Eur. Phys. J. B **18**, 107–119 (2000)

THE EUROPEAN PHYSICAL JOURNAL B

EDP Sciences

The statistical mechanics of turbo codes

A. Montanari[1,a] and N. Sourlas[2]

[1] Scuola Normale Superiore and INFN – Sezione di Pisa, 56100 Pisa, Italy
[2] Laboratoire de Physique Théorique de l'École Normale Supérieure[b], 24 rue Lhomond, 75231 Paris Cedex 05, France

Received 14 March 2000 and Received in final form 17 July 2000

Abstract. The "turbo codes", recently proposed by Berrou *et al.* [1] are written as a disordered spin Hamiltonian. It is shown that there exists a threshold Θ such that for signal to noise ratios $1/w^2 > \Theta$ the error probability per bit vanishes in the thermodynamic limit, *i.e.* the limit of infinitely long sequences. The value of the threshold has been computed for two particular turbo codes. It is found that it depends on the code. These results are compared with numerical simulations.

PACS. 75.10.Hk Classical spin models – 75.10.Nr Spin-glass and other random models – 89.70.+c Information science

1 Introduction

The recent invention of "turbo codes" by Berrou and Glavieux [1] is considered a major breakthrough in communications. For the first time one can communicate almost error-free for signal to noise ratios very close to the theoretical bounds of information theory. Turbo codes are quickly becoming the new standard for error correcting codes in digital communications. The invention of turbo codes and their iterative decoding algorithm was empirical. There is no theoretical understanding of why they are so successful. The decoding algorithm is thought to be an approximate algorithm. We think that turbo codes are interesting, even outside the context of communication theory, because they provide a non trivial example of a disordered system which can be studied numerically with a fast algorithm.

In this paper we will study turbo codes and turbo decoding using the tools of statistical mechanics of disordered systems. One of us has already shown in the past [2–5] that there is a mathematical equivalence between error correcting codes and theoretical models of spin glasses. In particular the logarithm of the probability for any given signal, conditional on the communication channel output, has the form of a spin glass Hamiltonian. We will construct the Hamiltonian which corresponds to the turbo codes and study its properties. This will clarify why they are so successful. In particular we will show that there exists a threshold Θ such that for signal to noise ratios $1/w^2 > \Theta$ the average error probability per bit $\overline{P_{\rm e}}$ vanishes in the thermodynamic limit, *i.e.* the limit of infinitely long sequences. In $\overline{P_{\rm e}}$ the average is taken over a large class of turbo codes (see later) and over "channel" noise. The rate of these codes is finite. The value of the threshold has been computed for two particular turbo codes. It was found that it depends on the code. We also compare these results with numerical simulation. Our results are typical of the statistical mechanics approach: we study only the average performance of turbo codes, not the performance of any particular one. Furthermore there exist "very few" particular codes performing "much worse" than the average.

Let us first briefly remind the connection between error-correction codes and spin-glass models. In the mathematical theory of communication both the production of information and its transmission are considered as probabilistic events. A source is producing information messages according to a certain probability distribution. For sake of simplicity we shall consider a "flat" probability distribution, *i.e.* any message is produced with the same probability. Messages of length N are sequences of N symbols or "letters of an alphabet" $a_1, a_2, \cdots, a_N$. We will assume for simplicity a binary alphabet, *i.e.* $a_i = 0$ or 1 with equal probability. Instead of a_i we can equally well use Ising spins $\sigma_i = (-1)^{a_i} = \pm 1$. A message of length N will correspond to a determined sequence of N spins $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_N\}$. The messages are sent through a noisy transmission channel. If a $\sigma = \pm 1$ is sent through the transmission channel, because of the noise the output will be a real number $\sigma^{\rm out}$, in general different from σ. Let us call $Q(\boldsymbol{\sigma}^{\rm out}|\boldsymbol{\sigma})\mathrm{d}\boldsymbol{\sigma}^{\rm out}$ the probability for the transmission channel's output to be between $\boldsymbol{\sigma}^{\rm out}$ and $\boldsymbol{\sigma}^{\rm out} + \mathrm{d}\boldsymbol{\sigma}^{\rm out}$, when the input was $\boldsymbol{\sigma}$. The probability distribution $Q(\boldsymbol{\sigma}^{\rm out}|\boldsymbol{\sigma})$ is supposed to be known. For reasons of simplicity, we assume that the noise is independent for any pair of bits ("memoryless channel"), *i.e.*

$$Q(\boldsymbol{\sigma}^{\rm out}|\boldsymbol{\sigma}) = \prod_i Q(\sigma_i^{\rm out}|\sigma_i). \tag{1.1}$$

[a] e-mail: montanar@cibs.sns.it
[b] UMR 8549, Unité Mixte de Recherche du Centre National de la Recherche Scientifique et de l'École Normale Supérieure.

We shall here, for simplicity, consider only the case of the so called "Gaussian channel" defined by

$$Q_{\rm g}(\sigma^{\rm out}|\sigma) \equiv \frac{1}{\sqrt{2\pi w^2}} \exp\left\{-\frac{(\sigma^{\rm out}-\sigma)^2}{2w^2}\right\}. \quad (1.2)$$

Shannon calculated the channels capacity $\mathcal{C}$, *i.e.* the maximum information per use of the channel that can be transmitted. For a Gaussian channel he obtained:

$$\mathcal{C}_{\rm g} = \frac{1}{2}\log_2(1+\frac{1}{w^2})\cdot \quad (1.3)$$

We will consider only the case of "binary input channels", *i.e.* the case when all channel inputs are ± 1. In this case the capacity is reduced to

$$\mathcal{C}'_{\rm g} = \int {\rm d}\sigma\, Q_{\rm g}(\sigma|+1)\log_2 Q_{\rm g}(\sigma|+1) - \int {\rm d}\sigma\, Q_{\rm g}(\sigma)\log_2 Q_{\rm g}(\sigma), \quad (1.4)$$

$$Q_{\rm g}(\sigma) \equiv \frac{1}{2}[Q_{\rm g}(\sigma|+1)+Q_{\rm g}(\sigma|-1)]. \quad (1.5)$$

Under the above assumptions, communication is a statistical inference problem. Given the transmission channel's output and the statistical properties of the source and of the channel, one has to infer what message was sent. In order to reduce communication errors, one may introduce (deterministic) redundancy into the message ("channel encoding") and use this redundancy to infer the message sent through the channel ("decoding"). The algorithms which transform the source outputs to redundant messages are called error-correcting codes.

A large class of error correcting codes can be described as follows. Instead of sending the N original bits σ_i, often called the "letters of the source word", one sends M bits $\{J_1^{\rm in},\ldots,J_M^{\rm in}\}$, with $M>N$, called the "letters of the code word", constructed in the following way

$$J_k^{\rm in} = \sum_{i_1\ldots i_{l_k}=1}^{N} C^{(k)}_{i_1\ldots i_{l_k}}\sigma_{i_1}\cdots\sigma_{i_{l_k}}; \quad k=1,\ldots,M, \quad (1.6)$$

where the "connectivity" matrix $C^{(k)}_{i_1\ldots i_{l_k}}$ has elements zero or one. For any k, all the $C^{(k)}_{i_1\ldots i_{l_k}}$ except from one are equal to zero. As a consequence the $J_k^{\rm in}$ are equal to ± 1. The kth bit of the encoded message (*i.e.* $J_k^{\rm in}$) is then the product of l_k bits of the original message (*i.e.* the σ's on the r.h.s. of the above equation). The matrix $C^{(k)}_{i_1\ldots i_{l_k}}$ defines the code, *i.e.* it tells from which of the σ's to construct the kth bit of the code. This kind of codes are called parity checking codes because $J_k^{\rm in}$ counts the parity of the minuses among the l_k σ's. The ratio $R=N/M<1$ is called the rate of the code. It specifies the redundancy of the code: low R's correspond to highly redundant codes.

As we previously said, the output of the channel is a sequence of M real numbers $\mathbf{J}^{\rm out}=\{J_1^{\rm out},\ldots,J_M^{\rm out}\}$, which are random variables obeying the probability distribution $Q(J_k^{\rm out}|J_k^{\rm in})$. The next problem consists in decoding the output of the channel $\mathbf{J}^{\rm out}$. Knowing the noise probability (*i.e.* $Q(J_k^{\rm out}|J_k^{\rm in})$), the code (*i.e.* the connectivity matrix $C^{(k)}_{i_1\ldots i_{l_k}}$) and the channel output $\mathbf{J}^{\rm out}$, one has to infer the message that was sent. The quality of inference depends on the choice of the code. According to the famous Shannon's channel encoding theorem, there exist codes which, in the limit of infinitely long messages, allow error-free communication, provided the rate of the code R is less than the channel capacity $\mathcal{C}$. This theorem says that such "ideal" codes exist, but does not say how to construct them.

We have shown that there exists a close mathematical relationship between error-correcting codes and theoretical models of disordered systems. Once the channel output $\mathbf{J}^{\rm out}$ is known, it is possible to compute the probability $\mathcal{P}(\boldsymbol{\tau}|\mathbf{J}^{\rm out})$ for any particular sequence $\boldsymbol{\tau}=\{\tau_1,\ldots,\tau_N\}$ to be the *source* word (*i.e.* the information message). The equivalence between spin-glass models and error correcting codes is based on the following property. The probability $\mathcal{P}(\boldsymbol{\tau}|\mathbf{J}^{\rm out})$ for any sequence $\boldsymbol{\tau}$ to be the information message, conditional on the channel output $\mathbf{J}^{\rm out}$ is given by

$$\ln\mathcal{P}(\boldsymbol{\tau}|\mathbf{J}^{\rm out}) = {\rm const.} + \sum_{k=1}^{M} B_k \times \sum_{i_1\ldots i_{l_k}=1}^{N} C^{(k)}_{i_1\ldots i_{l_k}}\,\tau_{i_1}\cdots\tau_{i_{l_k}} \equiv -H(\boldsymbol{\tau}) \quad (1.7)$$

where

$$B_k \equiv B(J_k^{\rm out}) \equiv \frac{1}{2}\ln\frac{Q(J_k^{\rm out}|1)}{Q(J_k^{\rm out}|-1)}\cdot \quad (1.8)$$

For a Gaussian channel (1.2) it is easy to get $B_k = J_k^{\rm out}/w^2$. We recognise in equation (1.7) the Hamiltonian of a p-spin spin-glass Hamiltonian. The distribution of the couplings is determined by the probability $Q(J^{\rm out}|J^{\rm in})$. In the case when $Q(J^{\rm out}|J^{\rm in}) = Q(-J^{\rm out}|-J^{\rm in})$ (the case of a "symmetric channel"), $B(J^{\rm out}) = -B(-J^{\rm out})$ and one recovers the invariance of the spin-glass Hamiltonian under gauge transformations.

There are different possible strategies for inferring the information message from the probability distribution $\mathcal{P}(\boldsymbol{\tau}|\mathbf{J}^{\rm out})$. "Minimum error probability decoding" (or MED), which is widely used in communications, consists in choosing the most probable sequence $\boldsymbol{\tau}^0$. This is equivalent to finding the ground state of the above spin-glass Hamiltonian. Instead of considering the most probable instance, one may only be interested in the most probable value $\tau_i^{\rm MAP}$ of the "bit" τ_i (maximum *a posteriori* probability or MAP decoding) which can be expressed in terms of the magnetization at temperature $T=1/\beta$ equal to one [8–10]:

$$\tau_i^{\rm MAP} = {\rm sign}\ (m_i); \quad m_i = \frac{1}{Z}\sum_{\boldsymbol{\tau}}\tau_i\ \exp\{-H(\boldsymbol{\tau})\} \quad (1.9)$$

where $H(\boldsymbol{\tau})$ is defined by equation (1.7). It is remarkable that $\beta = 1$ coincides with the Nishimori temperature in spin glasses [11]. MAP decoding is an essential ingredient in turbo decoding (see later).

A widely adopted indicator of the quality of a code is the error probability per bit $P_{\rm e}^{(\rm d)}$. This is defined as the expected fraction of wrong bits in the decoded message when the decoding procedure d is used. When all messages are equally probable and the transmission channel is memoryless and symmetric, $P_{\rm e}^{(\rm d)}$ is the same for all input sequences. It is enough to compute it in the case where all input bits are equal to one. In this case we get

$$P_{\rm e}^{(\rm d)} = \frac{1}{2}(1 - m^{(\rm d)}) \equiv \frac{1}{2}\left(1 - \frac{1}{N}\sum_{i=1}^{N}\tau_i^{(\rm d)}\right) \quad (1.10)$$

where $\tau_i^{(\rm d)}$ is the symbol sequence produced by the decoding procedure.

One can derive a very general lower bound for the error probability per bit $P_{\rm e}^{(\rm MED)}$. It is convenient to introduce a more compact notation. Let us call Ω_i (with $1 \le i \le N$) the set of couplings in which the spin τ_i appears, that is $\Omega_i \equiv \{k \in \{1 \dots M\}$ such that $C^{(k)}_{i_1 \dots i_{l_k}} = 1$ and $i \in \{i_1 \dots i_{l_k}\}\}$. Let Λ_k (with $1 \le k \le M$) denote the set $\{i_1, \dots, i_{l_k}\}$ of spin indices such that $C^{(k)}_{i_1 \dots i_{l_k}} = 1$. In other words $\{\sigma_i | i \in \Lambda_k\}$ is the set of spin coupled by the kth interaction term in the Hamiltonian (1.7). The Hamiltonian (1.7) reads $H(\boldsymbol{\tau}) = -\sum_{k=1}^{M} B_k \prod_{i \in \Lambda_k} \tau_i$.

For the MED decoding procedure the decoded message is determined by the following equations:

$$\tau_i^{\rm MED} = \mathrm{sign}\left[\sum_{k \in \Omega_i} B_k \prod_{j \in \Lambda_k \setminus i} \tau_j^{\rm MED}\right]. \quad (1.11)$$

We proceed *ab absurdo*. Let us suppose that:

(A) $\tau_i^{\rm MED} = +1$ for any $i = 1, \dots, N$;
(B) $|\Omega_i|$ (the size of the set Ω_i) is finite for any $i = 1, \dots, N$; we make the simplifying hypothesis of a "regular" code with $|\Omega_i| = C$ independently of i;
(C) $|\Lambda_k|$ is finite for any $k = 1, \dots, M$; again let us take $|\Lambda_k| = \overline{l}$ for any k.

By equation (1.11) and hypothesis 1 $\sum_{k \in \Omega_i} B_k > 0$ for $i = 1, \dots, N$. Now, because of hypothesis 1 and 1 we can choose a subset $\{i(1), \dots, i(N')\} \subset \{1, \dots, N\}$ such that $N' = O(N)$ when $N \to \infty$ and $\Omega_{i(\alpha)} \cap \Omega_{i(\beta)} = \varnothing$ when $\alpha \neq \beta$. In other words $\sigma_{i(\alpha)}$ and $\sigma_{i(\beta)}$ are not coupled by any interaction if $\alpha \neq \beta$. The probability that $\sum_{k \in \Omega_i} B_k > 0$ for $i = 1, \dots, N$ is bounded from above by the probability that $\sum_{k \in \Omega_{i(\alpha)}} B_k > 0$ for $\alpha = 1, \dots, N'$. The variables $\sum_{k \in \Omega_{i(\alpha)}} B_k$ and $\sum_{k \in \Omega_{i(\beta)}} B_k$ are independent and identically distributed if $\alpha \neq \beta$. Moreover defining[1] $P_{\rm flip} = \mathbb{P}[B_k < 0]$ we have

$$\mathbb{P}[\sum_{k \in \Omega_{i(\alpha)}} B_k > 0] = 1 - \mathbb{P}[\sum_{k \in \Omega_{i(\alpha)}} B_k < 0] \le 1 - \mathbb{P}[B_k < 0 \ \forall k \in \Omega_{i(\alpha)}] = 1 - P_{\rm flip}^C. \quad (1.12)$$

We obtain then the following bound:

$$\mathbb{P}\left[\sum_{k \in \Omega_{i(\alpha)}} B_k > 0, \alpha = 1, \dots, N'\right] \le \left(1 - P_{\rm flip}^C\right)^{N'}. \quad (1.13)$$

Generally speaking $P_{\rm flip}$ is strictly positive for any non vanishing noise level. For example for the Gaussian channel defined in equation (1.2) one gets $P_{\rm flip} = \int_{1/w}^{\infty} {\rm d}x/\sqrt{2\pi}\, {\rm e}^{-x^2/2}$. In the thermodynamic limit ($N \to \infty$) $N' \to \infty$ and the r.h.s. of equation (1.13) $\left(1 - P_{\rm flip}^C\right)^{N'} \to 0$. As a consequence $\tau_i^{\rm MED} = +1$ for $i = 1, \dots, N$ cannot satisfy equation (1.11) and some spins must be flipped, *i.e.* some errors must occur.

The occurrence of a "small" number of errors $n_{\rm err}(N)$ such that $\lim_{N\to\infty} n_{\rm err}(N)/N = 0$ does not imply $P_{\rm e}^{(\rm MED)}$ to be non zero in the $N \to \infty$ limit. However the reasoning outlined above can be repeated if the hypothesis 1 above is substituted by

(A') The number of flipped spins $\tau_i^{\rm MED} = -1$ in the ground state $n_{\rm err}(N)$ is such that $\lim_{N\to\infty} n_{\rm err}(N)/N = 0$.

We conclude that $n_{\rm err}(N)$ is of order N and $P_{\rm e}^{(\rm MED)}$ is non zero. Transmission with zero error probability is then possible only in the limit $C \to \infty$.

The rate of the code is given by $R = N/M = \overline{l}/C$. A necessary condition for a finite rate code to achieve zero error probability, is that the number of spins coupled together (*i.e.* $\overline{l}$) diverges in the thermodynamic limit ($N \to \infty$). This condition is realized in Derrida's random energy model [12] which has been shown to be an ideal code [2] (in that case $R = 0$). In references [13,14] the authors investigated the behavior of a class of codes of the type described above where the connectivity matrix was constructed randomly. They concluded that zero error probability per bit is effectively achieved in the limit $C \to \infty$, $\overline{l} \to \infty$ at $\overline{l}/C = R$ fixed if the rate R is lower than the channel capacity. We will show in the following that zero error probability can be achieved using recursive turbo codes, but not using non recursive turbo codes.

2 Convolutional codes

Convolutional codes are the building blocks of turbo codes. In this section we shall describe both non recursive and recursive convolutional codes and the corresponding spin models. The information message, *i.e.* the

[1] We use the notation $\mathbb{P}[A]$ for denoting the probability of the event A.

source word, will be denoted by: $\boldsymbol{\tau} \equiv (\tau_1, \ldots, \tau_N)$. It is convenient to think of the source producing a symbol per unit time, *i.e.* in τ_i, i denotes the time. Similarly the code word can be considered also as a time series. Convolutional codes are very easy to implement in hardware. They require a certain (small) number r of memory registers. We shall call r the range of the code. Let's denote by $\Sigma_1(t), \ldots, \Sigma_r(t)$ the content of the memory registers at time t. At each time step the content of each memory register is shifted to the right: $\Sigma_{j+1}(t+1) = \Sigma_j(t)$ for $j = 1, \ldots, r-1$. For convenience of notation we define $\Sigma_0(t) \equiv \Sigma_1(t+1)$, and the "register sequence" (or "register word") $\boldsymbol{\sigma}$ according to the rule $\sigma_i \equiv \Sigma_0(i)$.

The code word letters at time t are constructed from the content of the memory registers at time t. The encoding for a general convolutional code can be regarded as a two step procedure: (I) At every time step t update the values of the memory registers, *i.e.* compute $\Sigma_1(t+1)$ (since for the others $\Sigma_{j+1}(t+1) = \Sigma_j(t)$). The bit $\Sigma_1(t+1)$ will be a function of the incoming source letter τ_i, $i = t$, and of the value of the registers at time t. The choice of the function depends on the particular code. (II) Compute the new letters of the code word as a function of the register sequence $\Sigma_i(t)$, $i = 0, \cdots, r$. Again the choice of the function depends on the particular code.

For simplicity we consider codes of rate $R = 1/2$. In this case the encoded message (the code word) has the form $\mathbf{J} \equiv (J_1^{(1)}, \ldots, J_N^{(1)}; J_1^{(2)}, \ldots, J_N^{(2)})$. We will provide explicit examples of convolutional codes in the following.

The register letters σ's are function of the source word:

$$\boldsymbol{\tau} \mapsto \boldsymbol{\sigma}(\boldsymbol{\tau}) \equiv (\sigma_1(\boldsymbol{\tau}), \ldots, \sigma_N(\boldsymbol{\tau})). \tag{2.1}$$

When not ambiguous we shall omit in the following the dependence of $\boldsymbol{\sigma}$ upon $\boldsymbol{\tau}$.

For a code to be meaningful, the correspondence between source words $\boldsymbol{\tau}$ and register words $\boldsymbol{\sigma}$ must be a one to one correspondence. Because of this, instead of the probability distribution (1.7) of the source word conditional on the channel output, we may as well consider the probability distribution of the register word (conditional on the channel output). The resulting expressions can be simpler and more illuminating (see later).

2.1 Non recursive convolutional codes

In order to define non recursive convolutional codes we must specify the two steps outlined above.

(I) For non recursive convolutional codes the register word is identical to the source word:

$$\sigma_i(\boldsymbol{\tau}) = \tau_i. \tag{2.2}$$

This implies $\Sigma_0(i) = \tau_i$.

(II) The code word $\mathbf{J}$ is a function of the register word defined as:

$$J_i^{(\alpha)} = \prod_{j=0}^{r} (\sigma_{i-j})^{\kappa(j;\alpha)}; \quad i = 1, \ldots, N. \tag{2.3}$$

($\alpha = 1, 2$ for $R = 1/2$ codes). The exponents $\kappa(j;\alpha) \in \{0, 1\}$ define the code. Each letter $J_i^{(\alpha)}$ of the code word is the product (or equivalently the modulo 2 addition, if one uses the language of bits instead of Ising spins) of the memory registers for which $\kappa(j;\alpha) = 1$.

We shall assume hereafter that $\kappa(0;1) = \kappa(0;2) = 1$. To avoid redundancy we choose r such that either $\kappa(r;1)$ or $\kappa(r;2)$ are different from 0. To make equation (2.3) meaningful for $i = 1, \ldots, r$ we define $\sigma_j = +1$ for $j \leq 0$. Notice however that the exact definition of $J_1^{(\alpha)}, \ldots, J_r^{(\alpha)}$ is irrelevant in the thermodynamic limit.

We now give some examples of these definitions and relate them to the general scheme outlined in the Introduction. For each example we describe explicitly the step (II) and the corresponding exponents $\kappa(\cdot;\cdot)$ as well as the connectivity matrix.

(a) The simplest non trivial convolutional code has range $r = 1$:

$$J_i^{(1)} = \sigma_i \sigma_{i-1} \Rightarrow \kappa(0;1) = \kappa(1;1) = 1\ ; \quad \kappa(j;1) = 0 \text{ for } j \geq 2, \tag{2.4}$$

$$J_i^{(2)} = \sigma_i \Rightarrow \kappa(0;2) = 1\ ; \quad \kappa(j;2) = 0 \text{ for } j \geq 2. \tag{2.5}$$

This can be described using the following connectivity matrix: $C_{i_1,i_2}^{(k)} = \delta_{i_1,k}\delta_{i_2,k-1}$ for $k = 1, \ldots, N$ and $C_i^{(k)} = \delta_{i,k}$ for $k = N+1, \ldots, 2N$. Here and in the other examples $M = 2N$ since the rate is $R = 1/2$.

(b) A simple code with range $r = 2$ whose behaviour will be examined in what follows:

$$J_i^{(1)} = \sigma_i \sigma_{i-1} \sigma_{i-2} \Rightarrow \kappa(0;1) = \kappa(1;1) = \kappa(2;1) = 1; \quad \kappa(j;1) = 0 \text{ for } j \geq 3, \tag{2.6}$$

$$J_i^{(2)} = \sigma_i \sigma_{i-2} \Rightarrow \kappa(0;2) = \kappa(2;2) = 1\ ; \quad \kappa(j;2) = 0 \text{ for } j \neq 0, 2. \tag{2.7}$$

The corresponding connectivity matrix is $C_{i_1,i_2,i_3}^{(k)} = \delta_{i_1,k}\delta_{i_2,k-1}\delta_{i_3,k-2}$ for $k = 1, \ldots, N$ and $C_{i_1,i_2}^{(k)} = \delta_{i_1,k}\delta_{i_2,k-2}$ for $k = N+1, \ldots, 2N$.

(c) The code with range $r = 4$ used by Berrou and collaborators to build the first example of turbo code:

$$J_i^{(1)} = \sigma_i \sigma_{i-1} \sigma_{i-2} \sigma_{i-3} \sigma_{i-4} \Rightarrow \kappa(0;1) = \cdots = \kappa(4;1) = 1\ ; \kappa(j;1) = 0 \text{ for } j \geq 5, \tag{2.8}$$

$$J_i^{(2)} = \sigma_i \sigma_{i-4} \Rightarrow \kappa(0;2) = \kappa(4;2) = 1\ ; \quad \kappa(j;2) = 0 \text{ for } j \neq 0, 4. \tag{2.9}$$

The connectivity matrix is easily constructed: $C_{i_1,i_2,i_3,i_4,i_5}^{(k)} = \delta_{i_1,k}\delta_{i_2,k-1}\delta_{i_3,k-2}\delta_{i_4,k-3}\delta_{i_5,k-4}$ for $k = 1, \ldots, N$ and $C_{i_1,i_2}^{(k)} = \delta_{i_1,k}\delta_{i_2,k-4}$ for $k = N+1, \ldots, 2N$.

As we said above the numbers $\kappa(j;\alpha)$ define the particular code. It is sometimes useful to represent convolutional codes in the following compact form. Using

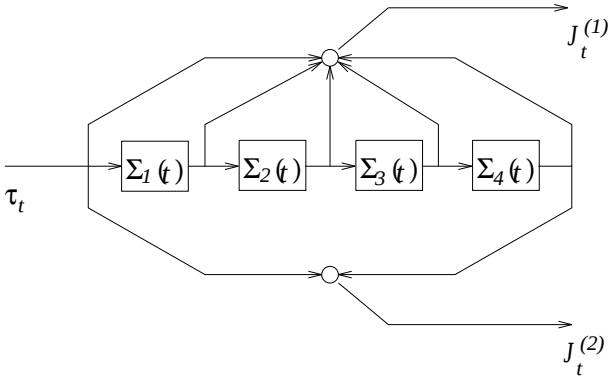

Fig. 1. Schematic representation of the encoder for a non recursive convolutional code called code (c) in the text and defined by equations (2.8, 2.9).

the correspondence of "bits" which take values 0 or 1 and Ising spins, one can formally represent the source word as a polynomial on $\mathbb{Z}_2$ (a polynomial whose coefficients are zeros or ones) $H(x) \equiv \sum_{j=1}^{N} H_j x^j$; $\tau_j \equiv (-1)^{H_j}$. The coefficients of the different powers of x are the source letters. The variable x is introduced only for book keeping purposes. Similarly for the register word $G(x) \equiv \sum_{j=1}^{N} G_j x^j$; $\sigma_j \equiv (-1)^{G_j}$ and for the code word $\mathcal{G}^{(\alpha)}(x) \equiv \sum_{j=1}^{N} \mathcal{G}_j^{(\alpha)} x^j$; $J_j^{(\alpha)} \equiv (-1)^{\mathcal{G}_j^{(\alpha)}}$ To each of the two sets of numbers $\kappa(j;1)$ and $\kappa(j;2)$ associate a polynomial of degree r

$$g_\alpha(x) = \sum_{j=0}^{r} \kappa(j;\alpha)\, x^j. \tag{2.10}$$

Using these definitions, encoding for non recursive convolutional codes can be restated as follows

$$G(x) = H(x)\,, \tag{2.11}$$

$$\mathcal{G}^{(1)}(x) = g_1(x)G(x), \quad \mathcal{G}^{(2)}(x) = g_2(x)G(x). \tag{2.12}$$

Equations (2.11) and (2.12) correspond respectively to the steps (I) and (II) of the encoding procedure. Multiplication is done using modulo two addition. All these polynomials have coefficients equal to 0 or 1. From equation (2.12) it follows that $\mathcal{G}_i^{(\alpha)} = \sum_{j=0}^{r} \kappa(j;\alpha) G_{i-j}$: the encoded message is obtained from the register sequence through a convolution (whence the denomination of the code) with the numbers $\kappa(j;\alpha)$. Written in terms of spin variables $J_i^{(\alpha)}$ and σ_i, this convolution coincides with equation (2.3).

The polynomials $g_\alpha(x)$ define the code and are called generating polynomials. The generating polynomials which correspond to the examples (a)-(c) above are:
(a) $g_1(x) = 1 + x$, $g_2(x) = 1$;
(b) $g_1(x) = 1 + x + x^2$, $g_2(x) = 1 + x^2$;
(c) $g_1(x) = 1 + x + x^2 + x^3 + x^4$, $g_2(x) = 1 + x^4$.

The structure of a non recursive convolutional encoder is reproduced in Figure 1. The encoding procedure works as follows: the bits of the message enter the shift register and the contents of the appropriate registers are added (modulo two) for producing the encoded message.

2.2 Recursive convolutional codes

Recursive convolutional codes are most easily defined in terms of the generating polynomials. The difference with non recursive codes is in the step (I), *i.e.* in the relation between the source word and the register word. In the non recursive case this relation was given by equation (2.2) or by equation (2.12). For recursive codes the two steps works as follows.

(I)

$$G(x) = \frac{1}{g_1(x)} H(x). \tag{2.13}$$

(II) The second step works exactly as in the non recursive case, *cf.* equation (2.12): we have $\mathcal{G}^{(1)}(x) = g_1(x)G(x)$ and $\mathcal{G}^{(2)}(x) = g_2(x)G(x)$.

Combining the two steps we obtain:

$$\mathcal{G}^{(1)}(x) = H(x), \quad \mathcal{G}^{(2)}(x) = \frac{g_2(x)}{g_1(x)} H(x). \tag{2.14}$$

In general $g_2(x)/g_1(x)$ is not a polynomial, nevertheless $\mathcal{G}^{(2)}(x)$ can be computed using only r memory registers (see later). Notice that unlike in the non recursive case the role of the two polynomials $g_1(x)$ and $g_2(x)$ is not symmetric. Two different recursive codes can be defined by permuting them.

We can restate this procedure in terms of the spin variables $\boldsymbol{\tau}$, $\boldsymbol{\sigma}$ and $\mathbf{J}$.

(I) From equation (2.13) one obtains $G(x) = H(x) + [g_1(x)-1]G(x)$ (remember that $+1 = -1 \bmod 2$) that is, using $\kappa(0;1) = 1$, $G_i = H_i + \sum_{j=1}^{r} \kappa(j;1) G_{i-j}$. This relation can be rewritten in terms of the spin variables as follows:

$$\sigma_i(\boldsymbol{\tau}) = \tau_i \prod_{j=1}^{r} (\sigma_{i-j})^{\kappa(j;1)}. \tag{2.15}$$

This relation defines the register word (the σ's) as a function of the source word (the τ's) recursively: once $\sigma_1(\boldsymbol{\tau}), \ldots, \sigma_{i-1}(\boldsymbol{\tau})$ have been computed equation (2.15) allows to determine $\sigma_i(\boldsymbol{\tau})$.

(II) The second step works as in the non recursive case, *cf.* equation (2.3): $J_i^{(\alpha)} = \prod_{j=0}^{r} (\sigma_{i-j})^{\kappa(j;\alpha)}$ for any $i = 1, \ldots, N$ and $\alpha = 1, 2$.

From equation (2.15) it follows that:

$$\tau_i(\boldsymbol{\sigma}) = \prod_{j=0}^{r} (\sigma_{i-j})^{\kappa(j;1)}. \tag{2.16}$$

Notice that $J_i^{(1)} = \tau_i$: in the recursive case a part of the encoded message is always the message itself.

The encoding procedure is depicted in Figure 2.

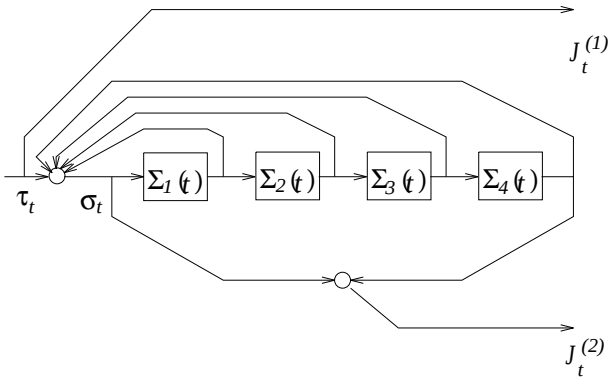

Fig. 2. The encoder for the recursive convolutional code with the same generating polynomials as in Figure 1 (*cf.* Eqs. (2.8, 2.9)).

2.3 Decoding and spin model formulation

The step (II) of the encoding procedure for convolutional codes (from the register word to the code word: $\boldsymbol{\sigma} \to \mathbf{J}$) is of the form (1.6), see equation (2.3). The corresponding connectivity matrix has been worked out explicitly for the examples (a)-(c) in Section 2.1 and can be easily written in the general case. We can then use the method explained in the introduction in order to write the probability distribution of the register word, conditional to the channel output, as the Boltzmann weight of a spin model with random couplings. The Hamiltonian is in this case:

$$H(\boldsymbol{\sigma};\mathbf{J}^{\rm out}) = -\sum_{i=1}^{N}\left\{ B(J_i^{(1),{\rm out}})\prod_{j=0}^{r}(\sigma_{i-j})^{\kappa(j;1)} + B(J_i^{(2),{\rm out}})\prod_{j=0}^{r}(\sigma_{i-j})^{\kappa(j;2)}\right\}, \quad (2.17)$$

where $B(\cdot)$ is defined in equation (1.8). The spin model (2.17) is one dimensional with two types of couplings. The range of the interaction coincides with the range of the code.

The Hamiltonian (2.17) is expressed as a function of the spins of the register sequence σ_i, instead of the source sequence τ_i used in the introduction. For non recursive codes $\tau_i = \sigma_i$. For recursive codes τ_i is given by equation (2.16). In this last case decoding can be thought of as the computation of an expectation value of a product of σ's. More precisely $\tau_i(\boldsymbol{\sigma})$ is a "local" product of spin variables: the position of the σ's on the r.h.s. of equation (2.16) are separated at most by a distance r. Such a product could be called a "composite operator" in the language of statistical field theory.

The spin Hamiltonian is the same for both the recursive and non recursive codes. We define the decoding at arbitrary temperature $T \equiv 1/\beta$ as follows:

$$\tau_i^\beta \equiv {\rm sign}(\langle\tau_i(\boldsymbol{\sigma})\rangle_\beta), \quad (2.18)$$

$$\langle O(\boldsymbol{\sigma})\rangle_\beta \equiv \frac{1}{Z(\mathbf{J}^{\rm out};\beta)} \times \sum_{\boldsymbol{\sigma}} O(\boldsymbol{\sigma})\exp\{-\beta H(\boldsymbol{\sigma};\mathbf{J}^{\rm out})\}, \quad (2.19)$$

where the expression for $\tau_i(\boldsymbol{\sigma})$ is given by equation (2.16) or by equation (2.2) depending whether the code is recursive or not.

As seen in the introduction there are two widely used decoding strategies:

- minimum error decoding (also called maximum likelihood decoding) which consists in finding the most probable sequence of bits and corresponds to the choice $\beta = \infty$ in equation (2.18): $\tau_i^{(\rm MED)} \equiv \tau_i^{\beta=\infty}$;
- maximum *a posteriori* probability decoding which consists in finding the sequence of most probable bits and corresponds to the choice $\beta = 1$ in equation (2.18): $\tau_i^{\rm MAP} \equiv \tau_i^{\beta=1}$. This is the strategy which enters in turbo decoding.

Both this strategies can be implemented in a very efficient way using the transfer matrix technique. The corresponding algorithms are known in communication theory as the Viterbi algorithm [6] for the $\beta = \infty$ case and the BCJR algorithm [7] for the $\beta = 1$ case. The complexity of these algorithms grows like $N2^r$.

The use of the letters of the register word (*i.e.* of the σ variables) makes evident the similarity between recursive and non recursive codes: they correspond to the same spin model. This implies *e.g.* that, if zero temperature decoding is adopted, the probability of transmitting a message without errors is the same for the two codes. In the limit $r \to \infty$ it is possible to construct convolutional codes corresponding to spin models with infinite connectivity and couplings between an infinite number of spins. They should allow to transmit without errors when the noise is low enough. In practice, because of the growing complexity of the transfer matrix algorithm, a compromise between low r's (which are simpler to decode) and high r's (which show better performances) must be found. The values of r used in practical cases are between 2 and 7.

We can write the decoding strategy in terms of the source word (*i.e.* of the τ variables) without making use of the register word (*i.e.* of the σ variables). From equations (2.18, 2.19) we get

$$\tau_i^\beta = {\rm sign}\left\{\frac{1}{Z(\mathbf{J}^{\rm out};\beta)}\sum_{\boldsymbol{\tau}}\tau_i\exp\{-\beta H(\boldsymbol{\sigma}(\boldsymbol{\tau});\mathbf{J}^{\rm out})\}\right\}. \quad (2.20)$$

For non recursive codes, because of equation (2.2), the source word and the register word are identical. For recursive codes equation (2.16) cannot be inverted in a local way. When we rewrite equation (2.17) in terms of the τ variables we obtain a non local Hamiltonian.

As a simple illustration of this observation we can consider the Hamiltonian corresponding to the code (a). When written in terms of the σ variables the Hamiltonian reads, *cf.* equations (2.4, 2.5, 2.17):

$$H^{(a)}(\boldsymbol{\sigma};\mathbf{J}) = -\sum_{i=1}^{N} B(J_i^{(1),\mathrm{out}})\sigma_i\sigma_{i-1} - \sum_{i=1}^{N} B(J_i^{(2),\mathrm{out}})\sigma_i. \quad (2.21)$$

Recall that $\kappa(0;1) = \kappa(1;1) = 1$ and $\kappa(j;1) = 0$ for $j \geq 2$, see equation (2.4). From equation (2.16) we get $\tau_i = \sigma_i\sigma_{i-1}$. We want to express the σ's as a function of the τ's. We easily get $\sigma_i = \tau_i\sigma_{i-1}$ which implies $\sigma_i = \tau_i\tau_{i-1}\cdots\tau_2\tau_1$. These relations allow us to express the Hamiltonian (2.21) an a function of the τ's:

$$H^{(a)}(\boldsymbol{\sigma}(\boldsymbol{\tau});\mathbf{J}) = -\sum_{i=1}^{N} B(J_i^{(1),\mathrm{out}})\tau_i - \sum_{i=1}^{N} B(J_i^{(2),\mathrm{out}})\prod_{j=1}^{i}\tau_j. \quad (2.22)$$

Let us consider a generic recursive convolutional code. We can express Hamiltonian (2.17) in terms of the τ variables using the generating polynomials technique. We define the numbers $\rho(j) \in \{0,1\}$ as follows: $g_2(x)/g_1(x) \equiv \sum_{j=0}^{\infty}\rho(j)x^j \pmod 2$. From equation (2.14) we obtain $\mathcal{G}_i^{(1)} = H_i$ and $\mathcal{G}_i^{(2)} = \sum_{j=0}^{\infty}\rho(j)H_{i-j}$. We can express these relations in terms of spin variables: $J_i^{(1),\mathrm{in}} = \tau_i$ (which we already derived) and $J_i^{(1),\mathrm{in}} = \prod_{j=0}^{\infty}(\tau_{i-j})^{\rho(j)}$. Since we imposed the boundary condition $\tau_i = +1$ for $i \leq 0$, we get

$$H(\boldsymbol{\sigma}(\boldsymbol{\tau});\mathbf{J}) = -\sum_{i=1}^{N} B(J_i^{(1),\mathrm{out}})\tau_i - \sum_{i=1}^{N} B(J_i^{(2),\mathrm{out}})\prod_{j=0}^{i-1}(\tau_{i-j})^{\rho(j)}. \quad (2.23)$$

Written in this form recursive codes look very different from non recursive ones with the same range. If $g_2(x)$ is not divisible by $g_1(x)$ the corresponding spin models have infinite connectivity and interactions with infinite range; they are similar, in this respect, to $r = \infty$ non recursive codes. Nevertheless they do not behave, in general, radically better than non recursive codes with the same range because there exists, as we have shown, a change of variables (from τ to σ) which makes the model local.

3 Turbo codes

A turbo code is defined by the choice of a convolutional code and of a permutation of N objects. We use for the permutation the following notation:

$$P : \{1,\ldots,N\} \to \{1,\ldots,N\}, \quad (3.1)$$
$$i \mapsto P(i), \quad (3.2)$$

and we shall denote by P^{-1} the inverse permutation ($P(P^{-1}(i)) = P^{-1}(P(i)) = i$). The basic idea is to apply the permutation P to the source sequence $\boldsymbol{\tau}$ to produce a new sequence $\boldsymbol{\tau}^P$. Obviously $\boldsymbol{\tau}^P$ does not carry any new information because P is known. The sequences $\boldsymbol{\tau}$ and $\boldsymbol{\tau}^P$ are the inputs of two set of registers, each one implementing a convolutional encoding. In this way the rate of the code is decreased (*i.e.* greater redundancy). One can increase the rate by erasing some of the outputs [1], but we will not consider here this possibility.

The properties of the code can strongly depend on the choice of the permutation. We will see later that permutations "near" the identity give bad codes. We shall think to a "good" permutation as to a random permutation. In the limit $N \to \infty$ they are "far" from the identity with probability one. We shall discuss this point later in this section.

In the first step (I) *two* different register words $\boldsymbol{\sigma}^{(1)}$ and $\boldsymbol{\sigma}^{(2)}$ are constructed from the information message. In the second step (II) the code word $\mathbf{J}$ is constructed from the register words $\boldsymbol{\sigma}^{(1)}$ and $\boldsymbol{\sigma}^{(2)}$. The turbo code has rate $R = 1/3$ and the encoded message has the form $\mathbf{J} = \{J_1^{(0)},\ldots,J_N^{(0)}; J_1^{(1)},\ldots,J_N^{(1)}; J_1^{(2)},\ldots,J_N^{(2)}\}$. The "building blocks" will be a rate $1/2$ recursive convolutional code, defined by the constants $\kappa(j;1)$ and $\kappa(j;2)$ and a permutation P. The encoding procedure can be described more precisely as follows.

(I) From the message $\boldsymbol{\tau} = \{\tau_1,\ldots,\tau_N\}$ we construct the permuted sequence $\boldsymbol{\tau}^P = \{\tau_1^P,\ldots,\tau_N^P\}$ according to the rule $\tau_i^P \equiv \tau_{P(i)}$. Two register words $\boldsymbol{\sigma}^{(1)} = \{\sigma_1^{(1)},\ldots,\sigma_N^{(1)}\}$ and $\boldsymbol{\sigma}^{(2)} = \{\sigma_1^{(2)},\ldots,\sigma_N^{(2)}\}$ are then produced from the sequences $\boldsymbol{\tau}$ and $\boldsymbol{\tau}^P$ as follows:

$$\sigma_i^{(1)} \equiv \sigma_i(\boldsymbol{\tau}), \quad \sigma_i^{(2)} \equiv \sigma_i(\boldsymbol{\tau}^P). \quad (3.3)$$

The application $\boldsymbol{\tau} \mapsto \sigma_i(\boldsymbol{\tau})$ is defined as for recursive convolutional codes:

$$\sigma_i(\boldsymbol{\tau}) = \tau_i\prod_{j=1}^{r}(\sigma_{i-j}(\boldsymbol{\tau}))^{\kappa(j;1)}. \quad (3.4)$$

This implies the relations

$$\tau_i = \prod_{j=0}^{r}(\sigma_{i-j}^{(1)})^{\kappa(j;1)} \qquad \tau_i^P = \prod_{j=0}^{r}(\sigma_{i-j}^{(2)})^{\kappa(j;1)}. \quad (3.5)$$

In the following we shall use the notation

$$\epsilon_i(\boldsymbol{\sigma}) \equiv \prod_{j=1}^{r}(\sigma_{i-j})^{\kappa(j;1)} \quad (3.6)$$

for any spin configuration $\boldsymbol{\sigma} = \{\sigma_1,\ldots,\sigma_N\}$. The relations (3.5) can then be shortened as $\tau_i = \epsilon_i(\boldsymbol{\sigma}^{(1)})$ and $\tau_i^P = \epsilon_i(\boldsymbol{\sigma}^{(2)})$.

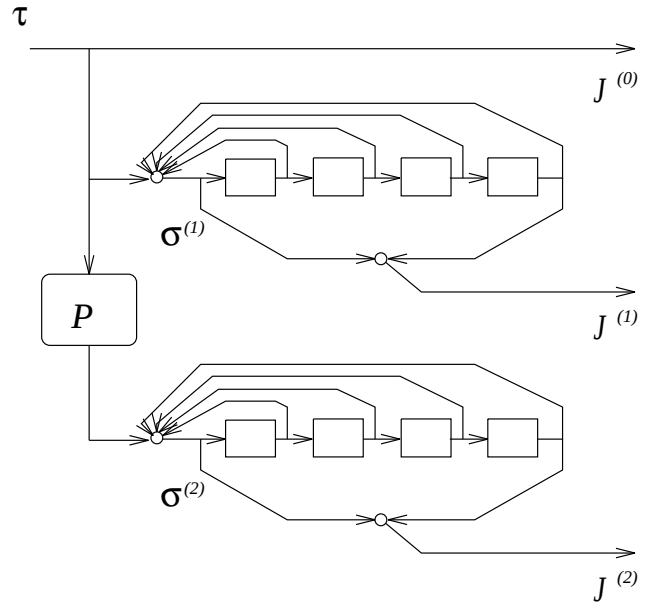

Fig. 3. Schematic representation of the encoder for a recursive turbo code with generating polynomials as in the previous figures (*cf.* Eqs. (2.8, 2.9)). Notice the presence of the interleaver (denoted by P) which implements the permutation.

(II) The code word $\mathbf{J} = \{J_1^{(0)}, \ldots, J_N^{(0)}; J_1^{(1)}, \ldots, J_N^{(1)}; J_1^{(2)}, \ldots, J_N^{(2)}\}$ is constructed as follows from the register words:

$$J_i^{(0)} \equiv \prod_{j=0}^{r} (\sigma_{i-j}^{(1)})^{\kappa(j;1)}, \quad J_i^{(1)} \equiv \prod_{j=0}^{r} (\sigma_{i-j}^{(1)})^{\kappa(j;2)}, \quad J_i^{(2)} \equiv \prod_{j=0}^{r} (\sigma_{i-j}^{(2)})^{\kappa(j;2)}. \quad (3.7)$$

There is an apparent asymmetry between the roles of $\boldsymbol{\sigma}^{(1)}$ and $\boldsymbol{\sigma}^{(2)}$ in equation (3.7). However from the definition of the permuted sequence ($\tau_i^P = \tau_{P(i)}$) and equation (3.5) we get

$$\prod_{j=1}^{r} (\sigma_{i-j}^{(2)})^{\kappa(j;1)} = \prod_{j=1}^{r} (\sigma_{P(i)-j}^{(1)})^{\kappa(j;1)}, \quad (3.8)$$

whence $J_i^{(0)} = \prod_{j=0}^{r} (\sigma_{P^{-1}(i)-j}^{(2)})^{\kappa(j;1)}$. Moreover from equations (3.5, 3.7) we obtain $J_i^{(0)} = \tau_i$.

The structure of the encoding procedure (permutation and parallel encoding through recursive convolutional encoders) is reproduced in the scheme of Figure 3.

It is convenient to write the turbo codes Hamiltonian as a function of both register words $\boldsymbol{\sigma}^{(1)}$ and $\boldsymbol{\sigma}^{(2)}$. For convolutional codes we had a unique register word in one-to-one correspondence to the source word. For turbo codes we have two different sets of registers and a different word for each set. The two register words satisfy the constraint (3.8). We will shorten equation (3.8) as $\epsilon_i(\boldsymbol{\sigma}^{(2)}) = \epsilon_{P(i)}(\boldsymbol{\sigma}^{(1)})$, see equation (3.6). We treat $\boldsymbol{\sigma}^{(1)}$ and $\boldsymbol{\sigma}^{(2)}$ as if they were independent and impose the constraint by a Kronecker δ function:

$$\mathcal{P}(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)} | \mathbf{J}^{\rm out}) = \frac{1}{Z(\mathbf{J}^{\rm out})} \times \prod_{i=1}^{N} \delta(\epsilon_{P(i)}(\boldsymbol{\sigma}^{(1)}), \epsilon_i(\boldsymbol{\sigma}^{(2)})) \, \exp\{-H(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}; \mathbf{J}^{\rm out})\}, \quad (3.9)$$

$$H(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}; \mathbf{J}^{\rm out}) = -\sum_{i=1}^{N} \left\{ B(J_i^{(0),{\rm out}}) \prod_{j=0}^{r} (\sigma_{i-j}^{(1)})^{\kappa(j;1)} + B(J_i^{(1),{\rm out}}) \prod_{j=0}^{r} (\sigma_{i-j}^{(1)})^{\kappa(j;2)} + B(J_i^{(2),{\rm out}}) \prod_{j=0}^{r} (\sigma_{i-j}^{(2)})^{\kappa(j;2)} \right\}. \quad (3.10)$$

In this way the probability distribution is a local function of the spin variables $\boldsymbol{\sigma}^{(1)}$ and $\boldsymbol{\sigma}^{(2)}$. The corresponding physical model can be seen as the union of two one-dimensional spin chains (the register words $\boldsymbol{\sigma}^{(1)}$ and $\boldsymbol{\sigma}^{(2)}$) which interact through the constraint in equation (3.9).

A particularly simple case is obtained if P is chosen to be the identity permutation. In this case from equation (3.8) we get $\boldsymbol{\sigma}^{(1)} = \boldsymbol{\sigma}^{(2)}$. We can use the constraint in equation (3.9) for eliminating one of the two register words. The code becomes a convolutional one with the same rate (1/3) and the same generating polynomials. We shall use the convolutional code obtained in this way as a standard comparison term for the performances of turbo codes (see Figs. 4–6). The outcome of this comparison (we will find that recursive turbo codes have a much lower error probability than convolutional codes) demonstrates the importance of the choice of the permutation.

For a generic random permutation it is rather difficult to express one of the two register words in terms of the other using equation (3.8). Let us consider as an example the code (a). In this case $\kappa(0;1) = \kappa(1;1) = 1$ and $\kappa(j;1) = 0$ for $j \geq 2$, see equation (2.4). Equation (3.8) reads $\sigma_i^{(2)} \sigma_{i-1}^{(2)} = \sigma_{P(i)}^{(1)} \sigma_{P(i)-1}^{(1)}$ whence

$$\sigma_i^{(1)} = \prod_{j=1}^{i} \sigma_j^{(1)} \sigma_{j-1}^{(1)} = \prod_{j=1}^{i} \sigma_{P^{-1}(j)}^{(2)} \sigma_{P^{-1}(j)-1}^{(2)}. \quad (3.11)$$

It is simple to show that, for a random permutation, the number of different $\sigma^{(2)}$'s in the product on the r.h.s. of equation (3.11) is of order N. What do we learn from this simple example? If we try to express the probability distribution (3.9) in terms of a unique register word ($\boldsymbol{\sigma}^{(1)}$ or $\boldsymbol{\sigma}^{(2)}$) using the constraint in equation (3.9), *i.e.* equation (3.8) we obtain a model with large connectivities.

An exception is given by non recursive turbo codes. In this case $\kappa(j;1) = \delta_{j,0}$. Using equation (3.5) we get $\tau_i = \sigma_i^{(1)}$ and $\tau_i^P = \sigma_i^{(2)}$ whence $\sigma_i^{(2)} = \sigma_{P(i)}^{(1)}$, see equation (3.8). The two register words are related simply by a permutation. The probability distribution can

be easily written in terms of a unique register word:

$$\mathcal{P}_{\rm non-rec}(\boldsymbol{\sigma}^{(1)}|\mathbf{J}^{\rm out}) = \frac{1}{Z(\mathbf{J}^{\rm out})}\exp\{-H_{\rm non-rec}(\boldsymbol{\sigma}^{(1)},(\boldsymbol{\sigma}^{(1)})^P;\mathbf{J}^{\rm out})\} \tag{3.12}$$

$$H_{\rm non-rec}(\boldsymbol{\sigma}^{(1)},\boldsymbol{\sigma}^{(2)};\mathbf{J}^{\rm out}) \equiv -\sum_{i=1}^{N}\left\{B(J_i^{(0)\rm out})\sigma_i^{(1)} + B(J_i^{(1)\rm out})\prod_{j=0}^{r}(\sigma_{i-j}^{(1)})^{\kappa(j;2)} + B(J_i^{(2)\rm out})\prod_{j=0}^{r}(\sigma_{i-j}^{(2)})^{\kappa(j;2)}\right\}. \tag{3.13}$$

The spin model corresponding to this type of code has a finite connectivity $C = 1+2\sum_{j=0}^{r}\kappa(j;2)$. This finite *versus* infinite connectivity is the essential difference between non recursive and recursive turbo codes and explains why recursive turbo codes are so better and why they can achieve zero error probability for low enough noise.

4 Turbo decoding

In this section we discuss the decoding of turbo codes. In order to decode turbo codes one would like to compute expectation values with respect to the Hamiltonian (3.10).

There is no exact (and practical) decoding algorithm for turbo codes. Berrou *et al.* [1] have proposed a very ingenious algorithm, called turbo decoding, which is thought to be approximate. Turbo decoding is an iterative procedure. At each step of the iteration, one considers one of the two chains, *i.e.* either the couplings $\mathbf{J}^{(0)}$ and $\mathbf{J}^{(1)}$ or $\mathbf{J}^{(0)}$ and $\mathbf{J}^{(2)}$ and proceeds to MAP decoding. The information so obtained is injected to the next step by adding appropriate external fields to the Hamiltonian. The algorithm terminates if a fixed point is reached.

In order to explain the algorithm more precisely, we introduce the following expectation values:

$$\Xi_i[\mathbf{B},\mathbf{B}'] \equiv \frac{1}{Z}\sum_{\boldsymbol{\sigma}}\epsilon_i(\boldsymbol{\sigma})\exp\left\{\sum_{i=1}^{N}B_i\prod_{j=0}^{r}(\sigma_{i-j})^{\kappa(j;1)} + \sum_{i=1}^{N}B'_i\prod_{j=0}^{r}(\sigma_{i-j})^{\kappa(j;2)}\right\}. \tag{4.1}$$

The Ξ_i's can be computed in an efficient way by using the finite temperature transfer matrix algorithm. They are expectation values of the products of σ's which appear in equation (3.5), see equation (3.6). Then we introduce the iteration variables $\boldsymbol{\theta}^{(m)}(t) \equiv (\theta_1^{(m)}(t),\ldots,\theta_N^{(m)}(t))$ and $\boldsymbol{\Gamma}^{(m)}(t) \equiv (\Gamma_1^{(m)}(t),\ldots,\Gamma_N^{(m)}(t))$ with $m = 1,2$. In terms of these variables the iteration reads

$$\theta_i^{(1)}(t+1) = \Xi_i[\mathbf{B}^{(0)} + \boldsymbol{\Gamma}^{(1)}(t),\mathbf{B}^{(1)}], \tag{4.2}$$

$$\theta_i^{(2)}(t+1) = \Xi_i[\mathbf{B}^{(0),P} + \boldsymbol{\Gamma}^{(2)}(t),\mathbf{B}^{(2)}], \tag{4.3}$$

$$\Gamma_i^{(1)}(t+1) = \mathrm{arctanh}\left[\theta_{P^{-1}(i)}^{(2)}(t+1)\right] - \Gamma_{P^{-1}(i)}^{(2)}(t) - B_i^{(0)}, \tag{4.4}$$

$$\Gamma_i^{(2)}(t+1) = \mathrm{arctanh}\left[\theta_{P(i)}^{(1)}(t+1)\right] - \Gamma_{P(i)}^{(1)}(t) - B_{P(i)}^{(0)}, \tag{4.5}$$

where $\mathbf{B}^{(m)} \equiv (B(J_1^{(l)\rm out}),\ldots,B(J_N^{(l)\rm out}))$, $l = 0,1,2$ and $B_i^{(0),P} \equiv B_{P(i)}^{(0)}$.

The meaning of the previous equations is the following. The θ_i are expectation values of a sequence of operators which can take only values ±1, computed independently for every element of the sequence. The information contained in θ_i can therefore be represented by an "external field" Γ_i such that $\theta_i = \tanh\Gamma_i$. In order to avoid double counting of information one subtracts the external fields of the previous iteration as shown in equations (4.4, 4.5). Hopefully the iteration converges to a fixed point:

$$\lim_{t\to\infty}\theta_i^{(1)}(t) = \lim_{t\to\infty}\theta_{P^{-1}(i)}^{(2)}(t) \equiv \theta_i^*. \tag{4.6}$$

The decoded message is obtained as follows:

$$\tau_i^{\rm turbo} \equiv \mathrm{sign}(\theta_i^*). \tag{4.7}$$

The system described by equations (3.9, 3.10) is seen in turbo decoding as the union of two one-dimensional subsystem. Each subsystem acts on the other one through a magnetic field (in the non recursive case) or through an additional coupling (in the recursive case).

To get some insight of equations (4.2–4.5) we define the free energy functionals $F^{(1)}$ and $F^{(2)}$:

$$\mathcal{Z}^{(1)}[\boldsymbol{\Gamma}] = \sum_{\boldsymbol{\sigma}}\exp\left\{\sum_{i=1}^{N}(B(J_i^{(0)}) + \Gamma_i)\prod_{j=0}^{r}(\sigma_{i-j})^{\kappa(j;1)} + \sum_{i=1}^{N}B(J_i^{(1)})\prod_{j=0}^{r}(\sigma_{i-j})^{\kappa(j;2)}\right\}, \tag{4.8}$$

$$\mathcal{Z}^{(2)}[\boldsymbol{\Gamma}] = \sum_{\boldsymbol{\sigma}}\exp\left\{\sum_{i=1}^{N}(B(J_{P(i)}^{(0)}) + \Gamma_{P(i)})\prod_{j=0}^{r}(\sigma_{i-j})^{\kappa(j;1)} + \sum_{i=1}^{N}B(J_i^{(2)})\prod_{j=0}^{r}(\sigma_{i-j})^{\kappa(j;2)}\right\}, \tag{4.9}$$

$$F^{(m)}[\boldsymbol{\theta}] \equiv \sum_{i=1}^{N}\theta_i\Gamma_i - \log\left(\mathcal{Z}^{(m)}[\boldsymbol{\Gamma}]\right)\Big|_{\theta_i=\frac{\partial\log(\mathcal{Z}^{(m)})}{\partial\Gamma_i}}. \tag{4.10}$$

It is then simple to show that θ^* is a solution of the equation:

$$\frac{\partial}{\partial\theta_i}\mathcal{F}^{\rm turbo}[\boldsymbol{\theta}] = 0, \tag{4.11}$$

where

$$\mathcal{F}^{\rm turbo}[\boldsymbol{\theta}] \equiv F^{(1)}[\boldsymbol{\theta}] + F^{(2)}[\boldsymbol{\theta}] - F_0[\boldsymbol{\theta}]\,, \qquad (4.12)$$

$$F_0[\boldsymbol{\theta}] \equiv \sum_{i=1}^{N} \left\{ -B_i^{(0)}\theta_i - s(\theta_i) \right\}, \qquad (4.13)$$

$$s(x) \equiv -\left(\frac{1+x}{2}\right)\log\left(\frac{1+x}{2}\right) - \left(\frac{1-x}{2}\right)\log\left(\frac{1-x}{2}\right). \qquad (4.14)$$

Equation (4.12) is an approximation to the true free energy functional of the total system which is given by:

$$\mathcal{Z}[\boldsymbol{\Gamma}] \equiv \sum_{\boldsymbol{\sigma}^{(1)}} \sum_{\boldsymbol{\sigma}^{(2)}} \prod_{i=1}^{N} \delta(\epsilon_{P(i)}(\boldsymbol{\sigma}^{(1)}), \epsilon_i(\boldsymbol{\sigma}^{(2)})) \times \exp\left\{ -H(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}; \mathbf{J}^{\rm out}) + \sum_{i=1}^{N} \Gamma_i \epsilon_i(\boldsymbol{\sigma}^{(1)}) \right\}, \qquad (4.15)$$

$$\mathcal{F}[\boldsymbol{\theta}; \mathbf{J}^{(0)}, \mathbf{J}^{(1)}, \mathbf{J}^{(2)}] \equiv \sum_{i=1}^{N} \theta_i \Gamma_i - \log\left(\mathcal{Z}[\boldsymbol{\Gamma}]\right)\Big|_{\theta_i = \frac{\partial \log(\mathcal{Z})}{\partial \Gamma_i}}, \qquad (4.16)$$

where $H(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}; \mathbf{J}^{\rm out})$ is given in equation (3.10). It is then evident that

$$\mathcal{F}^{\rm turbo}[\boldsymbol{\theta}] = \mathcal{F}[\boldsymbol{\theta}; \mathbf{J}^{(0)}, \mathbf{J}^{(1)}, \mathbf{0}] + \mathcal{F}[\boldsymbol{\theta}; \mathbf{J}^{(0)}, \mathbf{0}, \mathbf{J}^{(2)}] - \mathcal{F}[\boldsymbol{\theta}; \mathbf{J}^{(0)}, \mathbf{0}, \mathbf{0}] \qquad (4.17)$$

i.e. turbo decoding neglects terms of order $B(J_{i_1}^{(1)})B(J_{i_2}^{(2)})$.

5 Replica approach

We would like to compute the error probability per bit for turbo codes. As explained in the introduction, in the case of a symmetric transmission channel, it is enough to compute the magnetization in the case of all inputs $\tau_i = 1$. The error probability per bit is given by the probability of a local magnetization being negative. The similarity of the Hamiltonian (3.10) with the Hamiltonians of disordered spin systems is obvious. The disorder in the case of turbo codes has two origins. One is due to the (random) permutation which defines the particular code. The other is more conventional and is related to the randomness of the couplings which is due to the transmission noise. As usual in disordered systems, we can only compute the average over disorder and for that we have to introduce replicas.

Let us define the expectation value of the operator $\epsilon_i(\boldsymbol{\sigma})$ defined in equation (3.6) with respect to the probability distribution given by equations (3.9, 3.10):

$$\Theta_i[\mathbf{J}^{\rm out}, P] \equiv \sum_{\boldsymbol{\sigma}^{(1)}} \sum_{\boldsymbol{\sigma}^{(2)}} \epsilon_i(\boldsymbol{\sigma}^{(1)}) \mathcal{P}(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)} | \mathbf{J}^{\rm out}). \qquad (5.1)$$

The statistical properties of a turbo code can be derived from the probability distribution of this expectation value:

$$\mathcal{P}_i(\theta|P) \equiv \int \mathrm{d}Q[\mathbf{J}^{\rm out}]\, \delta\left(\theta - \Theta_i[\mathbf{J}^{\rm out}, P]\right), \quad i = 1, \ldots, N, \qquad (5.2)$$

where

$$\mathrm{d}Q[\mathbf{J}^{\rm out}] = \prod_{l=0}^{2} \prod_{i=1}^{N} Q(J_i^{(l),{\rm out}}| + 1)\mathrm{d}J_i^{(l),{\rm out}}. \qquad (5.3)$$

Then we define the average distribution

$$\overline{\mathcal{P}}(\theta) \equiv \frac{1}{N!} \sum_{P} \int \mathrm{d}Q[\mathbf{J}^{\rm out}]\, \delta\left(\theta - \Theta_i[\mathbf{J}^{\rm out}, P]\right), \qquad (5.4)$$

where the sum runs over all possible permutations. $\overline{\mathcal{P}}(\theta)$ is expected not to depend upon the site i in the thermodynamic limit ($N \to \infty$). The average error probability per bit is given by

$$\overline{P_{\rm e}} \equiv \int_{-\infty}^{0} \mathrm{d}\theta\, \overline{\mathcal{P}}(\theta). \qquad (5.5)$$

In any case $\overline{P_{\rm e}}$ is an upper bound for the error probability of the "best" code (*i.e.* the one built with the permutation which yields the lowest error probability).

The replicated partition function is given by:

$$\overline{Z^n} \equiv \frac{1}{N!} \sum_{P} \int \mathrm{d}Q[\mathbf{J}^{\rm out}] \times \sum_{\{\boldsymbol{\sigma}^{(1),a}\}} \sum_{\{\boldsymbol{\sigma}^{(2),a}\}} \prod_{a=1}^{n} \prod_{i=1}^{N} \delta(\epsilon_{P(i)}(\boldsymbol{\sigma}^{(1),a}), \epsilon_i(\boldsymbol{\sigma}^{(2),a})) \times \exp\left\{ -\sum_{a=1}^{n} H(\boldsymbol{\sigma}^{(1),a}, \boldsymbol{\sigma}^{(2),a}; \mathbf{J}^{\rm out}) \right\}. \qquad (5.6)$$

In order to compute the average over permutations it is convenient to use the occupation densities formalism, introduced by Monasson [15]. The occupation densities $c_m(\underline{\epsilon})$ are defined as the normalized fraction of sites such that $\epsilon_i(\boldsymbol{\sigma}^{(m),a}) = \epsilon^a$ for $a = 1, \ldots, n$:

$$c_m(\underline{\epsilon}) \equiv \frac{1}{N} \sum_{i=1}^{N} \delta_{\underline{\epsilon}, \underline{\epsilon}_i^{(m)}}, \qquad (5.7)$$

where $\underline{\epsilon} \equiv (\epsilon^1, \ldots, \epsilon^a, \ldots, \epsilon^n) \in \{-1, +1\}^n$, $\underline{\epsilon}_i^{(m)} \equiv (\epsilon_i(\boldsymbol{\sigma}^{(m),1}), \ldots, \epsilon_i(\boldsymbol{\sigma}^{(m),a}), \ldots, \epsilon_i(\boldsymbol{\sigma}^{(m),n}))$ are replicated spin variables and

$$\delta_{\underline{\epsilon}_1, \underline{\epsilon}_2} \equiv \prod_{a=1}^{n} \delta_{\epsilon_1^a, \epsilon_2^a}. \qquad (5.8)$$

The average over permutations is done in the Appendix A. The resulting replicated partition function reads:

$$\overline{Z^n} \equiv \int \mathrm{d}Q[\mathbf{J}^{\mathrm{out}}] \sum_{\{\boldsymbol{\sigma}^{(1),a}\}} \sum_{\{\boldsymbol{\sigma}^{(2),a}\}} \prod_{\underline{\epsilon}} \delta_{Nc_1(\underline{\epsilon}),Nc_2(\underline{\epsilon})} \times \exp\left\{-\sum_{a=1}^{n} H(\boldsymbol{\sigma}^{(1),a},\boldsymbol{\sigma}^{(2),a};\mathbf{J}^{\mathrm{out}}) + N\sum_{\underline{\epsilon}} c_1(\underline{\epsilon})\log c_1(\underline{\epsilon})\right\}. \tag{5.9}$$

The constraint enforced by the Kronecker delta functions in equation (5.6), *i.e.* the coupling of the two chains, is written in terms of the occupation densities. The occupation densities lack any spatial structure and the interaction between the two chains is of the mean field type. Notice that, unlike in mean field models, the replica trick does not reduce the problem to a single site one.

The occupation densities play the role of the order parameter and the partition function can be written as an integral over them. It is possible to find a no error saddle point of the c integral in equation (5.9) and to study its local stability.

We briefly report here the main results of this approach for the Gaussian channel described by equation (1.2). A detailed analysis will be presented elsewhere [16]. For recursive turbo codes there exists a low noise phase $w^2 < w_c^2$ where the error probability vanishes in the thermodynamic limit (*i.e.* for infinitely long sequences). In this phase the model is completely ordered:

$$\overline{\mathcal{P}}(\theta) = \delta(\theta - 1). \tag{5.10}$$

The local stability analysis yields the critical value w^2_{loc} such that for $w^2 > w^2_{\mathrm{loc}}$ the no-error phase is destroyed by small fluctuations. Clearly $w^2_{\mathrm{loc}} \geq w_c^2$. We computed w^2_{loc} for the two cases listed below. For both the rate is $R = 1/3$ so that the Shannon noise threshold as given by equation (1.3) is $w^2_{\mathrm{Shannon}} = 1/(2^{2/3} - 1) \simeq -2.31065$ dB. Error free communication can take place only for $w^2 < w^2_{\mathrm{Shannon}}$.

- For model (a), defined by equations (2.4, 2.5) one gets $w^2_{\mathrm{loc}} = 1/\ln 4 \simeq 1.41855$ dB.
- For model (b), defined by equations (2.6, 2.7) one obtains $w^2_{\mathrm{loc}} = -1/(2\ln x_c)$ where $x_c \simeq 0.741912\ldots$ is the only real solution of the equation

$$2x^5 + x^2 = 1 \tag{5.11}$$

The resulting value $w^2_{\mathrm{loc}} \simeq -2.23990$ dB is quite near to the Shannon threshold.

6 Discussion

We formulated turbo codes as a spin model Hamiltonian and we obtained new results using the replica method. It is well known that this method is not mathematically rigorous. So it is natural to question the validity of our results.

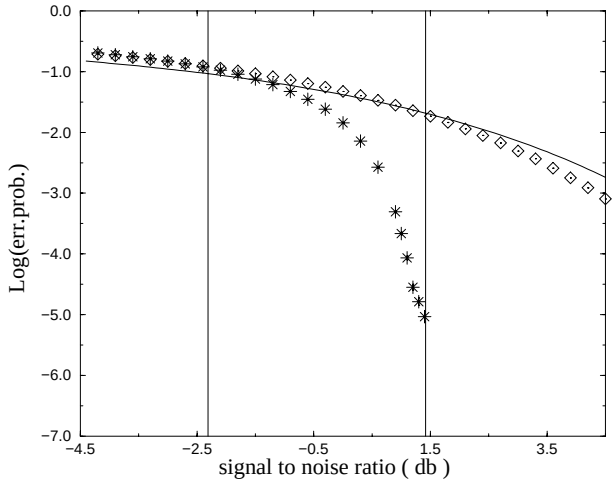

Fig. 4. Numerical results for the error probability per bit of the recursive turbo code built from the convolutional code (a) (*cf.* Eqs. (2.4, 2.5)). Stars ($*$) refer to the turbo code, diamonds ($\Diamond$) to the convolutional code obtained by setting the permutation P equal to the identity permutation, and the continuous line to the uncoded message. The leftmost vertical line is located at the Shannon threshold ($w^2 = w^2_{\mathrm{Shannon}}$) while the rightmost at the threshold of local stability ($w^2 = w^2_{\mathrm{loc}}$, see Sect. 5).

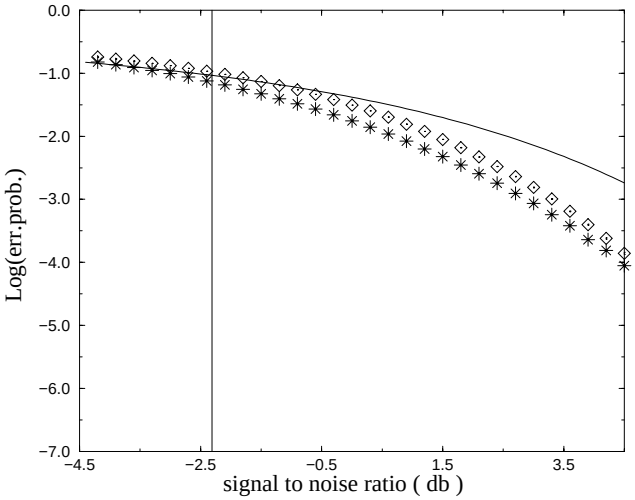

Fig. 5. Numerical results for the error probability per bit of the non recursive turbo code built from the convolutional code (a) (*cf.* Eqs. (2.4, 2.5)). The symbols have the same meaning explained in the caption of Figure 4.

For this purpose we have carried out numerical simulations of the following codes: the recursive turbo code corresponding to the convolutional code (a) of Section 2, its error probability is reported in Figure 4; the non recursive turbo code obtained by permuting the generating polynomials of the previous one (see Fig. 5); the recursive turbo code corresponding to the code (b) of the same section (see Fig. 6). We used the Berrou *et al.* turbo decoding algorithm and averaged over 200 to 500 realizations of the disorder.

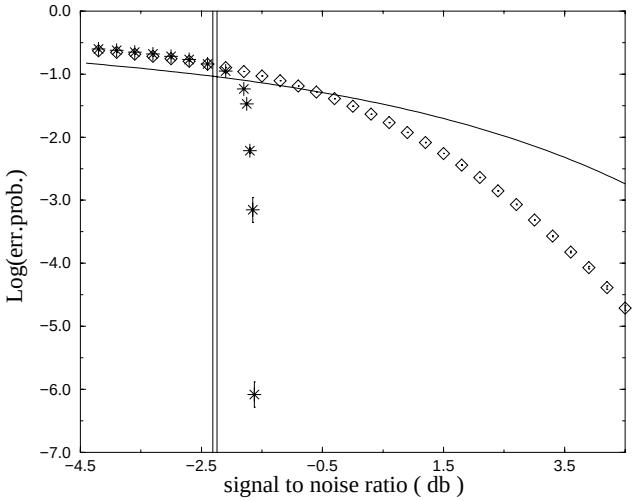

Fig. 6. Numerical results for the error probability per bit of the recursive turbo code built from the convolutional code (b) (*cf.* Eqs. (2.6, 2.7)). For the meaning of various symbols see the caption of Figure 4. Notice that in this case $w^2_{\rm loc} \sim w^2_{\rm Shannon}$.

The first conclusion is that recursive turbo codes are much better codes than non recursive ones. Furthermore our results for recursive turbo codes are compatible with the existence of a threshold $w^2_{\rm c}$ such that for $w^2 < w^2_{\rm c}$ the error probability per bit is zero, while no such threshold seems to exist for non recursive codes. This is in agreement with replica theory. Zero error probability can only be achieved in the $N \to \infty$ limit. Our simulations are for $N = 10^5$. It would be interesting to perform a detailed study of finite size corrections, *i.e.* of the N dependence of the error probability per bit.

We now discuss the numerical value of the noise threshold w^2_c. The first remark is that both numerically and analytically, the critical value is below Shannon's bound and that it depends on the convolutional code (*i.e.* on the generating polynomials). The second remark is that the analytical value of threshold, $w^2_{\rm loc} \simeq 1.4186$ dB is in very good agreement with the numerical value for the code (a). For code (b) $w^2_{\rm loc} \simeq -2.240$ dB while one gets $w^2_{\rm c} \simeq -1.7$ dB from the simulations. It would be interesting to understand this disagreement. As we said in the previous section, $w_{\rm loc}$ was calculated by a local stability analysis of the ordered phase, *i.e.* we assumed that the transition is of second order. A possible explanation would be that the transition is second order for code (a) and first order for code (b). Numerical results seem to support this hypothesis, as the variation of the error probability as a function of noise is much sharper in case (b). However a much more careful analysis of finite size effects is necessary in order to settle this question numerically. One should also look analytically for the occurrence of a first order transition.

Turbo decoding is an approximate implementation of temperature one decoding. It can be easily generalized to arbitrary temperature decoding. An issue of practical relevance is the behavior of these alternative decoding strategies. This behavior is determined by the phase structure of the model (3.10).

After the completion of this work we learned that Richardson and Urbanke obtained some similar results by completely different methods [17].

Appendix A: The replicated partition function

In this appendix we prove the following identity which holds in the thermodynamic ($N \to \infty$) limit:

$$\frac{1}{N!}\sum_P \prod_{i=1}^N \delta_{\underline{\epsilon}^{(1)}_{P(i)},\underline{\epsilon}^{(2)}_i} = \exp\left\{N\sum_{\underline{\epsilon}} c_1(\underline{\epsilon})\log c_1(\underline{\epsilon})\right\} \times \prod_{\underline{\epsilon}} \delta_{Nc_1(\underline{\epsilon}),Nc_2(\underline{\epsilon})}, \quad \text{(A.1)}$$

where the $\underline{\epsilon} \equiv (\epsilon^1,\ldots,\epsilon^a,\ldots,\epsilon^n) \in \{-1,+1\}^n$ are replicated spin variables and

$$c_m(\underline{\epsilon}) \equiv \frac{1}{N}\sum_{i=1}^N \delta_{\underline{\epsilon},\underline{\epsilon}^{(m)}_i}. \quad \text{(A.2)}$$

If we set $\underline{\epsilon}^{(m),a}_i = \epsilon_i(\boldsymbol{\sigma}^{(m),a})$ the identity (A.1) allows to obtain the replicated partition function (5.9) from equation (5.6).

The l.h.s. of equation (A.1) is non vanishing only if the (replicated) spin variables $\underline{\epsilon}^{(1)}$ are a permutation of the $\underline{\epsilon}^{(2)}$. If this happens then the number of sites i such that $\underline{\epsilon}^{(1)}_i$ takes a given value $\underline{\epsilon}$ is the same as the number of sites j such that $\underline{\epsilon}^{(2)}_j$ takes the same value. In other words $Nc_1(\underline{\epsilon}) = Nc_2(\underline{\epsilon})$ for any $\underline{\epsilon} \in \{-1,+1\}^n$. It is clear that also the converse is true. If, for any $\underline{\epsilon} \in \{-1,+1\}^n$, $\underline{\epsilon}^{(1)}_i = \underline{\epsilon}$ as many times as $\underline{\epsilon}^{(2)}_j = \underline{\epsilon}$ for $i,j = 1,\ldots,N$ then there exist a permutation P such that $\underline{\epsilon}^{(1)}_{P(i)} = \underline{\epsilon}^{(2)}_i$ for all the $i = 1,\ldots,N$. We can rewrite the l.h.s. of equation (A.1) as follows

$$\frac{1}{N!}\,(\text{nb. of permutations } P \text{ s.t. } \underline{\epsilon}^{(1)}_{P(i)} = \underline{\epsilon}^{(2)}_i \text{ for } 1 \le i \le N) \prod_{\underline{\epsilon}} \delta_{Nc_1(\underline{\epsilon}),Nc_2(\underline{\epsilon})}. \quad \text{(A.3)}$$

We have seen that there exists at least one permutation such that $\underline{\epsilon}^{(1)}_{P(i)} = \underline{\epsilon}^{(2)}_i$ for $1 \le i \le N$ if the constraint in equation (A.3) is satisfied. What is the number of these permutations? Let us fix one of them, say $\hat{P}$. We have that $\underline{\epsilon}^{(1)}_{\hat{P}(i)} = \underline{\epsilon}^{(2)}_i$. We are still free to permute any two sites i and j such that $\underline{\epsilon}^{(1)}_i = \underline{\epsilon}^{(1)}_j$. More precisely, if P' is such that $\underline{\epsilon}^{(1)}_{P'(i)} = \underline{\epsilon}^{(1)}_i$ for any $i = 1,\ldots,N$, then $\tilde{P} \equiv P'\hat{P}$ satisfies $\underline{\epsilon}^{(1)}_{\tilde{P}(i)} = \underline{\epsilon}^{(2)}_i$. In fact $\underline{\epsilon}^{(1)}_{\tilde{P}(i)} = \underline{\epsilon}^{(1)}_{P'\hat{P}(i)} = \underline{\epsilon}^{(1)}_{\hat{P}(i)} = \underline{\epsilon}^{(2)}_i$. In this manner we can construct all the permutations we are interested in.

The permutation P' can be constructed as follows. For each $\underline{\epsilon} \in \{-1,+1\}^n$ we choose a permutation among

the $Nc_2(\underline{\epsilon})$ sites i such that $\underline{\epsilon}_i^{(2)} = \underline{\epsilon}$. We construct P' as the product of these 2^n permutations: the order does not matter since permutations which act on different sites commute. The number of permutation we are looking for is then the product of a factor for each $\underline{\epsilon} \in \{-1,+1\}^n$. The factor associated to a given $\underline{\epsilon}$ is the number of permutations of $Nc_2(\underline{\epsilon})$ objects. We can then rewrite equation (A.3) as

$$\frac{1}{N!}\prod_{\underline{\epsilon}}(Nc_2(\underline{\epsilon}))!\prod_{\underline{\epsilon}}\delta_{Nc_1(\underline{\epsilon}),Nc_2(\underline{\epsilon})}. \tag{A.4}$$

Using the Stirling formula and the property $\sum_{\underline{\epsilon}} c_2(\underline{\epsilon}) = 1$ and keeping track of the leading term for $N \to \infty$ we get equation (A.1).

References

1. C. Berrou, A. Glavieux, P. Thitimajshima, *Proc. 1993 IEEE Int. Conf. Commun. (Geneva, Switzerland, 1993)*, pp. 1064–1070.
2. N. Sourlas, Nature **339**, 693 (1989).
3. N. Sourlas, *Statistical Mechanics of Neural Networks*, Lecture Notes in Physics **368**, edited by L. Garrido (Springer Verlag, 1990).
4. N. Sourlas, École Normale Supérieure preprint (April, 1993).
5. N. Sourlas, *From Statistical Physics to Statistical Inference and Back,* edited by P. Grassberger, J.-P. Nadal (Kluwer Academic, 1994), p. 195.
6. A.J. Viterbi, IEEE Trans. Com. Technology **COM-19**, 751 (1971).
7. L. Bahl, J. Cocke, F. Jelinek, J. Raviv, IEEE Trans. Inf. Theory **IT-20**, 248 (1974).
8. P. Ruján, Phys. Rev. Lett. **70**, 2968 (1993).
9. N. Sourlas, Europhys. Lett. **25**, 159 (1994).
10. H. Nishimori, J. Phys. Soc. Jpn **62**, 2973 (1993).
11. H. Nishimori, J. Phys. C **13**, 4071 (1980).
12. B. Derrida, Phys. Rev. B **24**, 2613 (1981).
13. Y. Kabashima, D. Saad, Europhys. Lett. **45**, 97 (1999).
14. R. Vicente, D. Saad, Y. Kabashima, Phys. Rev. E **60**, 5352 (1999).
15. R. Monasson, J. Phys. A **31**, 513 (1998).
16. A. Montanari, Eur. Phys. J. B **18**, 121 (2000).
17. T. Richardson, private communication.