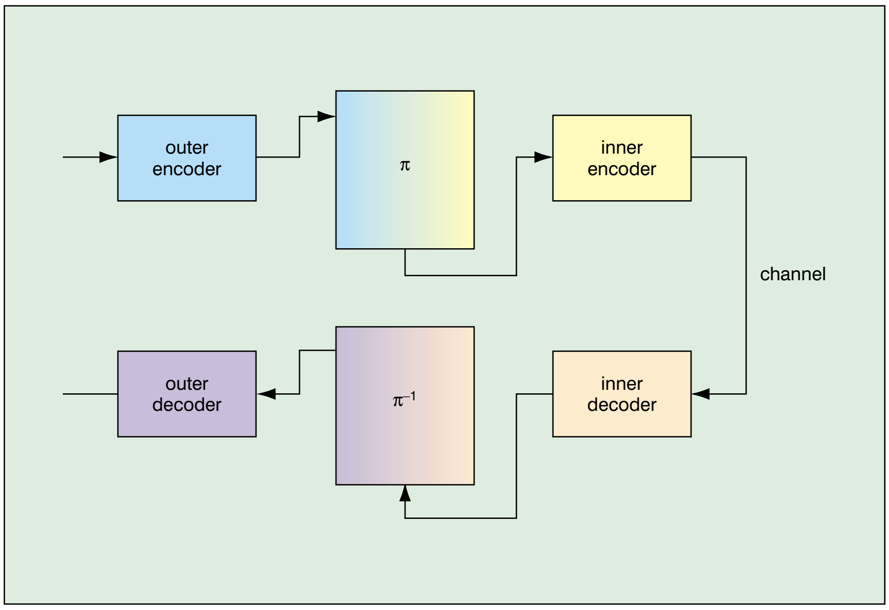

Fig. 4 Concatenated encoder and decoder with interleaver

The innovation brought by the advent of turbo-codes was the decoding technique: iterative decoding.

4 Iterative decoding: the 'turbo' principle

The conventional decoding technique for array codes is that shown in Fig. 4: the inner code is decoded first, then the outer. However, this may not always be as effective as we might hope.

Consider a received codeword array with the pattern of errors shown by the 'O's in Fig. 6. Suppose that both component codes are capable of correcting single errors only. As mentioned above, if there are more errors than this the decoder may actually introduce further errors into the decoded word. For the pattern shown this is the case for two of the column codewords, and errors might be added as indicated by 'X'. When this is applied to the outer (row) decoder some of the original errors may be corrected (indicated by a cross through the 'O'), but yet more errors may be inserted (marked with '+'). However, the original pattern would have been decoded correctly had it been applied to the row decoder first, since none of the rows contains more than one error.

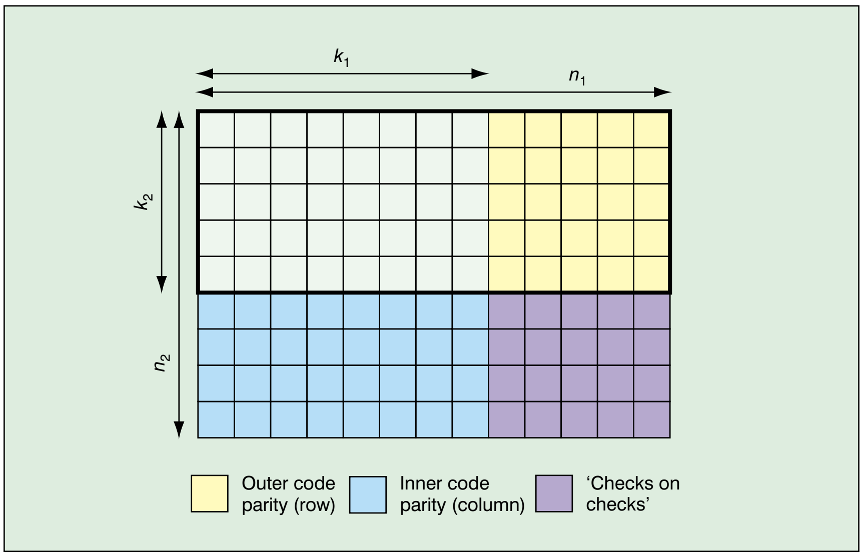

Fig. 5 Array for interleaved concatenated code

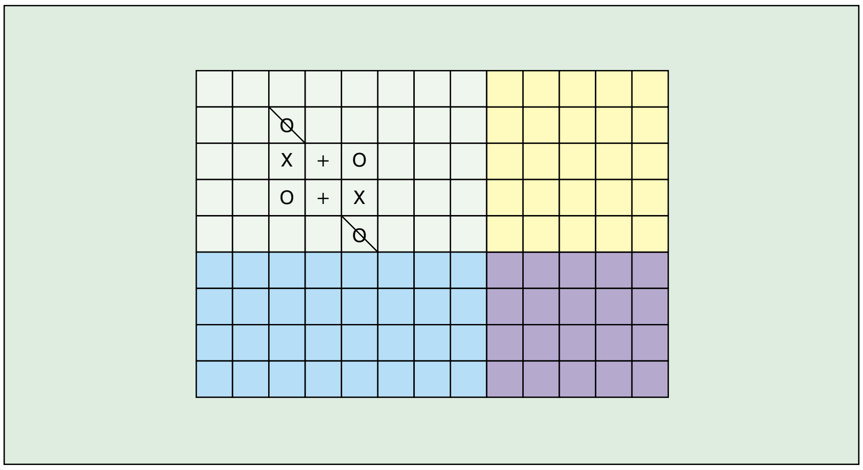

Fig. 6 Pattern of received errors ('O') in codeword array, with errors introduced by inner (column) decoder ('X') and outer (row) decoder ('+')

Note that if the output of the outer decoder were reapplied to the inner decoder it would detect that some errors remained, since the columns would not be codewords of the inner code. (A codeword of a single-error correcting code must contain either no errors or at least three.) This in fact is the basis of the iterative decoder: to reapply the decoded word not just to the inner code, but also to the outer, and repeat as many times as necessary. However, it is clear from the foregoing argument that this would be in danger of simply generating further errors. One further ingredient is required for the iterative decoder.

That ingredient is *soft-in, soft-out* (SISO) decoding. It is well known (see Section 5.6.2 of Reference 4) that the performance of a decoder is significantly enhanced if, in addition to the 'hard decision' made by the demodulator on the current symbol, some additional 'soft information' on the reliability of that decision is passed to the decoder. For example, if the received signal is close to a decision threshold (say between 0 and 1) in the demodulator, then that decision has low reliability, and the decoder should be able to change it when searching for the most probable codeword. Making use of this information in a conventional decoder, called *soft-decision decoding*, leads to a performance improvement of around 2 dB in most cases.

In the decoder of a concatenated code the output of one decoder provides the input to the next. Thus to make full use of soft-decision decoding requires a component decoder that generates 'soft information' as well as making use of it. This is the SISO decoder. Soft information usually takes the form of a *log-likelihood ratio* for each data bit. The likelihood ratio is the ratio of the probability that a given bit is '1' to the probability that it is '0'. If we take the logarithm of this, then its sign corresponds to the most probable hard decision on the bit (if it is positive, '1' is most likely; if negative, then '0'). The absolute magnitude is a measure of our certainty about this decision.

Subsequent decoders can then make use of this reliability information. It is likely that decoding errors will result in a smaller reliability measure than correct decoding. In the example this may enable the outer (row) decoder to correctly decode some of the errors resulting from the incorrect inner decoding. If not it may reduce the likelihood ratio of some, and a subsequent reapplication of the column decoder may correct more of the errors, and so on.

Note that the log-likelihood ratio exactly mirrors Shannon's quantitative measure of information content, mentioned above, in which the information content of a symbol is measured by the logarithm of its probability. Thus we can regard the log-likelihood ratio as a measure of the total information we have about a particular bit. In fact this information comes from several separate sources. Some comes from the received data bit itself: this is known as the *intrinsic information*. Information is also extracted by the two decoders from the other received bits of the row and the column codeword. When decoding one of these codes, the information from the other code is regarded as *extrinsic information*. It is this information that needs to be passed between decoders, since the intrinsic information is already available to the next decoder, and to pass it on would only dilute the extrinsic information.

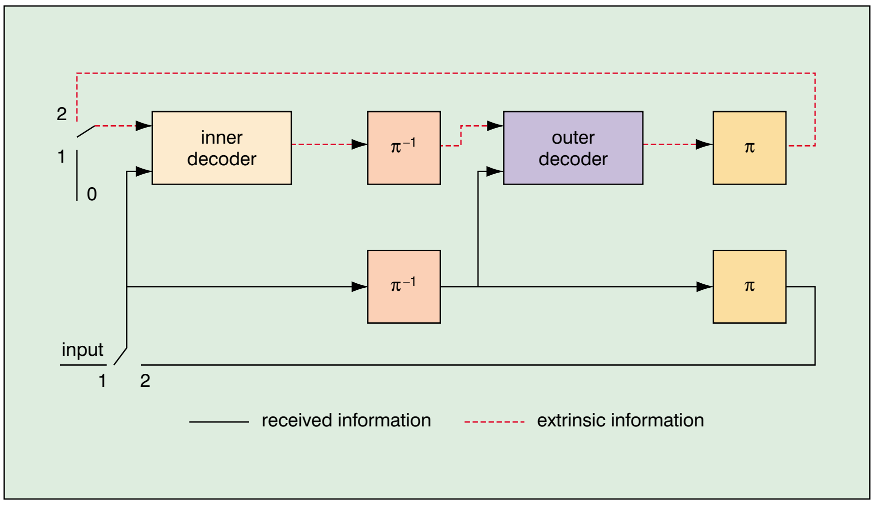

Fig. 7 Iterative decoder. The switches are in position 1 for the first iteration and in position 2 for subsequent iterations.

Hence the iterative decoder has the structure shown in Fig. 7, in which the intrinsic information has been separated from the extrinsic, so that the output of each decoder contains only extrinsic information to pass on to the next decoder. After the outer code has been decoded for the first time both the extrinsic information and the received data are passed back to the first decoder, re-interleaved back to the appropriate order for this decoder, and the whole process iterated again. It is this feedback that has given rise to the term 'turbo-code', since the original inventors likened the process to a turbo-charged engine, in which part of the power at the output is fed back to the input to boost the performance of the whole system. Thus the term 'turbo' should really be applied to the decoder structure rather than the codes themselves.

This structure assumes that the decoders operate much faster than the rate at which incoming data arrives, so that several iterations can be accommodated in the time between the arrivals of received data blocks. If this is not the case, the architecture may be replaced by a pipeline structure, in which data and extrinsic information are passed to a new set of decoders while the first one processes the next data block. In either structure at some point the decoder may be deemed to have converged to the optimum decoded word, at which point the combination of extrinsic and intrinsic information can be used to find the decoded data. Usually a fixed number of iterations is used—between 4 and 10, depending on the type of code and its length—but it is also possible to detect convergence and terminate the iterations at that point[8].

5 Parallel-concatenated recursive-systematic convolutional codes: turbo-codes

However turbo-codes are not in fact based on concatenated block codes of this sort (although we will come back to these product codes later). The turbo-codes invented by Berrou *et al.* should more formally be described as *parallel-concatenated recursive systematic convolutional codes.* (It is little surprise that the inventors' term 'turbo-codes' is much more popular!) We will now 'unpack' the meaning of these terms.

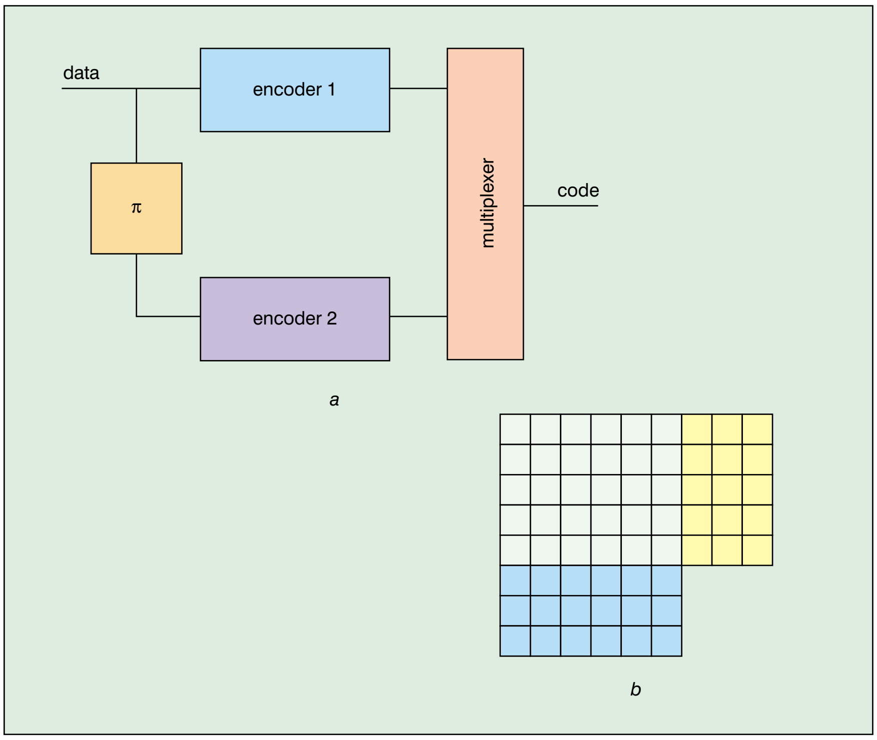

Fig. 8 Parallel concatenation: (*a*) encoder structure; (*b*) code array

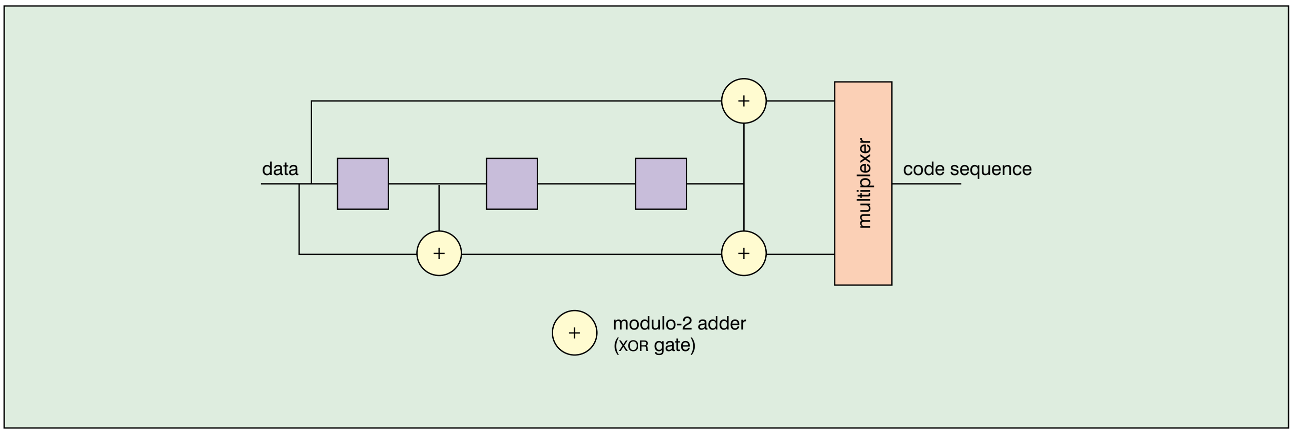

Fig. 9 Typical convolutional coder

Parallel-concatenated codes

The concatenated codes considered in Section 3 are more fully described as *serial-concatenated codes*, because the two encoders are connected in series. There is an alternative connection, called parallel concatenation, in which the same data is applied to two encoders in parallel, but with an interleaver between them, as shown in Fig. 8*a*. If systematic block codes and a rectangular interleaver are used, as in Section 3, but the systematic component of the second code output is not transmitted (since it is duplicated), then the code array is as shown in Fig. 8*b*. It is essentially the same as in Fig. 5, except that the 'checks on checks' are not present.

In turbo-codes the interleaver is not usually rectangular (i.e. not like that of Fig. 3), but for reasons that we will shortly examine it is *pseudorandom*, that is the data is read out in a predefined pseudorandom order. The design of interleaver is one of the key features of turbo-codes. Further, the encoders are not block codes, but convolutional codes.

Convolutional codes

Convolutional codes are fundamentally different from the block codes that we described in Section 2, and which were applied in the concatenated codes of Section 3. It is not possible to separate the codes into independent blocks. Instead each code bit depends on a certain number of previous data bits. They can be very conveniently encoded using a structure consisting of a shift register, a set of exclusive-OR (XOR) gates, and a multiplexer, as shown for a typical example in Fig. 9.

In these codes the concept of a codeword is replaced by that of a *code sequence*. Note, for example, that if a single data '1' is input (in a long sequence of data '0's) the result will be a sequence of code '0's and '1's as the '1' propagates along the shift register, returning to '0's once the '1' has passed through. The contents of the shift register define the *state* of the encoder: in this example it is non-zero while the '1' propagates through it, then returns to the zero state.

Parallel concatenation as illustrated in Fig. 8 clearly depends on using systematic codes. However the code of Fig. 9 is not systematic: the code sequence does not contain the data sequence. It could be made systematic by driving one of the inputs to the multiplexer directly from the data input. However it can be shown[9, p.312] that such codes are necessarily less powerful than non-systematic codes like that of Fig. 9. It is evident from the discussion of parallel concatenation above that a systematic code would be desirable. Fortunately there is a method of rearranging a non-systematic code in a systematic form that also has some other very desirable properties for turbo-codes.

Recursive-systematic coding

This rearrangement takes the form of introducing feedback to the encoder: for example the encoder of Fig. 9 becomes that of Fig. 10. It can quite easily be shown that the code generated by these two forms of encoder are equivalent (Section 7.1.3 of Reference 4) in that they

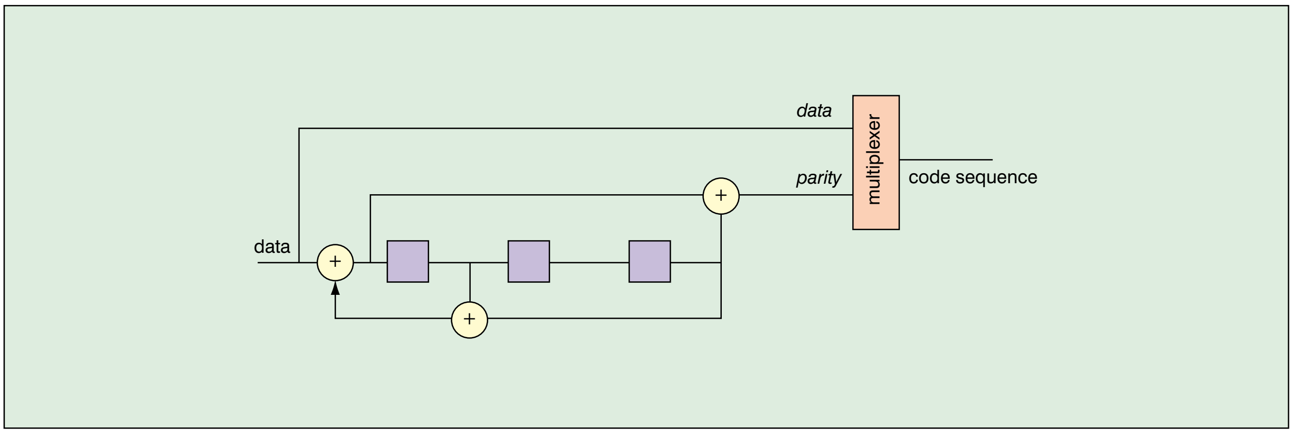

Fig. 10 Recursive-systematic form of the encoder of Fig. 9

contain the same set of code sequences: the difference lies in the mapping between data and code sequence. The code of Fig. 10 is clearly systematic, since one of the sequences fed to the multiplexer is the data stream. The other may be called the parity stream.

The behaviour of the two encoders with the same data sequence is nevertheless quite different. If a data sequence containing a single '1' is fed to the recursive-systematic encoder, because of the feedback the encoder will never return to the zero state but will continue indefinitely to produce a pseudorandom sequence of '1's and '0's. In fact only certain sequences, called *terminating sequences*, which must contain at least two '1's, will bring the encoder back to the zero state.

To see why this is a useful property in a parallel-concatenated code we must consider the minimum Hamming distance of the codes, mentioned in Section 2 above, where we noted that the larger the Hamming distance, the more powerful the code. For these codes, as for most codes, the minimum Hamming distance is in fact equal to the minimum number of '1's in any code sequence. Clearly a non-terminating data sequence, or one that terminates only after a long period, corresponds to a large Hamming distance. Now in a parallel-concatenated code the same data sequence is interleaved and applied to a second encoder (Fig. 8*a*). If a given data sequence happens to terminate the first encoder quickly, it is likely that once interleaved it will not terminate the second encoder, and thus will result in a large Hamming distance in at least one of the two encoders.

This is why the design of the interleaver is important. Data sequences that terminate both encoders quickly may readily be constructed for a rectangular interleaver. Moreover the regularity of its structure means that there are a large number of such sequences: the same pattern placed anywhere in the interleaver array will result in the same behaviour. A pseudorandom interleaver is preferable because even if (by chance) data sequences exist which result in a low overall Hamming distance, there will be very few of them, since the same sequence elsewhere in the input block will be interleaved differently.

Notice that if we simply place the recursive-systematic encoder of Fig. 10 in the parallel concatenated system of Fig. 8*a*, the resulting code will contain the systematic data twice. Hence the structure of Fig. 11 is used: one copy of the systematic data stream is multiplexed into the code stream along with the parity streams from each of the recursive encoders. Even this arrangement will result in a code of rate ⅓, a relatively low rate. This is commonly increased by *puncturing* the two parity streams. For example one bit might be deleted from each of the parity streams in turn, so that one parity bit remains for each data bit, resulting in a rate ½ code. Other rates are also possible by puncturing different proportions of the parity streams.

Fig. 12 shows the iterative decoder for these codes. It is fundamentally the same as the decoder of Fig. 7, but the separate parity streams are shown separately. If the code is punctured, 'dummy' parity symbols are reinserted in the parity streams to replace those that were deleted. These 'dummy' symbols take a level half way between the '1' level and the '0' level, and so when applied to the SISO decoders do not bias the decoding.

6 Performance and drawbacks of turbo-codes

The remarkable results that these techniques can produce are well illustrated in Fig. 13, which shows the performance of the 'original' turbo-code described by Berrou *et al.*[6], as determined by simulation. (The BER is plotted against bit energy to noise density ratio, which is directly proportional to SNR.) This shows that after one decoder iteration performance is good, but not outstanding. A further iteration results in a significant improvement, which continues with diminishing but still worthwhile returns. At 18 iterations the code achieves a BER better than 10^{-5} at a bit energy to noise density ratio of 0·7 dB—and for this code rate the Shannon bound is 0 dB. Thus was achieved a performance closer to the bound than anyone had previously imagined was possible.

Note, however, that this code uses an interleaver of length 65 536 bits (64 Kbits). Since this number of bits must inevitably be stored in the interleaver in the encoder and/or the decoder at any given time, this means that there is a *latency* at least this large. This in turn implies a delay through the encoder/decoder combination given by the product of the information bit period and the latency. Thus for an information rate of (say) 8 kbit/s (appropriate for speech transmission), there is a delay of 65 536/8 = 8192 ms, or more than 8 s. This delay is quite unacceptable in a telephone system, since it would be

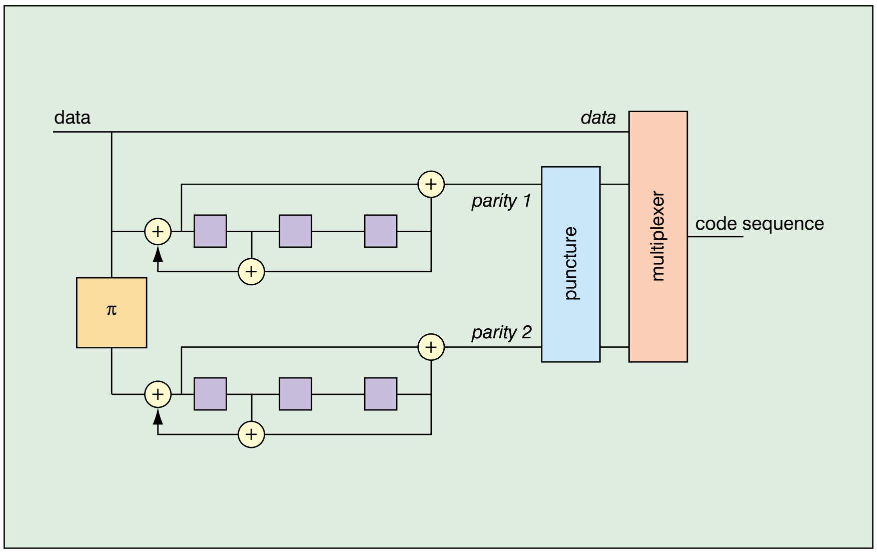

Fig. 11 Turbo-encoder with puncturing

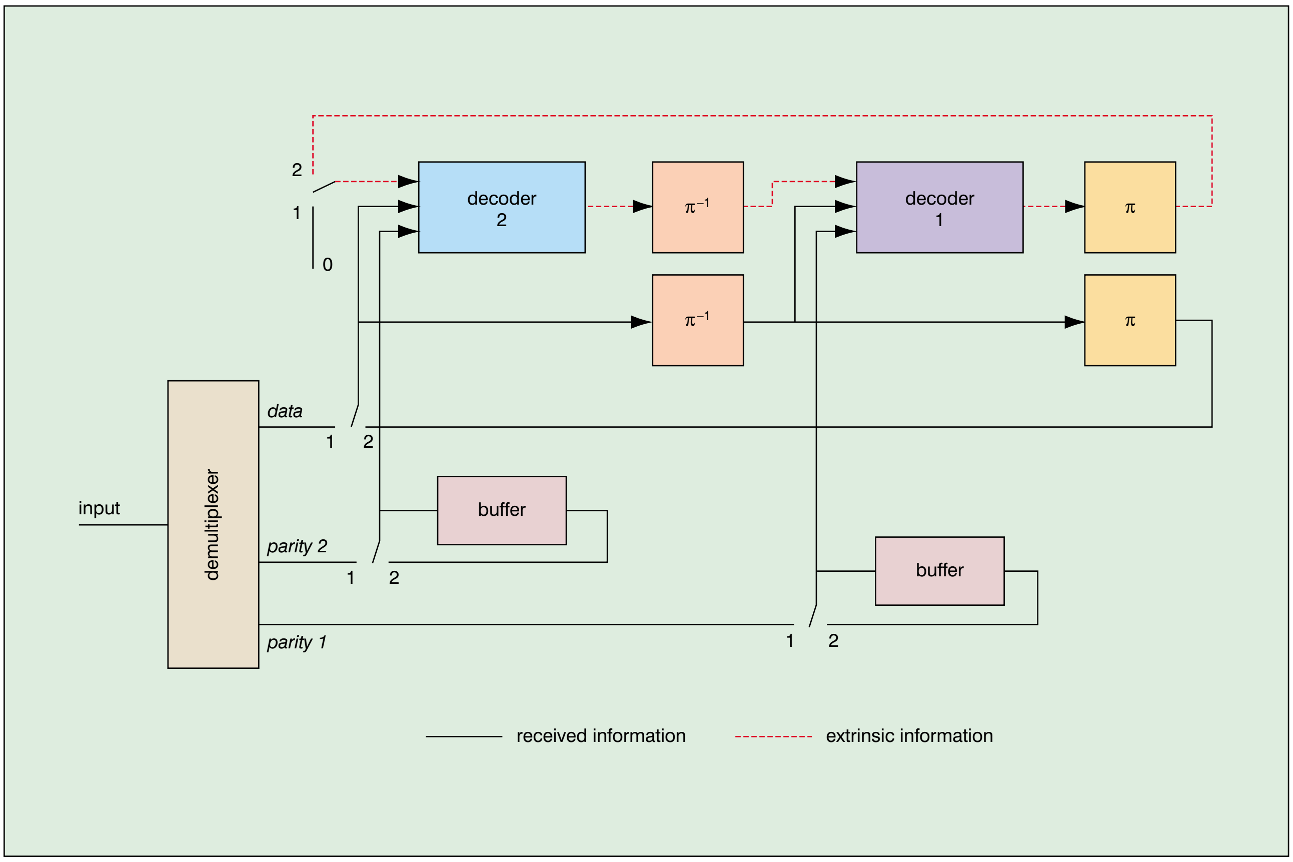

Fig. 12 Iterative turbo-decoder

highly disruptive of any conversation. In fact the delay limitation for speech services on wireless links is usually taken as 40 ms, which corresponds to a latency of 320. Thus if a turbo-code is to be used for speech services at this data rate the interleaver can be no larger than this.

Turbo-codes can be used with interleavers as short as this, but there are performance penalties to be paid. The first is inherent: the Shannon bound as plotted in Fig. 1 actually applies only if latency is unlimited. For limited latency Shannon subsequently showed[10] that capacity is further reduced, and at a latency of 320 this loss is equivalent to nearly 2 dB. In practice, however, a further problem arises: the so-called *error floor*.

In Fig. 13 we observe that the BER curve is very steep. However it has been noted for many turbo-codes that at a lower BER the curve flattens a little (although the term 'error floor' exaggerates this flattening). This occurs because although the effect of the pseudorandom interleaver means that code sequences at small Hamming distances are rare, they may nevertheless occur somewhere in the code. These then result in a small residual BER term which does not decrease so rapidly with signal-to-noise ratio. The probability of such sequences, and hence the level of this 'floor', decreases rapidly with increasing interleaver length. In Fig. 13 its level is well below the bottom of the graph; Fig. 14 shows the result for a much shorter code, in which a 'floor' appears at a BER around 10^{-6}. Note, however, that this code is still significantly better than other codes, such as convolutional codes, which can meet this delay requirement.

7 The ultimate codes? Other related codes

It might have been expected that the discovery of turbo-codes would have rendered unnecessary any further research in coding, but, as mentioned above, the reverse seems to be the case. The discovery has led to something of a renaissance in coding research, both theory and practice. Practically, of course, researchers and developers have been concerned to apply these codes in new applications, and indeed turbo-codes achieved practical application very soon after their discovery, as we shall see in the next section. Another important issue was the application of turbo-codes in more bandwidth-efficient coded modulation techniques[11], so as to fulfil the promise of increased spectrum efficiency—another area which has developed very rapidly (see Section 11.7 of Reference 4). However, the principles on which turbo-coding is based, and especially the principle of iterative decoding, have proved to be very fruitful in the development of other new codes, related to but distinct from the 'classical' turbo-codes of Berrou and Glavieux.

The most obvious of these, perhaps, are the serial-concatenated block codes, or product codes, as described in Section 3 above. The codes themselves, as we have said, were already well known, but after the announcement of turbo-codes it was soon realised that the iterative decoding technique could be used here, too, giving results not much poorer than turbo-codes. This development was termed *turbo-product codes*[12], although the innovation was not actually in the codes themselves but in the decoder. It is easy to show that for these codes there are no low-weight (small Hamming distance) codewords, and thus there is no error floor. This means that they may be better suited to applications where latency (and hence interleaver size) is limited but an error floor would be unacceptable because of the BER requirement. They also have the advantage that high-rate block codes exist, and thus it is easier to implement code rates near to unity: puncturing is not required. Further, block codes have decoders that can operate at high speeds (although they need to be adapted to allow soft input and output), which has led to ASIC decoders for these codes that can operate at very high data rates[13].

An obvious generalisation of product codes would be to use an array of more than two dimensions (more than two component codes). In this way the set of parity bit calculations involving each bit of the data becomes more and more complex. Such codes turn out to be special cases of another type of code that has been known for many years: the *low-density parity check* (LDPC) *codes*, or *Gallagher codes*[14], which date back to the 1960s. Soon after the discovery of turbo-codes it was shown[15] that these could also be decoded using a version of the iterative decoding algorithm. Although they do not approach the Shannon bound quite so closely as turbo-codes, these codes also do not exhibit an error floor. Another related code recently discovered is the *repeat-accumulate code*[16], which concatenates two codes so simple as to be almost trivial, to yield a very good overall performance.

Thus we now have available a plethora of code types, all capable of performing very close to the Shannon bound, with slightly different characteristics that suit them to different applications. We have also advanced in our understanding of why turbo-codes work so well, and in particular of the operation of the iterative decoder, with the introduction of tools such as the factor graph[17] and the extrinsic information transfer chart[18], which may well lead to further practical developments.

8 Applications of turbo-codes and of the ‘turbo’ principle

Moreover many applications of the ‘turbo’ principle, which we have already identified with the iterative approach, have emerged that are not exclusively concerned with turbo-codes. It turns out that iterative processing can be applied in many other situations where several processes are combined in the manner of a concatenated encoder. This is in addition to the many practical applications of turbo-coding itself, which we will now consider.

The very first such application was (literally!) the most far-reaching: deep-space exploration. FEC coding is essential to maintain communication with spacecraft exploring the solar system, because over the vast distances involved signal power is at a premium. An improvement of a small fraction of a decibel can make a

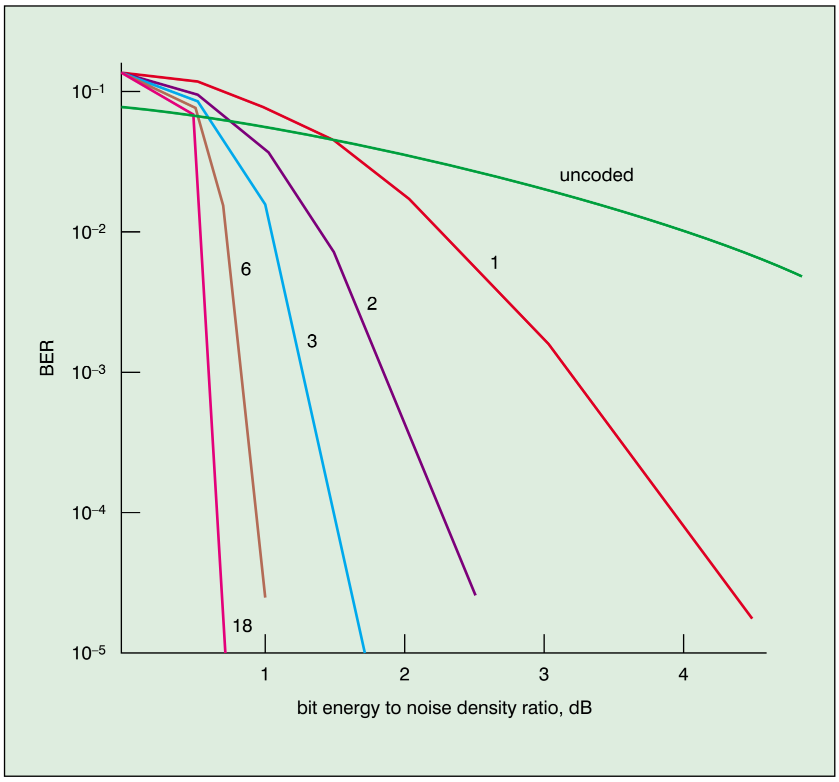

Fig. 13 Simulated performance of Berrou *et al.* code (rate ½, 64 Kbit interleaver, 16-state component codes), with various numbers of decoder iterations

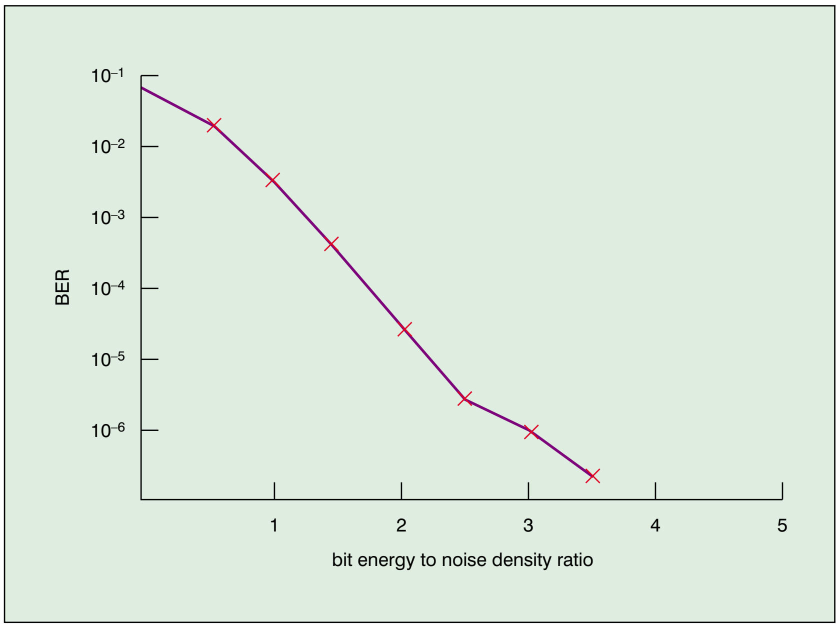

Fig. 14 Simulated BER for a rate ⅓, length 256 turbo-code, with 8 decoder iterations

Fig. 15 Surface of Mars seen from the Mars Rover (Courtesy of NASA/JPL/Caltech)

difference of millions of miles to the operational range of a mission. The Jet Propulsion Laboratory (JPL), which carries out research for NASA, was among the first to realise the potential of turbo-codes, and as a result turbo-codes were used in the Pathfinder mission of 1997 to transmit back to Earth the photographs of the Martian surface taken by the Mars Rover (Fig. 15).

However, of more practical significance to most of us is likely to be their application to mobile communications. Turbo-codes are one of the options for FEC coding in the UMTS third generation mobile radio standard[19]. A great deal of development has been carried out here, especially on the design of interleavers of different lengths, for application both to speech services, where latency must be minimised, and to data services that must provide very low BER.

Turbo-codes were discovered just too late to be incorporated in the terrestrial digital video broadcast (DVB-T) standard, which was finalised in the early 1990s, but they have recently been incorporated into the standard which will incorporate a return channel in digital broadcast systems.

All these applications will require practical implementations of the turbo-decoder. Although the iterative decoder makes decoding computationally feasible (by breaking it down into decoding operations for much simpler codes), it is still computationally complex, especially if many iterations are required. The algorithm can be implemented on DSP processors, but the computational load required means that such a decoder, using current DSP devices, could not operate at data rates higher than a few tens of kilobits per second. A more promising approach is custom-designed ASIC devices, or field-programmable gate arrays (FPGAs). For example, FPGA cores are now available which can operate at 90 Mbit/s[20]. Even higher speed decoders exist for turbo-product codes, because their component block codes permit very high speed decoding: a rate of 500 Mbit/s is claimed[13]. These speeds mean that computational complexity need not limit the maximum data rate achievable in turbo-coded wireless systems.

There are many processes in a wireless communications receiver which need to be performed jointly for best results, but for which the receiver complexity required would be excessive. In particular for an FEC-coded system processes like synchronisation and equalisation must be performed jointly with decoding for the best results. Synchronisation, for example, which involves carrier phase and symbol timing estimation, becomes considerably more difficult in a coded system, if performed separately before decoding, because the coding gain and the higher coded symbol rate mean that the synchroniser must work at a significantly lower signal-to-noise ratio. On the other hand simultaneous decoding and synchronisation becomes extremely complex, because the additional degrees of freedom introduced by synchronisation errors greatly increase the size of the space that must be searched to find the optimum decoding solution.

At this point we may invoke the 'turbo-principle' (even if we are not using turbo-codes), which allows us to perform synchronisation and decoding separately and still obtain the same overall performance as joint decoding and synchronisation. Fig. 16 illustrates joint carrier recovery and decoding (assuming that symbol timing is known). Initially, conventional, non-data-aided carrier phase estimation is performed. The result is used to decode the data. The decoded data is then fed back and used to improve the carrier estimate, using data-aided carrier recovery techniques, which can significantly improve the carrier phase estimate. This is then used to improve the decoding result, and so on. If the decoder outputs 'soft information', as the turbo-decoder does, this can be used to further improve the process by allowing the carrier phase estimator to make use of reliability information.

The same principle can be used to combine decoding with many similar processes, including symbol timing recovery, equalisation, multi-user detection in CDMA systems, and channel

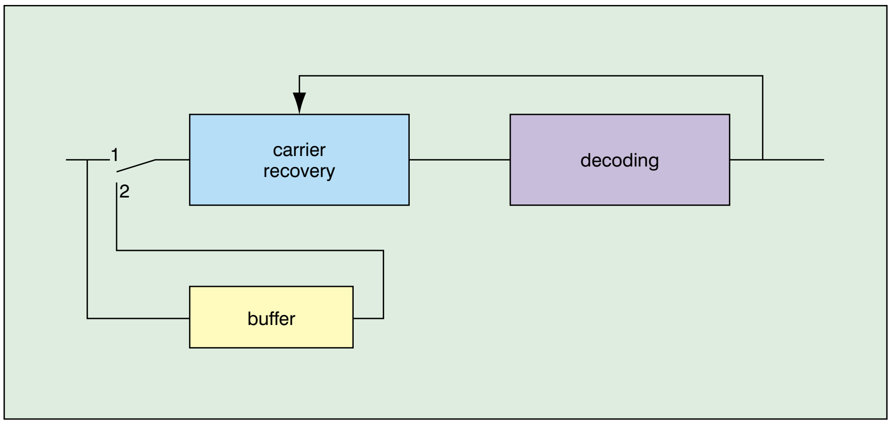

Fig. 16 Iterative carrier recovery and decoding. The switch takes position 1 for the first iteration and position 2 for subsequent iterations.

estimation. It can be integrated particularly well with turbo-decoding, since soft information is already available and very little additional computational overhead is then involved, but it will also work with most other types of code.

9 Conclusions

As we have seen, the reason turbo-codes have attracted so much attention in the last few years is that they represent the fulfilment of a quest, which lasted nearly 50 years, for a practical means of attaining the Shannon capacity bounds for a communication channel. We have reviewed the basic principles of turbo-codes, namely concatenated coding and iterative decoding, showing how it is that they achieve such remarkable performance. We have 'unpacked' the more formal description of turbo-codes as 'parallel-concatenated recursive-systematic convolutional codes'.

But are these the 'ultimate' error correction codes? Will they bring to an end the development of coding theory? The answer appears to be 'no': on the contrary, they have led to something of a renaissance of the subject. These principles of concatenated coding and (especially) iterative decoding have shown themselves to be very fruitful in producing further new codes, or at least new decoding techniques for old codes. Hence we have the 'turbo-product codes', in reality simply a new decoding approach for the well-known product codes, and renewed interest in Gallagher, or LDPC, codes, which date back to the 1960s, as well as the repeat-accumulate codes and many others. Applications of turbo-codes have also proliferated, from mobile 'phones to the further reaches of the solar system. The principles of turbo decoding can also be applied to many other processes, including equalisation and synchronisation.

References

1 Claude Shannon recently died. His obituaries give some interesting information on his life. See http://www.itsoc.org/shannon and http://www.thetimes.co.uk/article/0,,60-97442,00.html

2 SHANNON, C. E.: 'A mathematical theory of communication', *Bell Syst. Tech. J.*, July and October 1948, **27**, pp.379–423 and 623–656, obtainable at http://galaxy.ucsd.edu/new/external/shannon.pdf

3 FARRELL, P. G.: 'Coding as a cure for communications calamities—the successes and failures of error control', *Electron. Commun. Eng. J.*, 1990, **2**, (6), pp.213–220

4 BURR, A. G.: 'Modulation and coding for wireless communications' (Prentice-Hall, 2001)

5 BATTAIL, G.: 'We can think of good codes, and even decode them'. Eurocode'92, Udine, Italy, 26th–30th October 1992. Also CAMION, P., CHARPIN, P., and HARARI, S. (Eds.): 'CISM courses and lectures, no. 339' (Springer, 1993), pp.353–358

6 BERROU, C., GLAVIEUX, A., and THITIMAJSHIMA, P.: 'Near Shannon limit error-correcting coding: turbo codes'. Proc. IEEE Int. Conf. Commun., Geneva, Switzerland, 1993, pp.1064–1070

7 ELIAS, P.: 'Error-free coding', *IRE Trans. Inf. Theory*, September 1954, **PGIT-4**, pp.29–37

Alister Burr received a BSc degree from the University of Southampton in 1979, and a PhD from the University of Bristol in 1984. After working at Thorn-EMI Central Research Laboratories, he became a Lecturer at the Department of Electronics, University of York, where he is now Professor of Communications. His research is mainly in wireless communications, especially modulation and coding, including turbo-codes and, more recently, space-time codes and MIMO techniques. He is an IEE Member and a past Chair of the IEE Professional Group on Radiocommunications. He is currently Chair of Working Group 1 (Radio Systems Aspects) of COST 273: a European collaborative research project 'Towards mobile multimedia networks'.

Address: Department of Electronics, The University of York, York YO10 5DD, UK.
E-mail: alister@ohm.york.ac.uk

8 SCHURGERS, C., VAN DER PERRE, L., ENGELS, M., and DE MAN, H.: 'Adaptive turbo decoding for indoor wireless communication'. Proc. ISSSE'98, 1998 URSI Int. Symposium on Signals, Systems and Electronics, pp.107–111

9 LIN, S., and COSTELLO, D. J.: 'Error control coding: fundamentals and applications' (Prentice-Hall, 1983)

10 SHANNON, C. E.: 'Probability of error for optimal codes in a Gaussian channel', *Bell Syst. Tech. J.*, May 1959, **38**, pp.611–656

11 BURR, A. G.: 'Block versus trellis: an introduction to coded modulation', *Electron. Commun. Eng. J.*, August 1993, **5**, (4), pp.240–248

12 PYNDIAH, R.: 'Near optimum decoding of product codes: block turbo codes', *IEEE Trans. Commun.*, August 1998, **46**, (8), pp.1003–1010

13 For example, several decoders are supplied by Advanced Hardware Architectures—see http://www.aha.com/technology/showProdType.asp?iId=12

14 GALLAGHER, R. G.: 'Low density parity check codes', *IRE Trans. Inf. Theory*, January 1962, **IT-8**, pp.21–28

15 MACKAY, D. J. C.: 'Good error-correcting codes based on very sparse matrices', *IEEE Trans. Inf. Theory*, March 1999, **IT-45**, (2), pp.399–431

16 JIN, H., and MCELIECE, R. J.: 'RA codes achieve AWGN channel capacity', *in* FOSSIER, M., IMAI, H., LIN, S., and POLI, A. (Eds.): 'Applied algebra, algebraic algorithms and error-correcting codes. 13th International Symposium, AAECC-13, Honolulu, Hawaii, November 1999, Proceedings' (Lecture Notes in Computer Science 1719) (Springer-Verlag, 1999), pp.10–18

17 KSCHISCHANG, F., FREY, B., and LOELIGER, H-A.: 'Factor graphs and the sum-product algorithm', *IEEE Trans. Inf. Theory*, February 2001, **IT-47**, (2), pp.498–519

18 TEN BRINK, S.: 'Convergence of iterative decoding', *Electron. Lett.*, 13th May 1999, **35**, (10), pp.806–808

19 Details of the standard may be found at http://www.3gpp.org

20 For example from Small World Communications, see http://www.sworld.com.au/

First received 23rd April and in final form 10th May 2001.