Memory and Optimisation in Neural Network Models

Bruce McLean Forrest

Submitted for the degree of

Doctor of Philosophy

Department of Physics

University of Edinburgh

August 1988

To Mum, Dad and Jean,

and to the memory of Elizabeth Gardner.

Declaration

All of the work in this thesis was my own except where otherwise indicated. Some of the work has been published in

B M Forrest 1987 *J. Phys. A: Math. Gen.* **21** 245

B M Forrest 1988 in *Parallel Architectures and Computer Vision* ed I Page (Oxford University Press)

B M Forrest, D Roweth, N Stroud, G V Wilson and D J Wallace 1987 *The Computer Journal* **30** 413

B M Forrest, D Roweth, N Stroud, G V Wilson and D J Wallace 1987 *Preprint* Edinburgh 87/419 to appear in *Parallel Computing*

Bruce M. Forrest

8/8/88

Acknowledgements

My work at Edinburgh was supported by the Science and Engineering Research Council and G.E.C. Research Ltd.

I am particulary indebted to Elizabeth Gardner for her invaluable help and discussions with the work which appears in chapters 3 and 4. I am also grateful to David Wallace for his helpful suggestions, discussions, direction and encouragement and also to Alistair Bruce for useful comments and his encouragement.

For the work of chapter 5, I would also like to thank Bernard Buxton and David Murray of G.E.C. Research for their suggestions and comments.

Last, but not least, I greatly appreciate the friendship of my colleagues in the Physics department which has helped make my time in Edinburgh a most enjoyable one. In particular I would like to thank: Raúl Toral (un amigo inolvidable), Frank Śmieja, Shara Amin, Marc Hewitt, Andrew Canning, Gareth Richards, David Stephenson, Lyndon Clarke and Colin Farquhar.

Abstract

A numerical study of two classes of neural network models is presented.

The performance of Ising spin neural networks as content-addressable memories for the storage of bit patterns is analysed. By studying systems of increasing sizes, behaviour consistent with fintite-size scaling, characteristic of a first-order phase transition, is shown to be exhibited by the basins of attraction of the stored patterns in the Hopfield model. A local iterative learning algorithm is then developed for these models which is shown to achieve perfect storage of nominated patterns with near-optimal content-addressability. Similar scaling behaviour of the associated basins of attraction is observed. For both this learning algorithm and the Hopfield model, by extrapolating to the thermodynamic limit, estimates are obtained for the critical minimum overlap which an input pattern must have with a stored pattern in order to successfully retrieve it.

The role of a neural network as a tool for optimising cost functions of binary-valued variables is also studied. The particular application considered is that of restoring binary images which have become corrupted by noise. Image restorations are achieved by representing the array of pixel intensities as a network of analogue neurons. The performance of the network is shown to compare favourably with two other deterministic methods—a gradient descent on the same cost function and a majority-rule scheme—both in terms of restoring images and in terms of minimising the cost function.

All of the computationally intensive simulations exploit the inherent parallelism in the models: both SIMD (the ICL DAP) and MIMD (the Meiko Computing Surface) machines are used.

Nachdem es uns nicht gelungen ist, Spingläser mit Hilfe unseres Gehirns zu verstehen, versuchen wir jetzt unser Gehirn mit Hilfe von Spingläsern zu verstehen.

(After not having succeeded in understanding spin glasses with the help of our brain, we are now trying to understand our brain with the help of spin glasses.)

Anon. (Heidelberg 1986)

Contents

List of Figures

Chapter 1

Introduction to Neural Networks

The tasks which the human brain carries out routinely and apparently effortlessly require the processing of massive amounts of information. For example, the recognition of patterns (e.g., words or faces) involves a choice from amongst an enormous data-base containing tens or hundreds of thousands of alternatives. Indeed the recognition of a particular visual object out of a class of similar objects firstly requires interpretation of a two-dimensional array of light intensities—the retinal images—in terms of the three-dimensional scene which is being viewed: a phenomenal information processing feat in itself. On conventional computers of today, where instructions can be executed in nanoseconds, such processing tasks would typically require many thousands, or even millions, of instructions—if they can even be achieved at all. Yet these tasks can be carried out in the brain in fractions of a second, although the 'building block' of the brain—the *neuron*—is a nerve cell which 'operates' (changes its state of activity) merely in times of the order of milliseconds.

How is it that a collection of interconnected neurons—'components' which are slow on the timescale of present day integrated circuit components—can achieve such tremendous processing powers? Through the study of systems of interacting processing elements, neural network models attempt to go some way towards

providing answers to this question and to come close to emulating at least some of the processing capabilities of their real (biological) counterparts. At a more ambitious level, they would endeavour to explain the functioning of parts of the brain itself.

It is widely agreed that the emergent processing speed available to the brain must depend on a massive degree of parallelism. At any given instant a huge number of neurons simultaneously alter their state in response to each other. In light of the timescales mentioned above, the way in which information is stored and retrieved must also involve parallelism to a large extent: the recognition of a pattern from amongst perhaps hundreds of thousands of contenders cannot involve purely a sequential search through all the possible alternatives stored in memory, but must somehow involve a simultaneous 'consideration' of a huge number of possibilities.

Some 10^{10} neurons are contained in the brain, each of which is a complex entity in its own right. A neuron can be roughly described as being composed of a cell body which fires electrochemical pulses along its output fibre, the *axon*. The rate at which it does so is largely, if not completely, determined by the electrical potential of the cell. Connected to the axon are *synapses*—connections to the input fibres (*dendrites*) into other neurons. When a neuron fires, the electrochemical activity which propagates along its axon induces further electrical pulses to propagate along these input fibres via the synapses. These then in turn modify the electrical potential of the neurons into which they are connected. The incoming electrical signal along a synapse can either tend to increase or decrease the cell potential of the neuron: i.e., synapses can be either excitatory or inhibitory, respectively. The number of neurons which are connected in this way to another neuron can typically be of the order of 10^4, implying a population of some 10^{14} synapses in the brain.

Although many of the brain's functions have been successfully localised to par-

ticular areas within it, the understanding of the neural circuitry and of the 'algorithms' which arise remains largely a mystery. The vast number of neurons participating, each interacting with hundreds or thousands of others, makes the solving of this problem a daunting prospect.

Nevertheless, the underlying philosophy of most neural network models is that it may be possible to account for many of their capabilities in terms of the overall structure of the network and that they could be greatly independent of many of the complex details at the level of the individual neuron and its interactions. This is somewhat akin to the argument that the possible algorithms able to be executed by a computer would be unlikely to be discovered if one merely identified the behaviour of the individual electronic components of which it was constructed. This is also analogous to many phenomena encountered in physics—particularly in statistical physics—where the macroscopic, or bulk, behaviour of a system comprising a large number of interacting subsytems can be to a large extent explained by models in which only the barest details of the subsystems are required. For example, the way in which the net magnetisation or susceptibility of a ferromagnet varies with temperature and applied magnetic field; and the relationship between the pressure, temperature and density of many gases: near the critical point both can be successfully approximated by models which pay no great attention to the particular variety of their component atoms or molecules. Other examples include the ferromagnetic-paramagnetic critical behaviour of magnetic materials and the gas-liquid phase transition of substances: both can be accurately accounted for using models in which the component atoms or molecules are described by merely one number (a spin being 'up'/'down' in the former case and an molecule being present/not present in the latter).

The models which shall be considered in this thesis concern neural networks functioning as a memory for storage and fast retrieval of information (chapters 2, 3 and 4) and as an optimisation tool (chapter 5). The memory models attempt

to explain how a neural network can function as a *content-addressable* memory. A 'fact', or 'picture', which is stored in such a memory is retrieved by supplying a cue which contains a sufficiently large portion of that fact in order to distinguish it from the other entries stored in the memory: recall is based on addressing the entries by their content and not, as in conventional computers, by providing some address which points to a particular location.

Both types of model simplify the behaviour of a neuron so that it can be described by one real variable: the rate, V_i, at which it fires (where i denotes a particular neuron). The response function which governs this firing rate in terms of the cell potential, ϕ_i, of the neuron is approximated by a sigmoidal form:

$$V_i(\phi_i) = \frac{1}{1 + \exp\{-g(\phi_i - U_i)\}}, \tag{1.1}$$

where g is a parameter which determines the steepness of the response function and U_i is the 'threshold' of neuron i—it has the effect of shifting the response curve along the ϕ_i axis. $V_i = 0$ corresponds to a non-firing (or quiescent) state, while $V_i = 1$ denotes fully-firing.

The effect of the synapses impinging on a neuron are modelled as follows. A synaptic connection from the output (axon) of the jth neuron into the ith neuron has some synaptic strength T_{ij} which can be positive (excitatory) or negative (inhibitory). If neuron j is firing pulses at a rate V_j then it will increase the cell potential ϕ_i of neuron i by an amount $T_{ij}V_j$. The net potential $\phi_i(t)$ at time t is then given by

$$\phi_i(t) = \sum_j T_{ij} V_j(t). \tag{1.2}$$

Thus the very complex behaviour at the microscopic level—that of an individual neuron—has now been simplified to

$$V_i(t+1) = \frac{1}{1 + \exp\{-g(\phi_i(t) - U_i)\}} \quad \text{with} \quad \phi_i(t) = \sum_j T_{ij} V_j(t) \tag{1.3}$$

i.e., the state of a neuron is a non-linear function of the sum of its inputs. The firing rates $\underline{V}(t) \equiv \{ V_1(t), V_2(t), \dots \}$ of the neurons at time t can be thought of as a firing 'pattern' of the network.

Still to be specified are the architecture of the network—which neurons are connected to which—and the order in which the neurons alter their state (update): e.g., all the neurons could update in lockstep parallel; or only one neuron could be changing its state at a given time (sequential updating); or neurons could update in a random fashion, with possibly many of them changing state at any given instant (asynchronous updating). The models which shall be considered in chapters 2 to 4 are in general fully connected, i.e., each neuron can in principle influence and be influenced by any other. Those in chapter 5 can also be fully connected, but the particular example which is considered only has nearest-neighbour connections in a two-dimensional lattice. As for the order in which the neurons are updated, although the asynchronous method is probably the most similar to the situation in a real neural network, one would not expect many of the emergent processing capabilities to be dependent on the particular choice. The memory models of chapters 2, 3 and 4 utilise sequential updating, while the optimisation networks in chapter 5 are of a synchronous nature: half of the neurons are being updated in parallel.

It is principally the fully-connected nature and the fact that all of the neurons play more or less identical roles in these models which distinguish them from another class of models that have also been the subject of a great deal of attention in recent years. In *multi-layered networks*, the neurons are arranged in successive layers from an 'input' layer through a number of 'hidden' layers to an 'output' layer. Connections only exist from one layer to the next and only in the forward (input to output) direction. This implies a connectivity matrix of an upper-triangular form. All the neurons in one layer are updated in parallel, one layer at a time, starting with the input layer and ending with the output layer. In this way a certain firing pattern presented at the input layer will propagate through

the network and induce a firing pattern at the output layer.

Whereas in the multi-layered networks one is concerned that certain input patterns give rise (associate) to desired output patterns, in the models studied here the firing patterns of interest are the dynamically stable ones:

$$V_i(t') = V_i(t), \quad \forall i, \ \forall t' > t. \tag{1.4}$$

These are the persistent activity patterns which are identified as being 'stored' in the memory models or which correspond to solutions (minima) of a cost function in the optimisation networks.

Chapter 2 introduces a class of models in which the behaviour of a neuron is simplified even further so that it can be represented by one boolean variable: either it is firing or it is not. This essentially corresponds to taking the limit of $g \to \infty$ in (1.1), i.e., reducing the response function to a step function. The strong analogy with an assembly of interacting spin-$\frac{1}{2}$ Ising variables is described. It is this which has allowed the analysis of these models in a thermodynamical framework and which has been principally responsible for the upsurge of interest in these models amongst the physics community. The functioning of these models as distributed content-addressable memories is also explained.

Chapter 3 deals with a particular model which belongs to this class: the Hopfield model. It was in his seminal paper (Hopfield 1982) that Hopfield demonstrated the existence of a Hamiltonian, or energy, function for the network. This then allowed subsequent analysis of the model using the powerful techniques of statistical mechanics. A numerical study of the content-addressability of the model is presented and evidence for scaling and critical behaviour is obtained.

Unlike the Hopfield model in chapter 3, which prescribes the synaptic connection strengths according to a specific rule, chapter 4 concerns 'learning algorithms'. These are iterative improvement techniques for modifying the synapses ('learning') until the desired results (proper storage of designated bit patterns) are

achieved. Once again the content-addressability of these models is analysed numerically and similar, but improved, behaviour to that of the Hopfield model is found.

Finally, in chapter 5, the role of a neural network as an optimisation tool is considered. Whereas in the memory models the problem is to find an appropriate set of synaptic connections such that designated firing patterns are stored, in the optimisation network the connections are provided by the cost function under consideration and the task is to then find a firing pattern which corresponds to an optimal solution for the cost function. The problem to which the technique is applied is that of restoring binary images which have been corrupted by noise.

The simulation of a neural network of a reasonable size demands powerful computational resources as each neuron must in general communicate with every other one in a given updating sweep and normally many such sweeps are required before the whole network reaches stability. The computationally intensive demands of all the simulations throughout this thesis were met by implementing the models on parallel processors: the ICL DAP (chapters 3, 4 and 5) and a Meiko Computing Surface; an array of transputers (chapter 5). A description of these powerful resources is provided in chapters 3 (the DAP) and 5 (the Computing Surface). Details of the particular implementations are provided in the appendices.

Chapter 2

Ising Spin Neural Network Models

2.1 Introduction

In these models the behaviour of a neuron is simplified to the case where it can exist only two possible states of activity: either it can fire electrical pulses along its axon at maximum frequency, or it is quiescent and emits no signals. The neuron is fully-firing only if its current cell potential exceeds some threshold value (which can in principle be different for each neuron). This simplification not only has the advantage of making the whole model of the network more tractable analytically (or at least less intractable), but also has the consequence that each neuron can be represented by a single boolean variable—an obvious convenience for computer simulation of these models and for possible hardware implementation.

Mathematically, the two-state nature of a neuron is achieved by adopting the limit whereby the sigmoidal response function becomes infinitely steep as the cell potential approaches the neuron's threshold value: i.e., the response V_i is now related to the cell potential ϕ_i through the Heaviside threshold function θ:

$$V_i(\phi_i) = \theta(\phi_i - U_i), \tag{2.1}$$

where U_i is the neuron's threshold.

Each neuron being in only one of two possible states ("on" or "off") is reminiscent of a spin-$\frac{1}{2}$ Ising variable S_i in physics which can be "up" ($S_i = +1$) or "down" ($S_i = -1$). The analogy runs deeper than this, however. Changing notation to accommodate this by defining $S_i \equiv 2V_i - 1$ for the state of each (i'th) neuron, one then has

$$S_i = \begin{cases} +1 & \text{if neuron active} \\ -1 & \text{if neuron inactive} \end{cases} \tag{2.2}$$

and the response function is

$$S_i = sgn(\phi_i - U_i). \tag{2.3}$$

Recalling the form of the cell potential ϕ_i,

$$\phi_i = \sum_j T_{ij} V_j \tag{2.4}$$

and by transforming the otherwise arbitrary thresholds to

$$U_i \to 2U_i + \sum_j T_{ij}, \tag{2.5}$$

(1.3), in the limit of $g \to \infty$, becomes equivalent to

$$S_i = sgn(\sum_j T_{ij} S_j - U_i). \tag{2.6}$$

Thus each neuron can be viewed as an Ising spin which is "up" or "down" ($S_i = +1$ or -1) according to whether the "local field"

$$h_i = \sum_j T_{ij} S_j - U_i \tag{2.7}$$

is positive or negative. Hence, the dynamical law obeyed by each spin (neuron) is that it aligns itself with its local field h_i. Incorporating time, t, this is then

$$S_i(t+1) = sgn(h_i(t)) \quad \text{with} \quad h_i(t) = \sum_j T_{ij} S_j(t) - U_i. \tag{2.8}$$

The local field at site i (2.7) is composed of two parts: $-U_i$ is like an external (and site-dependent) field and $\sum_j T_{ij} S_j$ is the contribution to h_i from the other

spins S_j, mediated through their connections T_{ij} into site i. This is exactly analogous to having an effective magnetic field at lattice site i due to the other spins S_j which interact with spin S_i through bond strengths T_{ij}. One major difference, however, is that in the neural network models considered here each neuron can in principle interact with any other (so that the interactions T_{ij} are long-range), whereas in magnetic lattice models the interactions are typically restricted to some neighbourhood of each spin site.

In this framework an activity (firing) pattern of the network of N neurons at time t is represented by the spin configuration $\underline{S}(t) \equiv \{ S_1(t), S_2(t), \ldots, S_N(t) \}$ and those activity patterns which are "stored" in the memory of the network, i.e., the dynamically stable ones, correspond to those spin configurations in which every spin is already aligned with its local field:

$$S_i(t)h_i(t) > 0, \qquad i = 1, 2, \ldots, N. \tag{2.9}$$

Such stable configurations are the fixed points of the dynamics (2.8).

The neuron/spin analogy has resulted in the use of different terminology in the studies of these models. From here onwards the following terms shall be used interchangeably:

- neuron — node — spin
- synaptic connection strength — bond strength — weight
- network — system
- firing pattern — state of the network — spin configuration

The concept of memory in the model and the analogy of the collection of interacting neurons as a system of Ising spins can be made clearer if symmetry is imposed on the connection strengths:

$$T_{ij} = T_{ji}. \tag{2.10}$$

For, along with the no self-interaction constraint $T_{ii} = 0$, this implies the existence of an *energy* (or Lyapunov) function

$$E[\underline{S}] = -\frac{1}{2}\sum_{i,j} T_{ij} S_i S_j + \sum_i U_i S_i \tag{2.11}$$

such that under sequential updating (2.8) of the neurons, $E[\underline{S}]$ is monotonically decreasing. That this is the case can be seen by considering the change in energy due to the updating of the k'th neuron:

$$\begin{aligned} \Delta_k E &= E(t+1) - E(t) \\ &= -\left\{\sum_j T_{kj} S_j(t) - U_k\right\} \{S_k(t+1) - S_k(t)\} \\ &= -h_k(t) \{1 - S_k(t+1) S_k(t)\} S_k(t+1). \end{aligned} \tag{2.12}$$

But $S_k(t+1) = sgn(h_k(t))$, so $h_k(t) S_k(t+1) > 0$ and $1 - S_k(t+1) S_k(t) = 0$ or 2, thus $\Delta_k E \leq 0$.

This has the consequence that the stationary ("memorised") states $\underline{S}^*$ of the network correspond precisely to those which are local minima of the energy $E[\underline{S}]$, as they are the states which are stable to single spin flips (neuron state changes). As the network evolves, the state of the system $\underline{S}(t)$ traverses a downhill trajectory on this energy landscape $E[\underline{S}]$ which is defined over the 2^N corners of the N-dimensional hypercube containing all the possible states available to the system. The network will stabilise in a state which is a local minimum of E, and so a fixed point of the dynamics (2.8).

This picture provides a way of avoiding the need for running the network in order to discover the memory states (for a given set of T_{ij}): one "only" has to find those states which are local minima of E. So finding these states can be tackled as a statics problem as opposed to a dynamical one.

A stored state of the memory is then evoked if the initial state (the "cue" or "input") lies within its basin of attraction. The memory is *associative* as

it associates the final (stable) state with the initial state, and it is *content-addressable* because the stored item is recalled only if the initial state resembles it enough (contains enough of the stored state's content) to be in its basin of attraction.

The memory is also *distributed* since the storage of a particular item depends on the corresponding firing pattern being a stationary state, and this in turn depends in general on the values of all the synaptic connections T_{ij}. This makes for a memory robust to 'damage': if a few of the synapses are eliminated (set to zero) then the stability of previously stable patterns should not be affected to any great extent, i.e, the storage of patterns is not catastrophically disturbed. (This seems to be a characteristic of human memory too: memory representations, and perhaps other cognitive functions also, are spatially distributed rather than being localised neurally. Damage to a group of neurons or their synapses does not usually result in the sudden loss from memory of particular entries.)

Of course, the central question to be answered is how to choose these connections properly. That is, given a set of *nominated* patterns $\underline{S}^1$, $\underline{S}^2$, ... which one wishes to be stored in this neural memory, one must find an appropriate set of interactions T_{ij} such that each $\underline{S}^r$ is indeed a stable firing pattern. In other words, the problem to be solved is one of assigning the T_{ij} such that each spin configuration $\underline{S}^r$ is a local minimum of $E[\underline{S}]$ in (2.11). This is an inverse problem to the type usually solved in Statistical Physics where the form of the bond strengths is usually known and one then wishes to identify the ground and metastable states (minima) of the energy.

2.2 The Hopfield Model

One of the best-known models which falls into the category described above, and one which has received a great deal of attention since its appearance (Hopfield 1982) is the Hopfield model. Although most of the ideas incorporated in the model were not really new (some of them can be traced back to McCulloch and Pitts (1943) and Hebb (1949)), Hopfield did bring together many important ideas into the one model; not the least of which was the demonstration of the existence of an energy function for symmetric synapses. This has proved to be a powerful tool in the ensuing analysis of the model.

In this model the Hebbian prescription (based on (Hebb 1949)) for the synapses is utilised:

$$T_{ij} = \frac{1}{N}\sum_{r=1}^{p} S_i^r S_j^r \quad (i \neq j) \qquad\qquad (T_{ii} = 0), \tag{2.13}$$

where $\underline{S}^1, \underline{S}^2, \ldots, \underline{S}^p$ are the patterns to be stored. This obviously satisfies the symmetry requirements on T_{ij} and has the advantage of being a local 'learning' rule in that the change to a synapse due to the introduction of a new pattern $\underline{S}$

$$T_{ij} \longrightarrow T_{ij} + \frac{1}{N} S_i S_j$$

only depends on the activity of the two nodes (i and j) that it connects. The N^{-1} factor appears in order to keep the size of the connections finite even for the case where an extensive number ($O(N)$) of patterns are being considered. The individual neuron thresholds U_i are all taken to be zero. Henceforth this will be assumed unless otherwise stated.

The nominated states $\underline{S}^r$ in this model are chosen to be random, uncorrelated bit patterns (i.e., each S_i^r is an independent random variable assuming the value 1 or -1 with equal probability). Such a choice ensures the existence of both positive and negative connections in (2.11) and endows the model with the key ingredient of *frustration*. This is a phenomenon which occurs when all the bonds

T_{ij} cannot simultaneously be 'satisfied' (make their lowest possible contribution to the energy): a negative 'antiferromagnetic' (AFM) bond implies that the lowest contribution to the energy (2.11) is when the spin variables that it connects (S_i and S_j) have opposite signs, whereas a positive 'ferromagnetic' (FM) T_{ij} encourages both spins to be the same. When both types exist, it is easy to have, e.g., triples of bonds which exhibit frustration. A simple example would be $T_{12}, T_{23} > 0$ and $T_{31} < 0$, in which case it is impossible to choose a configuration of the three spins $\{ S_1, S_2, S_3 \}$ in which all three bonds are satisfied. This phenomenon can result in the existence of a huge number of configurations which are local minima of the energy.

If just one pattern $\underline{S}^1$ is stored ($p = 1$ in (2.13)), then the model is really just a fully-connected uniaxial Ising ferromagnet when the randomness is gauged away (as in the Mattis (1976) model): redefining the local definition of 'up' and 'down' by using new spin variables $S'_i \equiv S^1_i S_i$, the energy becomes

$$E[\underline{S}'] = -\frac{1}{2}\sum_{i \neq j} S'_i S'_j. \tag{2.14}$$

This only has the two minima: $\underline{S}' = \{ 1,1, \ldots ,1\}$ and $\{ -1,-1,\ldots ,-1\}$, i.e., all spins up or all spins down (hence 'ferromagnetic'). These states of course correspond respectively to complete alignment with the nominated pattern $\underline{S}^1$ and total misalignment with it. At the other extreme where the number of stored patterns becomes very large ($p \rightarrow \infty$), each T_{ij} tends to a random Gaussian variable of zero mean (as a consequence of the central limit theorem). This is very similar to the long-range Sherrington-Kirkpatrick(SK) (1975,1978) spin glass model in which the bond strengths are randomly positive or negative and typically drawn from a Gaussian distribution. (Of course, in the neural network model there are correlations between each T_{ij} as they are composed from a common set of random bit patterns.) So, in a sense, the case of a finite number of stored patterns is an interpolation between a ferromagnetic ($p = 1$) model and a spin glass ($p \rightarrow \infty$) model.

It is known from the analysis of such models as the SK one that spin glasses exhibit rich and complex behaviour due to the crucial ingredients of disorder and frustration. As a result of the latter characteristic it turns out that in the SK model the number of metastable states is exponential in the size of the system (Moore 1984). And due to the prevalent disorder, the system can exist in a *spin glass phase*: a type of ordering which is not found in pure systems. In such a phase the magnetic spins are frozen into thermal equilibrium orientations but lacking any long-range order: each individual spin on average points mostly either up or down, but the correlation between spins dies off rapidly to zero for spins that are separated by distances of the order of the size of the system. Physical examples of such spin glass systems are typically metallic alloy host materials which contain a few per cent of magnetic impurities, e.g., *Cu*Mn and *Au*Fe. The magnetic impurities are located at random sites on the host lattice and, due to the nature of the RKKY (Ruderman-Kittel-Kasuya-Yosida) interaction between these these spins (it is long-ranged and alternates between FM and AFM with separation), the system can be modelled by a lattice which contains a spin at every site but in which the bond strengths are chosen at random.

2.2.1 External Noise

The Hopfield model can be generalised to allow the introduction of external noise in the system by modifying the deterministic dynamical law (2.8) with a stochastic element. Instead of each spin definitely aligning with the current state of its local field there is now a chance that it will adopt the opposite alignment. This stochastic nature could be included to account for any noise that is present in a real neural system due to e.g., thermal processes. The dynamical process is

now

$$
\begin{aligned}
S_i(t)h_i(t) \leq 0 \;\; &\Rightarrow \;\; S_i(t+1) = \;\; sgn\{h_i(t)\} \\
S_i(t)h_i(t) > 0 \;\; &\Rightarrow \;\; S_i(t+1) = \begin{cases} -sgn(h_i(t)) & \text{with prob.} \;\; e^{-\beta S_i(t)h_i(t)} \\ sgn(h_i(t)) & \text{with prob.} \;\; 1 - e^{-\beta S_i(t)h_i(t)} \end{cases}
\end{aligned}
\qquad (2.15)
$$

where β parameterises the external noise: $\beta \rightarrow \infty$ (no noise) recovers the deterministic dynamics $S_i(t+1) = sgn(h_i(t))$; $\beta \rightarrow 0$ (maximum noise) is the case where every spin continually flips with respect to its previous state, regardless of the local field h_i.

This is exactly analogous to a Monte Carlo simulation using the Metropolis algorithm (Metropolis *et al.*1953) at a temperature $T = \beta^{-1}$. In that case spin flips which reduce energy are definitely accepted and those that would produce an energy increase of δE are accepted with probability $\exp\{-\delta E/T\}$. This is also the case here since the chance of a spin flipping against its local field is $\exp\{-\beta S_i h_i\}$ and the resultant increase in energy would be $\delta E = -S_i h_i$. Of course, the difference between these two algorithms is that in the Metropolis case, the spin flips are proposed with respect to the current alignment of the spin, whereas in (2.15) the spin flips are with respect to the local field h_i.

An alternative way of incorporating parameterised external noise is in the Little model (Little 1974, Little and Shaw 1978). There the probability that a spin adopts the state $S_i' \equiv S_i(t+1)$ at time $t+1$ is

$$
P(S_i') = \frac{e^{-\beta S_i' h_i(t)}}{e^{-\beta S_i' h_i(t)} + e^{+\beta S_i' h_i(t)}}, \qquad (2.16)
$$

which also reduces to (2.8) when $\beta \rightarrow \infty$.

The motivation behind such Monte Carlo simulations of physical systems is that they simulate the thermal fluctuations in a system which is in contact (can exchange energy with) its surroundings. It can be shown that after a sufficiently large number of updates (2.15) the system will be in equilibrium with

its surroundings in that the probability of the system being in a configuration $\underline{S}$ is given by the Boltzmann-Gibbs weight

$$P[\underline{S}] = \frac{1}{Z} e^{-\beta E[\underline{S}]}. \tag{2.17}$$

This is the *canonical distribution*, where the *partition function* Z is

$$Z = \operatorname*{Tr}_{\underline{S}} e^{-\beta E[\underline{S}]}. \tag{2.18}$$

The 'trace' Tr represents a summation over all (2^N) possible configurations $\underline{S}$.

The beauty of algorithms such as the Metropolis one is that they can generate configurations of the system correctly distributed according to the canonical distribution (2.17) without needing to expilicitly evaluate every term in the partition function (2.18); a task which would involve a prohibitive amount of computer time for even a modest system size.

It is well known from Statistical Mechanics that all of the thermodynamics (physical observables) can be obtained through the *free energy*

$$F = -T \ln Z. \tag{2.19}$$

For the general stochastic model ($T > 0$) it is the minima of the free energy F which are the most probable equilibrium states. So now the energy function $E[\underline{S}]$ plays an even more important role than before. In the no noise case it was no more than a mathematical function whose minima were the stable, and thus stored, states of the system. But it now also allows the controlled introduction of external noise and provides the means by which powerful techniques of thermodynamics and statistical mechanics can be exploited.

The disorder in the system (due to the randomness of the nominal bit patterns) must be averaged over by *quenched averaging*. This is done by choosing a given realisation of the nominal patterns and then using the corresponding free energy (2.19) to calculate any desired physical observables (such as how much the equilibrium states resemble the nominal states). Then the average over the choice

of nominal patterns must be carried out. This would involve the average of the logarithm of Z in (2.19); a requirement which can circumvented by the use of the 'replica trick' (Edwards and Anderson 1975, Kirkpatrick and Sherrington 1978, Parisi 1980)

$$lnZ = \lim_{n \to 0} \frac{Z^n - 1}{n}. \tag{2.20}$$

The average now to be carried out is over Z^n which just represents the partition function of a collection of n independent copies (replicas) of the system: a much easier task.

Using these theoretical tools in this pseudo-thermodynamical framework, Amit, Gutfreund and Sompolinsky (1985[a,b], 1987[a,b]) were able to extensively analyse the Hopfield model, finding how the stable states of the network were related to the nominal patterns it was attempting to store. Their analysis was presented for all values of temperature (thermal noise), but it will only be the zero temperature (determinsitic) limit which shall be required here. Their results are valid for the thermodynamic limit, whereby $N \rightarrow \infty$. In terms of the storage ratio $\alpha \equiv p/N$ (the ratio of the number of nominal states to the number of neurons in the network), they found that the network behaved as follows as α is increased from zero.

- For $p \geq 3$ (infinitesimal α) 'spurious' minima are created in the energy surface. These are 'mixture states' which correspond to linear combinations of nominal states. All of the nominal states are successfully stored, albeit with a few errors, as each one lies very close to a minimum of the energy. Due to their high correlation with a nominal state, these nearby stable states are designated as *ferromagnetic* (FM). Although the spurious states are only *metastable* in the sense that their energies are an extensive amount (O(N)) higher than the FM states, they are just as stable since they too lie in basins of attraction surrounded by energy barriers of order N.

- For finite α ($p = \mathrm{O}(N)$) another class of spurious states appear: the *spin glass* (SG) states. Unlike the mixture states, these are uncorrelated with any of the embedded nominal states.

Now, it will be convenient to introduce a natural order parameter of the system which measures the correlation of the state $\underline{S}$ of the system with a nominal pattern $\underline{S}^r$. This is the *overlap* between $\underline{S}$ and $\underline{S}^r$ and is defined as

$$m^r \equiv \frac{1}{N}\sum_{i=1}^{N} S_i S_i^r. \tag{2.21}$$

When $\underline{S}$ and $\underline{S}^r$ are identical, $m^r = 1$; when $\underline{S}$ is the complement of $\underline{S}^r$ then $m^r = -1$; if $\underline{S}$ is randomly chosen with respect to $\underline{S}^r$, then $m^r = \mathrm{O}(\frac{1}{\sqrt{N}})$. The latter will be the case if $\underline{S}$ and $\underline{S}^r$ are uncorrelated. Hence a spin glass state is a stable state of the system with $\mathrm{O}(\frac{1}{\sqrt{N}})$ overlap with all of the embedded patterns.

At this stage on the scale of increasing α the spurious states are still of higher energy than the retrieval (FM) states, which still correspond to the nominal patterns with a few errors. Nonetheless, the number of SG states is exponential in N and their existence signals the danger of 'confused recall': if the network is initialised in a state which is too highly distorted by noise from a nominal pattern, then this initial configuration could easily be one lying in the region of attraction of one of the plethora of spurious states.

- As α is increased beyond a first critical value, α_1^c, a phase transition occurs in the sense that the energy of the spurious states becomes lower than that of the FM states. Thus now the roles are reversed: the FM states have been demoted to metastability, while the SG states assume the mantle of the ground states of the system. The network still of course performs as an associative memory for the nominal states.

- Increasing α further, past a second critical value, α_2^c, the system undergoes another phase transition which this time has catastrophic consequences for

the network's performance in storing the desired patterns. The FM states associated with the nominal patterns are wiped out and the only remaining stable states are the SG states.

Amit *et al.* calculated the two critical storage ratios as

$$\alpha_1^c \simeq 0.05, \qquad \alpha_2^c \simeq 0.14 \tag{2.22}$$

within a replica-symmetric mean-field approximation.

The latter value is consistent with the seminal work of Hopfield who reported numerical simulations on systems of $N = 30$ and $N = 100$ neurons as indicating loss of memory capacity around $0.13 - 0.15$. More extensive numerical work (Amit 1987) supports these theoretical predictions.

The strong analogy of these models with fully-connected spin glass models has generated a great upsurge of interest amongst the physics community (e.g., Mézard *et al.* 1986, Gardner 1986, Bruce *et al.* 1987, van Hemmen 1986).

Chapter 3

Content-addressability of the Hopfield Model

3.1 Introduction

A memory is deemed *content-addressable* if the facts which it stores can each be evoked by a cue (i.e., input) which sufficiently resembles the desired fact to be retrieved. In other words, an entry in the memory is accessed by supplying an input containing a large enough part of the entry: each piece of stored information is addressed on the basis of its content. This is in marked contrast to the memory of a conventional computer which stores data in specific locations, each designated an address—usually a sequence of bits. It must be supplied with the exact bit-address or else, even if only one bit is wrong, a different memory location will be accessed and, in principle, a completely unrelated piece of data will be evoked. A content-addressable memory, on the other hand, can cope with distorted cues. The more distortion it can cope with (i.e., the less the cue need resemble the desired data), the greater is the content-addressability of the memory.

The analysis of the Hopfield model described in the preceding chapter was in terms of its ability to store random patterns. This involves determining whether

the nominal states, or states sufficiently resembling them, correspond to fixed points of the neural update dynamics (2.8). Thus, this is essentially a statics problem and the question of how large the regions of attraction are around the fixed points is not directly addressed. This is a central question which must be answered if one is to discover how well the model performs as a content-addressable memory. After all, the 'naive' choice of the unit matrix as the matrix of connection strengths ($T_{ij} = \delta_{ij}$) would ensure perfect storage of all 2^N possible patterns but, of course, with no content-addressability whatsoever. So it is not enough just to be concerned with how many fixed points can be successfully created in the dynamics from a given choice of the T_{ij}.

Given that the nominal patterns are successfully stored in the memory, how distorted can an input state be with respect to a nominal state such that the desired pattern will still be evoked? The answer is that the network can cope with no more distortion than would result in the input state lying just outside the region of attraction of the stored nominal pattern. Hence, to ascertain the content-addressability of the memory, one must determine the size of the basins of attraction of the stored patterns.

To tackle this task analytically one would have to calculate the probability that the network would stabilise on (or sufficiently close to) a nominal state if it were initialised in a state which was obtained by distorting that nominal pattern with a given amount of noise. Alternatively, one could attempt to calculate the mean overlap of the final (stabilised) state with the nominal pattern, given that the initial state had a certain overlap with it. If all the neurons are updated in parallel then calculation of the mean overlap after one updating sweep through the network from a given initial state is relatively straightforward. However, stabilisation generally takes more than one sweep and to then proceed and calculate the state of the network after a second sweep would be much more complicated as there are correlations to be taken into account with the previous sweep. Namely, the local field at any spin during the second sweep depends

on what happened to all of the spins during the first sweep. (Gardner *et al.* (1987[c])show how these complications necessitate the introduction of more and more order parameters.) Similarly, there are such troublesome correlations if the sweeping is performed sequentially. Indeed, the situation is exacerbated as these correlations arise immediately during the first sweep.

The daunting prospect of an analytical approach to the investigation of the Hopfield model's content-addressability is avoided here by attacking the problem by numerical simulation. As remarked earlier, this model is of a type particularly amenable to computer experiment since the state of each variable requires representation by only one boolean variable. It is the behaviour of the model in the thermodynamic limit ($N \rightarrow \infty$) which is desired, and, of course, any computer simulation must deal with a finite system. However, as is usually the case when simulating thermodynamical systems, the models simulated here were of increasing size N. The trend was then analysed to extrapolate the expected behaviour in the thermodynamic limit.

3.2 DAP Implementation

A limiting factor on the size of network able to be simulated in a reasonable time is the fact that this model is fully connected: every neuron must contact all of the others before it can update its state. This limitation was ameliorated by exploiting the parallelism of the ICL DAP (International Computers Limited Distributed Array Processor): despite the fact that all the simulations involved sequential updating of the neurons, there was *algorithmic parallelism* in calculating the matrix (T_{ij}) - vector (S_j) multiplies required in (2.8). The simulations also exploited *task parallelism* in that a number of simulations were being executed concurrently.

3.2.1 The DAP

The Distributed Array Processor (DAP) (Reddaway 1979) comprises a 64×64 square array of 4096 processing elements (PEs). Each PE is a bit-serial processor: it performs operations on one bit of data at a time. Although this means that arithmetical operations must be performed in software, the bit-serial nature makes the DAP a very powerful tool for problems involving logical variables, included amongst which are, of course, Ising spin neural network models. It also allows flexible word length (INTEGER or REAL *1,2,...,8), a feature not common amongst other computers. There are 4 Kbits (4096 bits) of memory associated with each PE, and for each bit in PE memory, the square lattice of 4096 PE's can be visualised as occupying a logical (bit) plane of the DAP. So the DAP consists of 4096 of these logical planes and therefore comprises a total memory of 2 Megabytes.

The DAP is an example of a SIMD machine - Single Instruction stream, Multiple Data stream machine. This is because all of the processors can perform the same operation on their own piece of data at the same time. Hence the DAP is performing this same operation (Single Instruction) simultaneously on different data (Multiple Data).

The A (activity) plane of the DAP gives it the ability to mask out certain PEs while a particular operation is being executed: at any site in this plane at which there is a .FALSE. (zero) bit, that PE will remain idle during the present operation cycle.

Communication on the DAP is achieved via the four nearest neighbour connections (conventionally called North, South, East and West) between the PEs. The array can be specified in the software as having either cyclic or fixed (planar) boundary conditions.

The programming language of the DAP is DAP-FORTRAN, a parallelised extension of FORTRAN-IV which incorporates matrix and vector variables. A matrix is declared as $M(,)$, which consists of the 64×64 matrix elements $M_{ij}, i, j = 1, \ldots 64$, where M_{ij} resides on the processor at the ith row and jth column of the DAP's array.

If A, B and C are all matrices, then the instruction $A = B * C$ will execute $A_{ij} = B_{ij} * C_{ij}$ simultaneously on all 4096 PEs. This holds equally well for $*$ replaced by $+, -$ or $/$.

Masking is achieved by using a logical matrix $L(,)$: $A(L) = B * C$ will only assign $A_{ij} = B_{ij} * C_{ij}$ on those processors ij at which L_{ij} is .TRUE., while A_{ij} will remain unchanged if L_{ij} is .FALSE.

The construct $M(,,n)$ represents an array of n matrices.

3.3 Numerical Results

To determine the typical sizes of the basins of attraction (and hence the content-addressability) of the nominal patterns $\underline{S}^r$ in the Hopfield model, states $\underline{S}^{(r,s)}(s = 1, 2 \ \ldots \ n_{m_0})$ were constructed, each having a given initial overlap m_0 with the pattern $\underline{S}^r$:

$$m(\underline{S}^r, \underline{S}^{r,s}) = m_0 = \frac{1}{N} \sum_{i=1}^{N} S_i^r S_i^{(r,s)} \tag{3.1}$$

The $(1 - m_0)/2$ sites at which $\underline{S}^r$ and $\underline{S}^{(r,s)}$ differed were chosen randomly for each s. The network was initialised in the state $\underline{S}^{(r,s)}$ and then iterated to stability under the single spin-flip dynamics (2.8). $\underline{S}^r$ was deemed to have been successfully recalled if the resultant stable state differed from it in no more than $N/16$ spin sites. This margin of recall error allows for the possibility of ferromagnetic states highly correlated with the nominal states being stored

instead of the nominal states themselves. The analytical work of Amit *et al.* suggests that these FM states differ in no more than $0.015N$ sites with respect to a nominal state, provided $\alpha < 0.14$. The more lenient margin of $N/16 \equiv 0.0675N$ was thought to be sufficient to account for any finite size effects—the analytical predictions are, after all, valid for the thermodynamical limit of $N \rightarrow \infty$.

Results shown in figure 3.1 for the mean fraction $f(m_0)$ of states which were recalled with less than $N/16$ errors from various initial overlaps m_0 were obtained by averaging over more than 1000 initial states for each point in the graph. (That is, at each value of storage parameter α the number of states iterated from, $N\alpha\ n_{m_0}$, was more than 1000.) In order to investigate the behaviour of the system in the thermodynamic limit, simulations were carried out for three different system sizes: $N = 512, 1024$ and 2048. Statistical error bars, typically of the order of $\pm 3\%$ are suppressed in the interests of clarity.

It is evident from the graph that as the system size increases, the rate at which $f(m_0)$ increases from near 0 to near 1 becomes more pronounced for values of $\alpha < 0.13$. The trend suggested is that this change may approach a discontinuous jump as $N \rightarrow \infty$, displaying a critical minimum overlap $m_0(\alpha)$ which an initial pattern must have with the stored pattern in order to be in its basin of attraction (and thus ensure its successful retrieval).

The nature of this cross-over appears to be sigmoidal with increasing steepness as N is increased. The following form for $f(m_0)$ was fitted:

$$f(m_0) = \frac{1}{1 + C\exp\{aN(m_0 - m_c)\}}, \tag{3.2}$$

where C and a are constants independent of N (but possibly dependent on α).

This is equivalent to expressing the ratio of probability of recall to that of non-recall in the following scaling form

$$\frac{f(m_0)}{1 - f(m_0)} = C\exp\{aN(m_0 - m_c)\}. \tag{3.3}$$

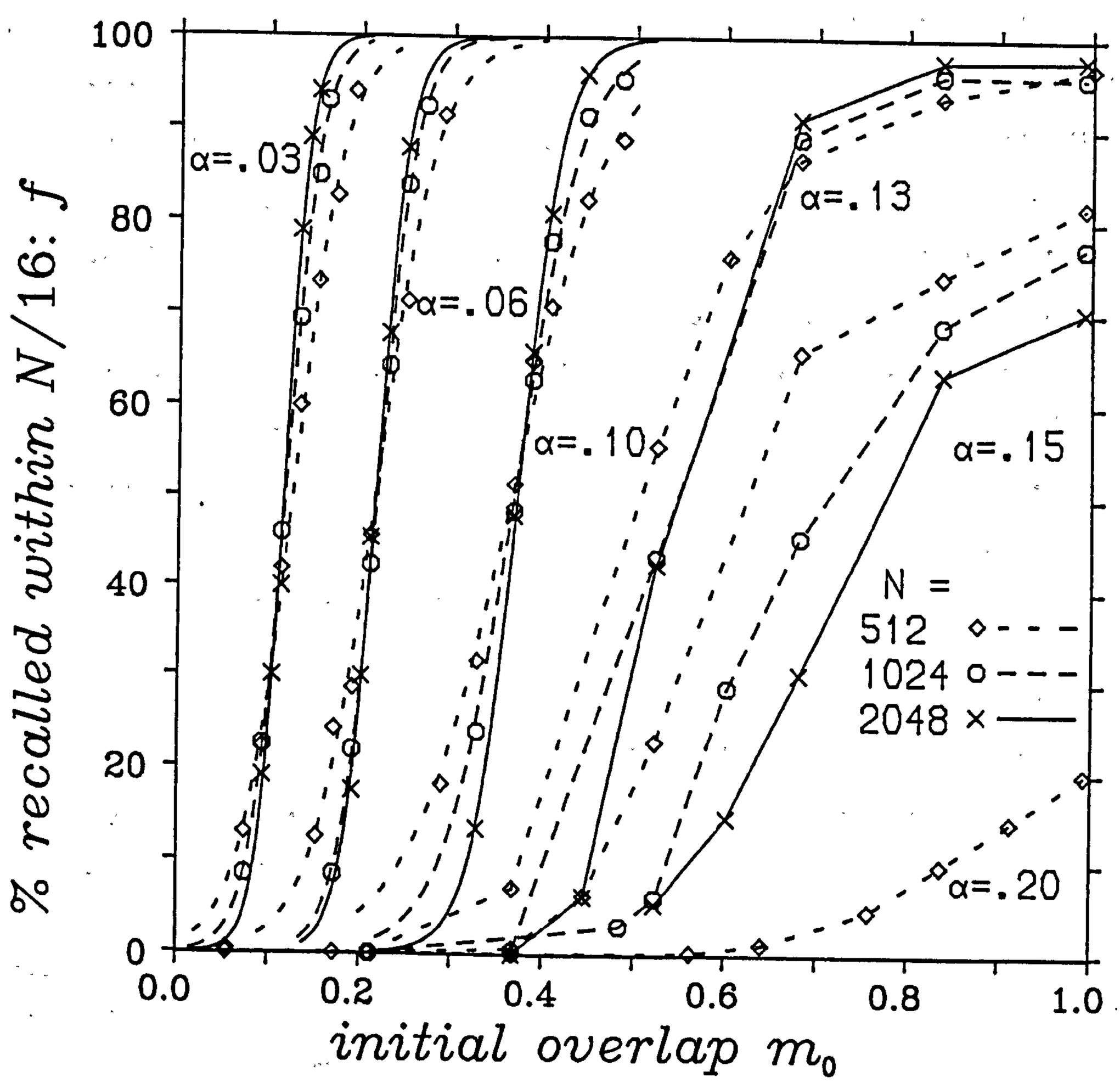

Figure 3.1: Mean recall fraction of the Hopfield model

To test this hypothesis, $\ln\{f/(1-f)\}$ was plotted against m_0 (figure 3.2). If (3.2) holds, then

$$\ln\left\{\frac{f}{1-f}\right\} = aNm_0 + \ln C - aNm_c, \tag{3.4}$$

so that this should reveal a linear relationship. This was indeed the case, as can be seen from figure 3.2. The best-fit straight lines were obtained by linear Gaussian regression: if each data point of the graph is denoted by $(x_i, y_i \pm e_i)$, where e_i is the error, then the procedure is as follows. For each data point generate the coordinate (x_i, y_i^*), where y_i^* is a random Gaussian variable of mean y_i and standard deviation e_i (this simulates the statistical error e_i in the measurement y_i). For this particular realisation of all the data points, find the least-squares straight line $y = gx + b$ through them. Repeat this procedure a large number of times, each time finding the best-fit gradients g and y-axis intercepts b. Then the mean best-fit line is $y = (\bar{g} \pm \delta g)x + (\bar{b} \pm \delta b)$, where $\bar{g}$ and $\bar{b}$ are the gradient and intercept averaged over all these realisations and δg and δb are their associated standard errors.

In (3.4) $y \equiv \ln\{f/(1-f)\}$, $x \equiv m_0$, $g \equiv aN$ and $b \equiv \ln C - aNm_c$, so that the best-fit scaling form (3.3) has $C = \exp\{b + aNm_c\}$. To obtain C the critical overlap m_c is required. This can be obtained in the following manner by extrapolating to $N \to \infty$.

Inverting (3.4), one has

$$m_0(f) = \frac{1}{aN}\left\{\ln\frac{f}{1-f} - \ln C\right\} + m_c, \tag{3.5}$$

so that as $N \to \infty$, or, equivalently, $N^{-1} \to 0$,

$$m_c = \lim_{N^{-1}\to 0}(m_0(f)), \tag{3.6}$$

Thus, for a given recall fraction f, a graph of m_0 (the initial overlap required to produce that recall fraction f) versus N^{-1} should produce a straight line intercepting the N^{-1}-axis at $m_0 = m_c$. Figure 3.3 shows this relationship. The

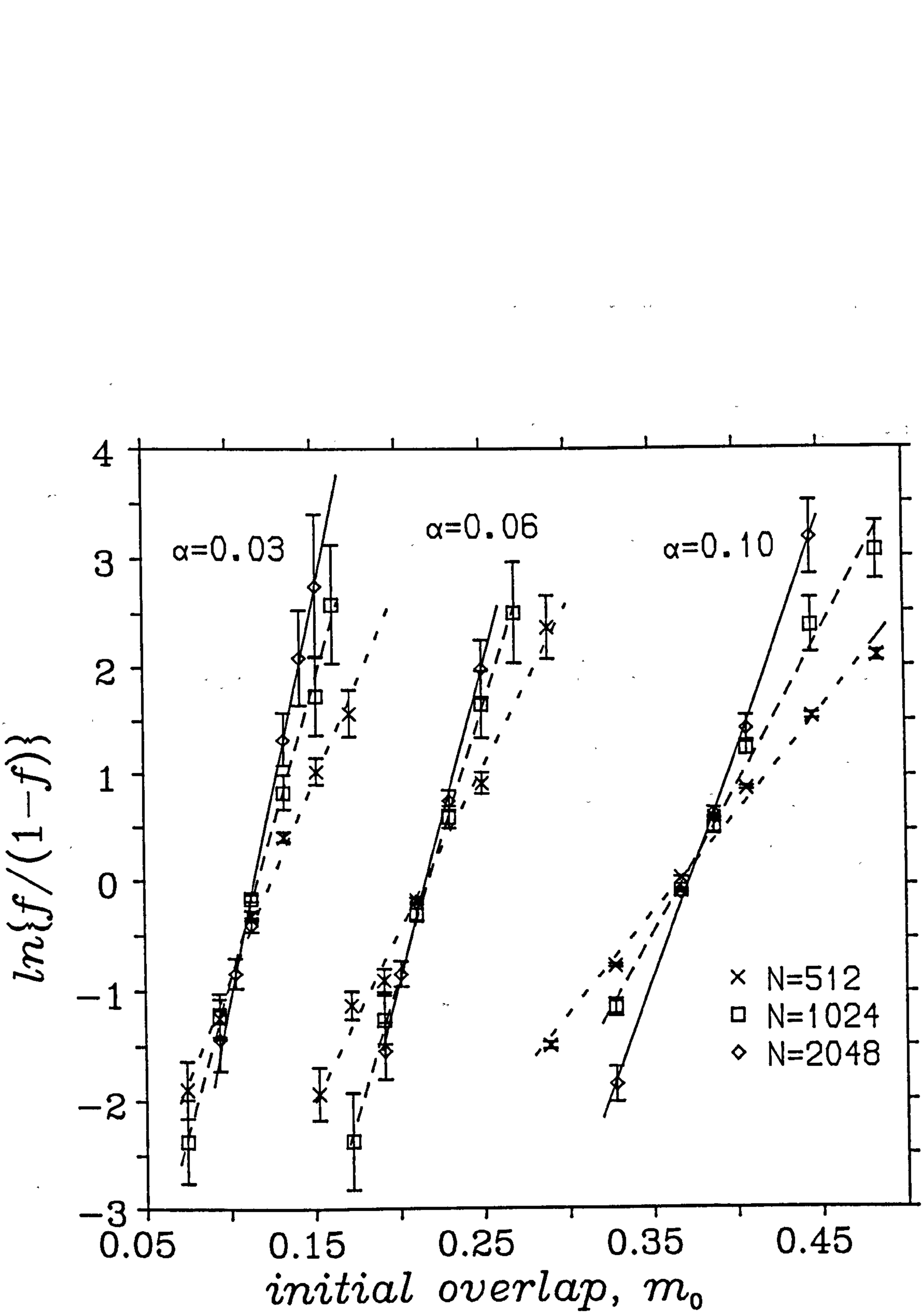

Figure 3.2: Finite size scaling of the basins of attraction in the Hopfield model

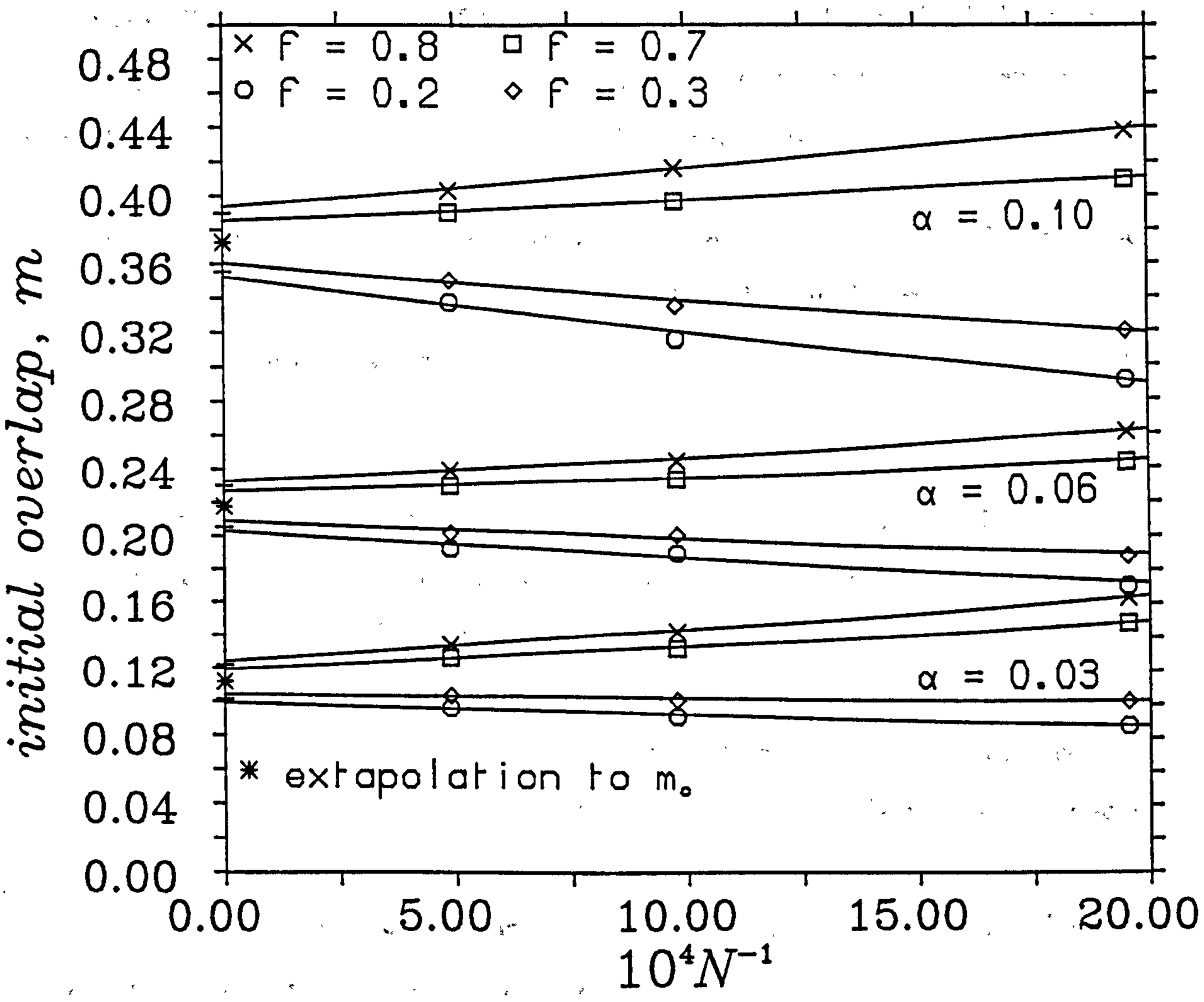

Figure 3.3: Extrapolation to the critical minimum overlap

initial overlap m_0 which produced a given recall fraction f was obtained from (3.4) by linear interpolation. The extrapolation to $N^{-1} \rightarrow \infty$ revealed the following estimates for the critical overlaps $m_c(\alpha)$:

$$m_c(0.03) = 0.111(10); \qquad m_c(0.06) = 0.218(13); \qquad m_c(0.10) = 0.372(17).$$

This then allowed evaluation of the constant C in (3.2) for each of these three values of α. The resultant scaling forms are shown in figure 3.1: they are the best-fit curves which appear for $\alpha = 0.03, 0.06$ and 0.10.

In contrast to the above behaviour, at $\alpha = 0.15$ in figure 3.1, $f(m_0)$ decreases as N increases. This is consistent with analytical studies of this model: this value of α lies above the critical value ($\alpha_2^c \simeq 0.14$) for which no FM states are stable, and so $f(m_0) = 0$ for $0 < m_0 \leq 1$ as $N \rightarrow \infty$.

Another measure of content-addressability is described in figure 3.4: the mean final overlap m_f with a nominal pattern that a state acquires from an initial state having overlap m_0. In a similar fashion to the mean recall fraction ($f(m_0)$), $m_f(m_0)$ is seen to increase more sharply with increasing N for $\alpha \leq 0.13$, while it deteriorates for $\alpha = 0.15$. Although, as with $f(m_0)$, m_f stays close to 1 for large enough m_0 at $\alpha \leq 0.13$, it does not exhibit the same all-or-none tendencies ($m_f \rightarrow 0$ or 1) for increasing N. This is because states which are not recalled ($f = 0$) nevertheless retain a non-zero final overlap with the nominal states ($m_f \neq 0$). It is only as $m_0 \rightarrow 0$ that $m_f \rightarrow 0$.

3.4 Discussion

Previous analytical work (Amit *et al.* 1985[a,b], 1987[a,b], Bruce *et al.*1987) and numerical studies (Amit 1987, Bruce *et al.*1987) had already demonstrated that the Hopfield model exhibits critical behaviour in terms of its success in storing nominated patterns in its memory. The numerical work presented in this chapter

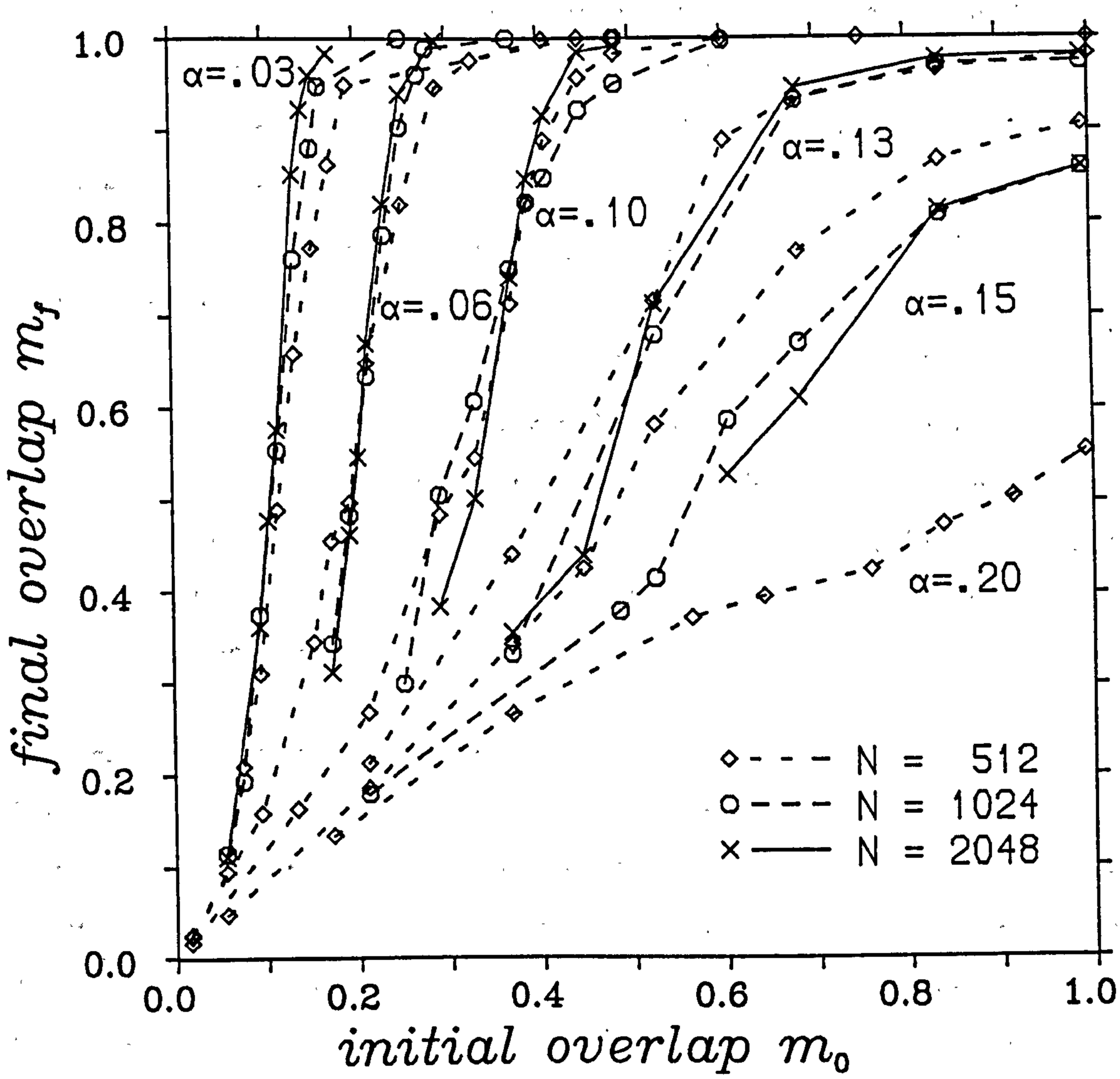

Figure 3.4: Mean final overlap of states iterated to stability in the Hopfield Model

has provided strong evidence to suggest that the associated basins of attraction of these nominal states also display behaviour consistent with a first order phase transition, at least for values of α in the retrieval (FM) phase ($\alpha < \alpha_2^c$). In such a phase transition there is typically an order parameter of the system which undergoes a discontinuous jump as a control parameter is continuously altered past a critical value. Usually the control parameter is the temperature of a physical system and the order parameter could be, e.g., the net magnetisation of a ferromagnet at zero external field, or the difference in the density of gas and liquid of a substance undergoing a liquid-gas phase transition. In the case here, the order parameter can be considered to be the recall probability of a nominal pattern from a distorted state, or, equivalently, the degree to which the final stabilised state resembles the nominal state in question: if it resembles it sufficiently then the system is "ordered" (with respect to the nominal pattern); if it doesn't resemble it, the system is in a "disordered" state. The control parameter here is the initial overlap of the state of the system with the desired nominal pattern. This is analogous to the net magnetisation in a uniaxial Ising ferromagnet, but where each spin is considered "up" ("down") according to whether it is (is not) aligned with the corresponding spin in the nominal configuration.

This critical behaviour of the basins of attraction would imply that they are becoming completely isotropic in the thermodynamic limit: whether or not a nominal pattern is retrieved is independent of which particular spins have been flipped in the noisy pattern and only depends on their number. Viewing each of the N spin variables as defining an axis in N-dimensional space, as $N \rightarrow \infty$ these basins of attraction approach the shape of an hypersphere: they extend for the same distance in every direction of phase space.

Despite the fact that even for $\alpha < \alpha_2^c$ perfect recall is not guaranteed in this model, the results obtained here have shown that it can operate as a good content-addressable memory—certainly for $\alpha \leq 0.10$. (Previous numerical work (Amit 1987) suggests good retrieval up to $\alpha = 0.13$.) For example, the result

$m_c(\alpha = 0.06) = 0.218$ indicates that in a large network each of the $0.06N$ patterns could be successfully recalled from up to 78.2% noise (39.1% spins flipped).

Not surprisingly, the critical minimum overlap $m_c(\alpha)$ is a decreasing function of α. It is not so much the growing number of nominal patterns, but rather the exponentially growing number of spuriously created states which reduce the region of attraction around the nominal patterns.

Further work would be required to ascertain the shape of m_c as a function of α. Estimates have been obtained here for three points of this graph ($\alpha = 0.03, 0.06$ and 0.10). Another point is $m_c \to 0$ as $\alpha \to 0$ since, for the storage of one pattern (hence $\alpha = 1/N \to 0$ as $N \to \infty$), perfect recall will be achieved from any $m_0 > 0$, as the only two stable states are the nominal pattern and its complement. Presumably $m_c \to 1$ as $\alpha \to \alpha_2^c$ since $\alpha > \alpha_2^c$ implies the non-existence of any FM states and, of course, of any region of attraction.

Chapter 4

Learning Algorithms

4.1 Introduction

'Learning' in neural networks is the process whereby the synaptic connections between the neurons are tuned or adjusted in response to the current neural activity in such a way that the activity patterns behave in the desired fashion. This can be viewed as a dynamical process that is reciprocal to that of recall. In the former process the activity of the neurons determine the dynamics of the synapses, while in the latter the fixed synaptic connections govern the dyamical activity of the neurons.

In layered networks learning corresponds to altering the synaptic connections (consistent with the layered architecture) in order that a given set of input activity patterns will induce an appropriate set of output patterns. For instance, one particular scheme that involves altering the interlayer connections in an output-to-input direction is the Error Back Propagation algorithm (Rumelhart *et al.*1986, Parker 1985, Le Cun 1985). This has received (and still is receiving) much attention in the neural network research community.

The learning algorithms of this chapter are of the type applied to the content-

addressable boolean neural networks dealt with in the previous two chapters. Hence, the goal of these algorithms is to modify the synapses until all of the nominal firing patterns are dynamically stable (and thus stored) in the neural memory.

In contrast to the Hebbian ansatz used in the Hopfield model, these algorithms use an iterative approach to tackle the problem of finding an appropriate set of connections. For a fully connected network, even allowing for the continued imposition of symmetry ($T_{ij} = T_{ji}$) and non self-interaction ($T_{ii} = 0$), this is a search through a very high ($\frac{1}{2}\,N(N-1)$) dimensional space of quasi-continuous variables. Nevertheless, it will be shown that solutions can be found without the need for resorting to (computationally expensive) stochastic searches. These algorithms also have the desired property of being *local* in that any change to a connection is only dependent on the states of the two neurons which it connects.

4.2 Learning Perfect Storage

The Hebbian prescription for the connection strengths has been shown to be rather limited in its capacity to store random uncorrelated patterns: it fails to store more than $0.15N$ of them, and for a general choice of connections it should be possible to store up to $2N$ such patterns (Cover 1965, Venkatesh 1986[a,b]). Even for values of α below 0.15 this storage prescription cannot ensure perfect recall of the patterns.

There exists an iterative learning algorithm (Wallace 1986, Bruce *et al.*1986), which is essentially an extension of perceptron learning (Minsky and Papert 1969), that has been shown capable of perfectly storing up to at least N random patterns on a network on N neurons. The idea behind the algorithm is conceptually simple: at each learning iteration every nominal pattern is tested for

stability, and for those that are not yet totally stable, the corresponding term in the Hebbian storage rule (2.13) is reinforced.

The algorithm proceeds as follows. An error mask ϵ_i^r is constructed at each site i for each nominal pattern r:

$$\epsilon_i^r = \frac{1}{2}[1 - sgn S_i^r \sum_{j=1}^{N} T_{ij} S_j^r] \tag{4.1}$$

so that ϵ_i^r is assigned the value 0 (respectively 1) if the spin at site i is (respectively is not) aligned with its local field. The connections T_{ij} are then modified accordingly

$$T_{ij} \longrightarrow T_{ij} + \frac{1}{N} \sum_{r=1}^{p} (\epsilon_i^r + \epsilon_j^r) S_i^r S_j^r, \qquad T_{ii} = 0 \tag{4.2}$$

This form was chosen to retain the symmetry of the connections and thus preserve the existence of the Lyapunov (energy) function (2.11) so that any trajectory under single spin-flip dynamics will terminate at a stable configuration (which minimises E).

An asymmetric rule could equally well be used (i.e., omitting the ϵ_j^r in (4.2)), but of course E would no longer exist. Convergence theorems exist for both the symmetric (Wallace 1986) and asymmetric (Gardner *et al.*1987[a]) versions. That is, it may be shown that if solutions exist for the T_{ij} then the learning algorithm will converge to one.

4.3 Numerical Results

Although the above learning algorithm had already been shown capable of storing perfectly up to at least N patterns on a network of N nodes, nothing quantitative was known about the resultant content-addressability of the stored states. Thus it was decided to simulate a network of N neurons employing the above algorithm (4.1,2) to store perfectly $N\alpha$ nominal states. These simulations

were once again carried out on the ICL DAP. Since the connections now had to be stored in the computer's memory—they are continually modified during learning—the size of system capable of being simulated was less than that for a Hopfield model simulation. In this case the networks contained $N = 256$ and 512 neurons. Details of the DAP implementation are provided in Appendix B.

Starting from the Hebbian storage prescription (2.13), when or if perfect storage of all $N\alpha$ states had been achieved, their content-addressability was then tested in the same way as before (section 3.3).

Figure 4.1 (m_f versus m_0) shows the mean final overlap m_f after iteration from a state with overlap m_0. Although perfect storage is indeed achieved for $\alpha = 0.5$ and 1.0, the states have very poor content-addressability. In fact, for $\alpha = 1.0$, initial states having merely one spin misaligned out of the 512 resulted in over 65% of the states failing to be recalled—the non-recalled states ended up with a final overlap of around 0.6 only.

Thus it would seem that this learning algorithm instals no appreciable content-addressability at these higher values of α (0.5 and 1.0), but instead creates fixed points of the dynamics having negligible regions of attraction. Ensuring perfect storage of the patterns does not then guarantee good content-addressability.

Nevertheless, this algorithm does substantially improve upon the Hebbian prescription for $\alpha = 0.13$, 0.15 and 0.2, as is demonstrated in figure 4.2. This compares the fraction of states recalled with less than $N/16$ errors from various initial overlaps m_0 before and after implementation of the algorithm. Simulations of systems with larger N would be required to determine whether appreciable content-addressability would be installed as $N \to \infty$. However, one would suspect from the shape of the curves that this would be the case.

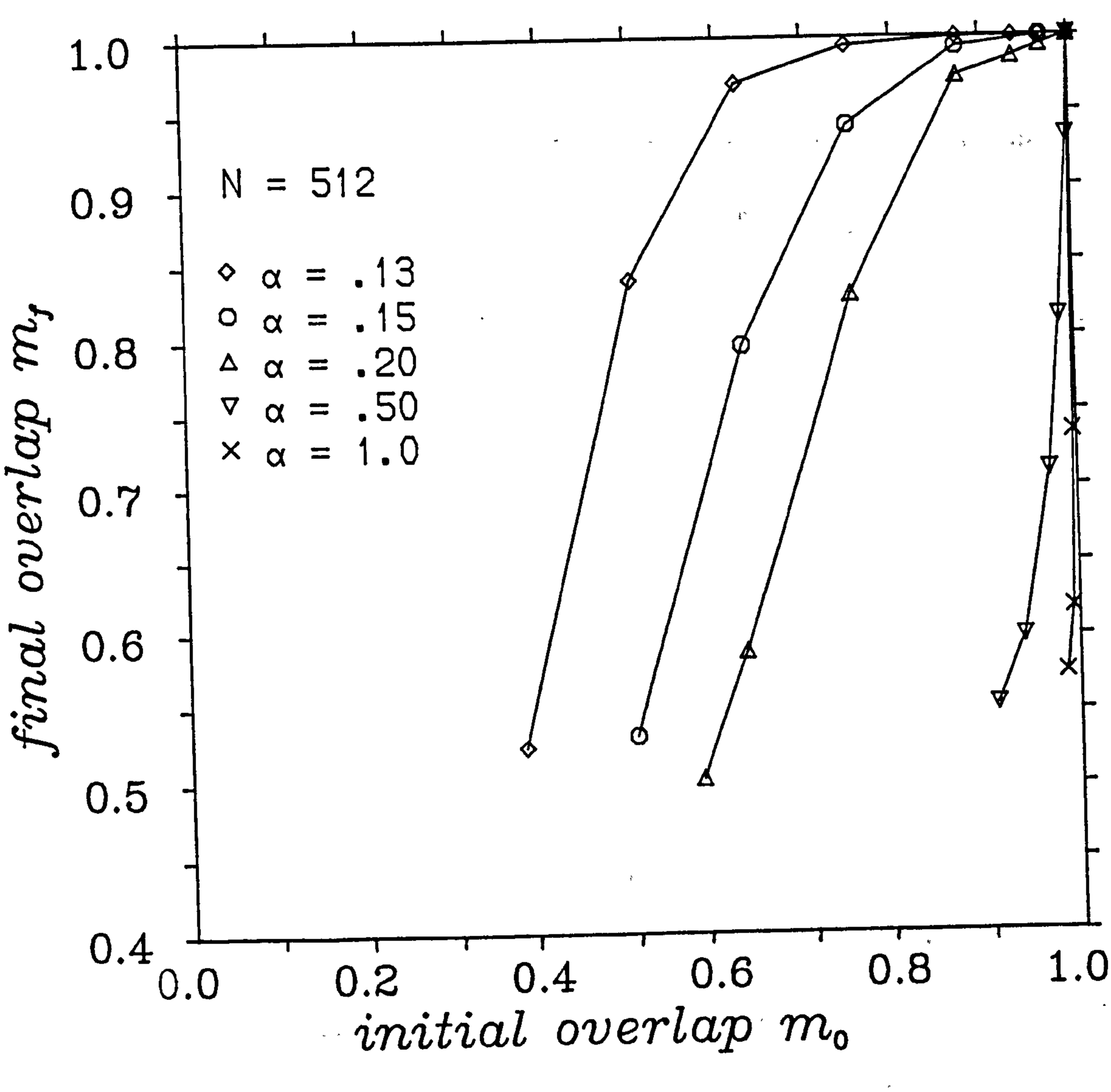

Figure 4.1: Mean final overlap of iterated states after learning

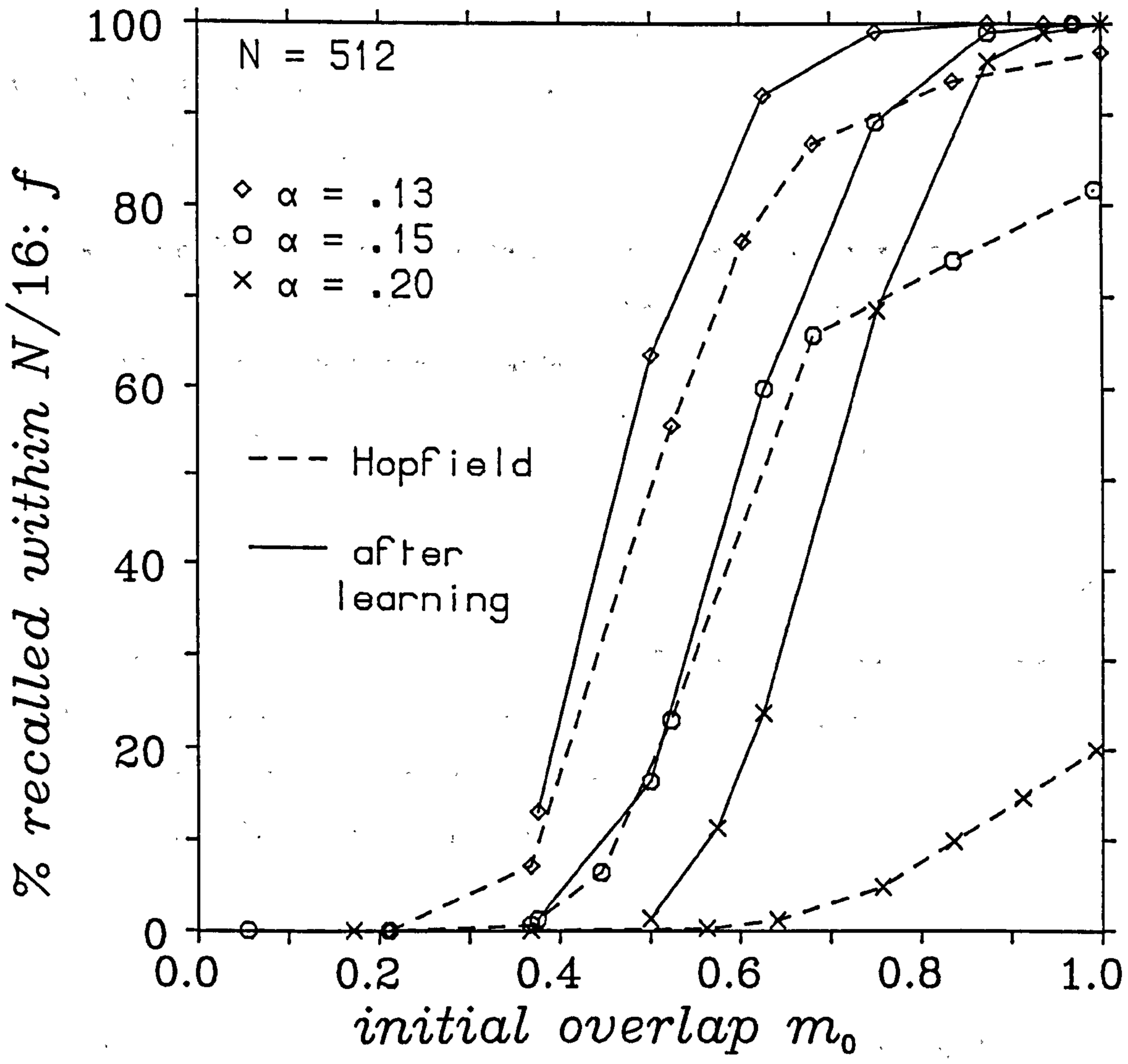

Figure 4.2: Comparison of content-addressability before and after learning

4.4 Learning Content-addressability

The work of the previous section has shown that merely ensuring that each nominal pattern is perfectly stored in the network does not necessarily guarantee that they will have large regions of attraction.

This may have been predicted from the fact that all the algorithm (4.1,2) demands of the T_{ij} is that in each nominal state all of the spins are aligned with their local field: a condition that can be satisfied even if the net alignment $h_i S_i^r$ is small, though positive.

Now, suppose the network is in a state $\underline{S}^{r*}$ which is a noisy version of $\underline{S}^r$, then if F denotes the set of spins which are misaligned with respect to $\underline{S}^r$, then an aligned spin S_i^r will experience a local field

$$h_i(\underline{S}^{r*}) = \sum_{j=1}^{N} T_{ij} S_j^r - 2 \sum_{j \in F} T_{ij} S_j^r = h_i(\underline{S}^r) - 2 \sum_{j \in F} T_{ij} S_j^r \tag{4.3}$$

and so will also flip if

$$2 S_i^r \sum_{j \in F} T_{ij} S_j^r > S_i^r h_i(\underline{S}^r). \tag{4.4}$$

Consequently, if the original alignments $S_i^r h_i(\underline{S}^r)$ are not strong enough, a small number of flipped spins may be sufficient to induce misalignment in some of the initially unperturbed spins and project the state of the system on a trajectory taking it further from the nominal state $\underline{S}^r$.

In light of this fact it seems that the most plausible explanation for the tiny basins of attraction created around the nominal states by the algorithm of § 4.2 for higher values of α is that the solutions found tend to have small (though positive) spin to local field alignment $S_i^r h_i$.

Learning schemes have been proposed (Gardner *et al.* 1987[a,b], Pöppel and Krey 1987) which try to alleviate this problem. These involve iterating from noisy versions of nominal patterns. By sampling a large enough representation of noisy

patterns for each nominal one, such schemes attempt to train the network to explicitly associate distorted patterns with the appropriate uncorrupted nominal ones.

The approach adopted here (similar to Diederich and Opper 1987, Gardner 1987, 1988, Krauth and Mézard 1987) attacks the problem of weak alignment between spin and local field directly by attempting to find solutions for the T_{ij} subject to the more stringent condition

$$S_i^r h_i(\underline{S}^r) > B > 0 \tag{4.5}$$

on the alignment of each spin S_i^r.

However, when enforcing this constraint, one must ensure that a simple scaling of all the connections is avoided, as the following will demonstrate.

For a given choice of connections T_{ij} and nominal patterns $\underline{S}^r, r = 1, 2, \ldots, p$, define the local fields

$$h_i(\underline{S}^r; \{T_{ij}\}) \equiv \sum_j T_{ij} S_j^r \quad \forall i \in [1, N]. \tag{4.6}$$

Now suppose $\exists$ { t_{ij} } with

$$S_i^r h_i(\underline{S}^r; \{t_{ij}\}) > B/k \;\; (k > 1) \quad \forall r \in [1, p], \;\; \forall i \in [1, N], \tag{4.7}$$

then define

$$T_{ij} \equiv k t_{ij} \quad \forall i, j \in [1, N] \tag{4.8}$$

(i.e., simply scale up all the connections by factor k). Then one has that

$$S_i^r h_i(\underline{S}^r; \{T_{ij}\}) = k S_i^r h_i(\underline{S}^r; \{t_{ij}\}). \tag{4.9}$$

Thus

$$S_i^r h_i(\underline{S}^r; \{T_{ij}\}) > B, \quad \forall r \in [1, p], \;\; \forall i \in [1, N]. \tag{4.10}$$

Hence, finding a set of connections satisfying the constraint (4.5) can be done by finding a solution to the simpler problem (4.7) (with the lower bound B/k) and then rescaling them by a factor k.

Note that the dynamics of the network remain invariant under any such scaling, for

$$S_i(t+1) = sgn\left\{\sum_j (kt_{ij})S_j(t)\right\} = sgn\left\{\sum_j t_{ij}S_j(t)\right\}. \qquad (4.11)$$

To prevent rescaling of the connections occurring the bound B in (4.5) must be scaled to be of the same order of magnitude as $S_i^r h_i(\underline{S}^r)$.

Now if the connections T_{ij} are $O(T)$, then $h_i(\underline{S}^r)$ will consist of N terms, each being $O(T)$ and each randomly positive or negative. Thus h_i will be $O(T\sqrt{N})$, as will $S_i^r h_i$. To ensure that B was also $O(T\sqrt{N})$ in (4.5), the form employed was

$$B = M < |T_{ij}| > \sqrt{N}, \qquad (4.12)$$

where the angular brackets denote the average with respect to all the bonds in the ith row of the T_{ij} matrix:

$$< |T_{ij}| > \equiv \frac{1}{N}\sum_{j=1}^{N} |T_{ij}|. \qquad (4.13)$$

This form of B implies that M is $O(1)$, independent of N and of any rescaling of the T_{ij}s.

Thus the learning algorithm is as before (§ 4.2) but with a modified error mask:

$$\begin{aligned} \epsilon_i^r &= \frac{1}{2}\left[1 - sgn\left\{S_i^r \sum_{j=1}^{N} T_{ij}S_j^r - M < |T_{ij}| > \sqrt{N}\right\}\right] \\ T_{ij} &\longrightarrow T_{ij} + \frac{1}{N}\sum_{r=1}^{p}(\epsilon_i^r + \epsilon_j^r)S_i^r S_j^r, \qquad T_{ii} = 0. \end{aligned} \qquad (4.14)$$

4.5 Numerical Results

In order to determine whether the above algorithm could not only provide perfect storage of all $N\alpha$ nominal patterns but also endow the network with appreciable content-addressability, systems of $N = 256$ and $N = 512$ were simulated.

Once again, starting from the Hebbian storage prescription (for a given M and α), the T_{ij} were iterated using (4.14) and if the algorithm reached completion (all $\epsilon_i^r = 0$), the content-addressability of the resultant solution for the T_{ij} was tested in the same manner as before (§ 3.3 and § 4.3).

Depicted in figures 4.3 and 4.4, for $\alpha = 0.25$ and 0.5 respectively, is the fraction of nominal patterns perfectly recalled from various initial overlaps after completion of the learning algorithm.

In both graphs the trend from $N = 256$ to $N = 512$ indicates that the ordinary learning algorithm of § 4.2, corresponding to $M = 0$ here, will probably install negligible content-addressability as $N \to \infty$. In contrast, however, the perfect recall fraction f_p for a given $M > 0$ is seen to exhibit behaviour for increasing N reminiscent of the recall fraction of the Hopfield model in § 3.3. As m_0 increases, a similar change from f_p near 0 to near 1 is displayed with increasing steepness for larger N. This suggests that for these non-zero values of M imposition of the algorithm will result in appreciable content-addressability of the nominal patterns in the thermodynamic limit.

To investigate this behaviour for $N \to \infty$ quantitatively, a finite-size scaling analysis similar to that of § 3.3 was carried out, fitting a best-fit scaling form

$$\frac{f_p(m_0)}{1 - f_p(m_0)} = C \exp\{aN(m_0 - m_c)\} \tag{4.15}$$

to the perfect recall fraction f_p. These curves appear as solid ($N = 512$) and dashed ($N = 256$) lines in figures 4.3 and 4.4 for the non-zero values of M. As with figure 3.2, this form was tested directly by looking for a linear relationship between $\ln\{f_p/(1 - f_p)\}$ and m_0—figures 4.5 and 4.6.

This scaling form appears to fit reasonably well, although perhaps not so well as those in figures 3.1 and 3.2. However, if (4.15) were the appropriate form of $f_p(m_0)$, one would not expect it to fit the numerical results as closely as those in figure 3.1 since the system sizes dealt with here are not so large—recall

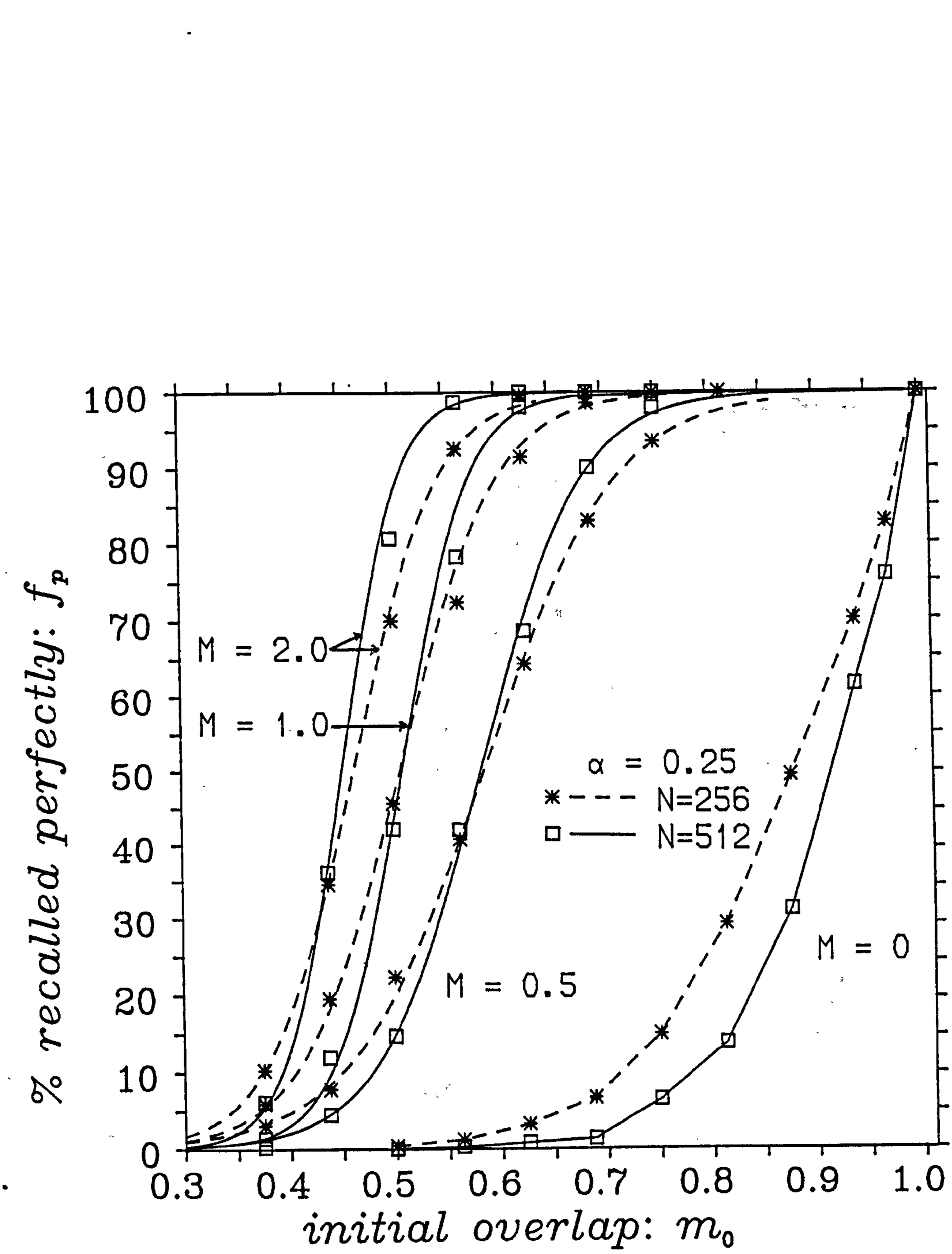

Figure 4.3: Perfect recall fraction after learning with (4.14) at $\alpha = 0.25$

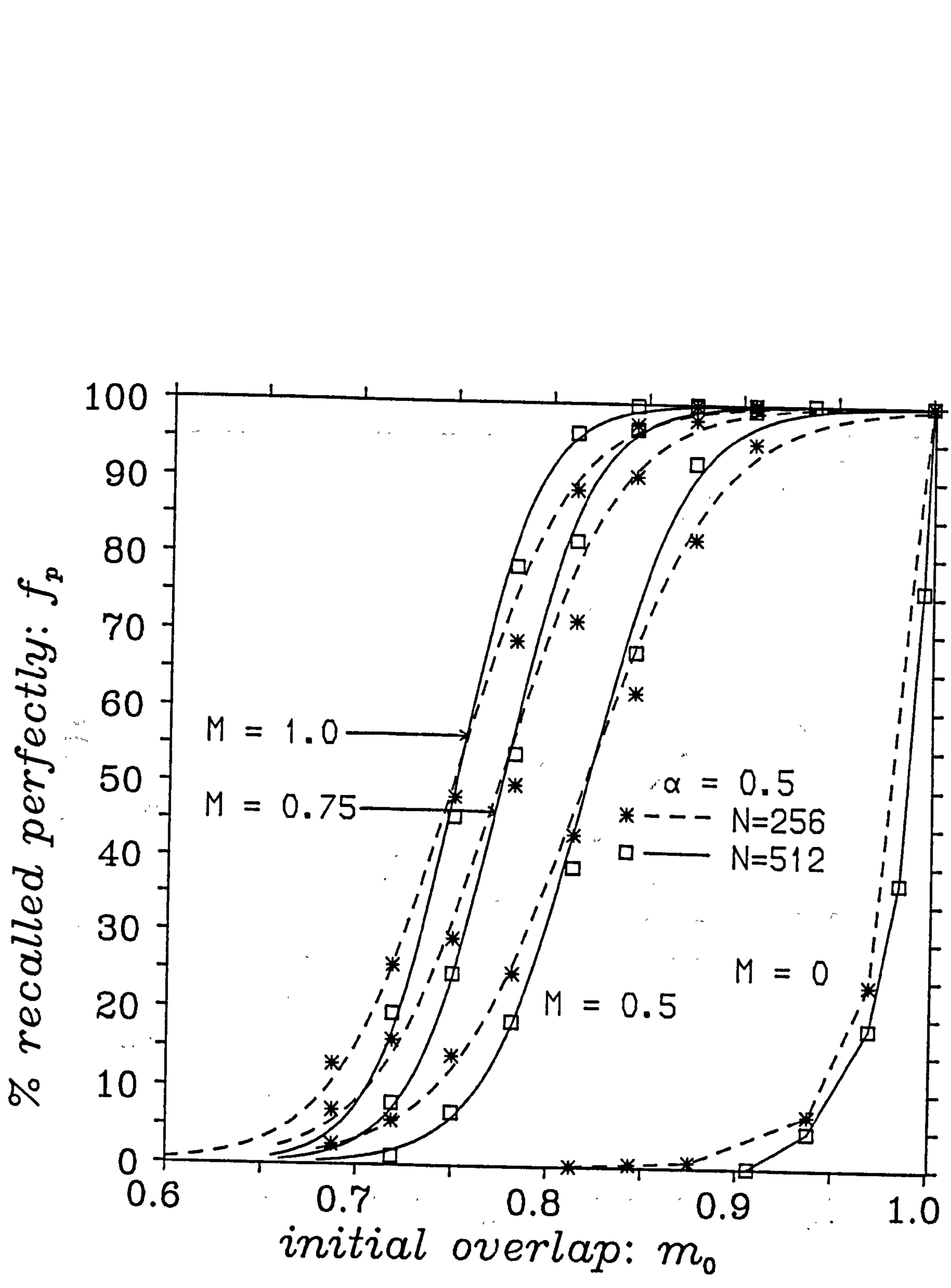

Figure 4.4: Perfect recall fraction after learning with (4.14) at $\alpha = 0.50$

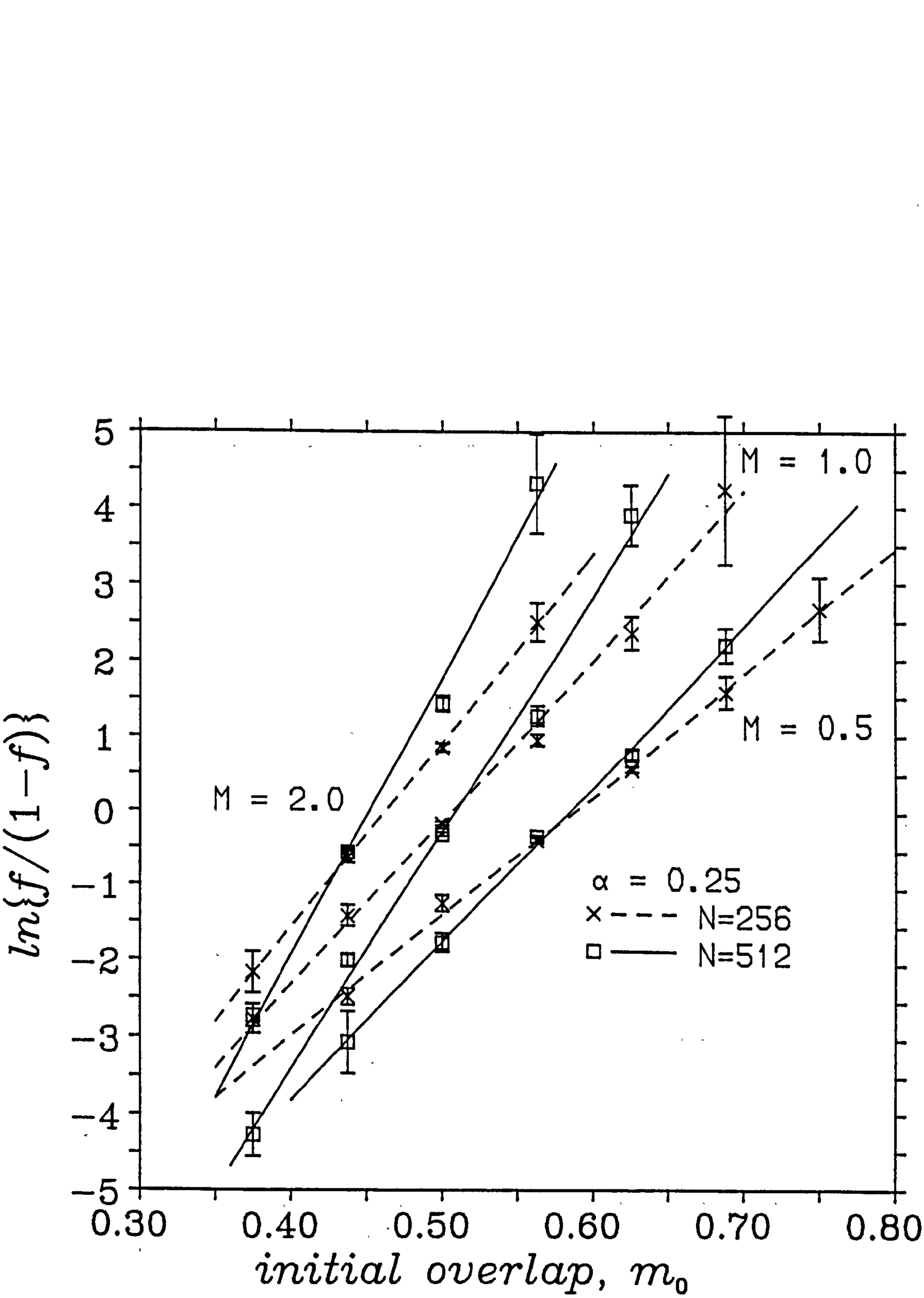

Figure 4.5: Testing the relationship (4.15) at $\alpha = 0.25$

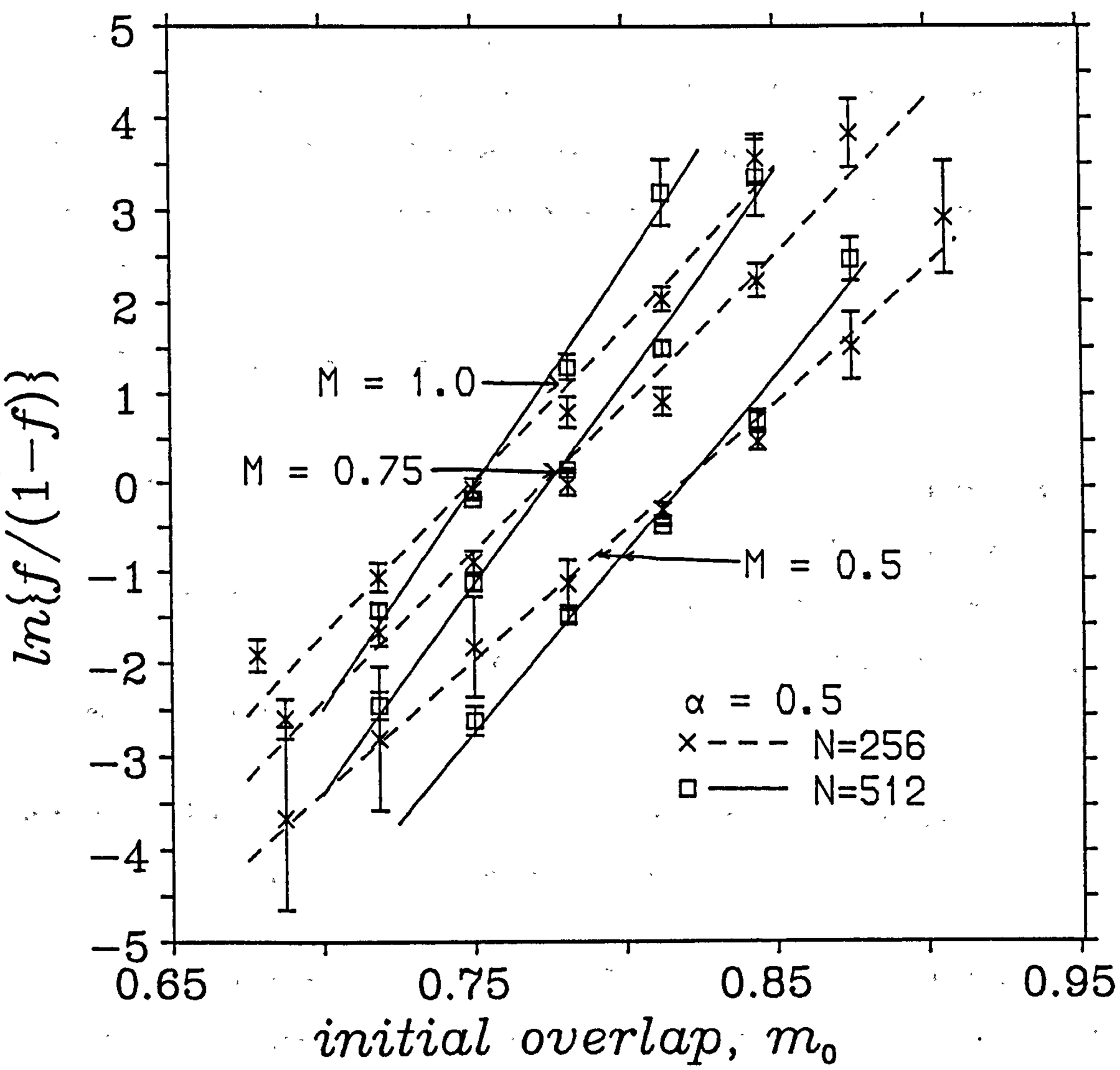

Figure 4.6: Testing the relationship (4.15) at $\alpha = 0.5$

that $N = 512, 1024$ and 2048 were simulated in § 3.3—and so any systematic corrections to scaling would be more prominent for $N = 256$ and 512.

A similar extrapolation to $N^{-1} \to 0$ yielded the following estimates for the critical minimum overlaps $m_c(\alpha, M)$ in (4.15):

$$m_c(\alpha = 0.25, M = 2.0) = 0.44(2); \qquad m_c(0.5, 1.0) = 0.75(3)$$

Of course, there were only two system sizes available and so the extrapolation was simply achieved by extension of the straight line defined by the two points at $N^{-1} = 1/256$ and $N^{-1} = 1/512$, whereas the extrapolation of § 3.3 used three sizes of N^{-1} and so involved a least-squares best-fit line through the points.

4.5.1 The Rate of Learning

This was monitored by noting the fraction of bit errors at each learning cycle. A learning cycle was defined as one complete iteration of (4.14) through all of the $N\alpha N$-bit nominal patterns, and the fraction of bit errors was the fraction of all the corresponding $N^2\alpha$ spins which were incorrectly stored at that stage of the learning algorithm.

The rate at which learning proceeded with respect to reducing this fractional error was exponential, as can be seen from figure 4.7.

As M increases, learning becomes more difficult and this exponential learning rate decreses. Presumably it will tend to zero as M approaches the optimal value attainable. The gradual change in curvature for $M = 2$ at $\alpha = 0.25$ and $M = 1$ at $\alpha = 0.5$ in figure 4.7 suggests that at the maximum attainable M, the number of errors may still initially decrease but then tail off to a constant value as learning proceeds.

An indication of the total effort needed to learn to completion as M is increased

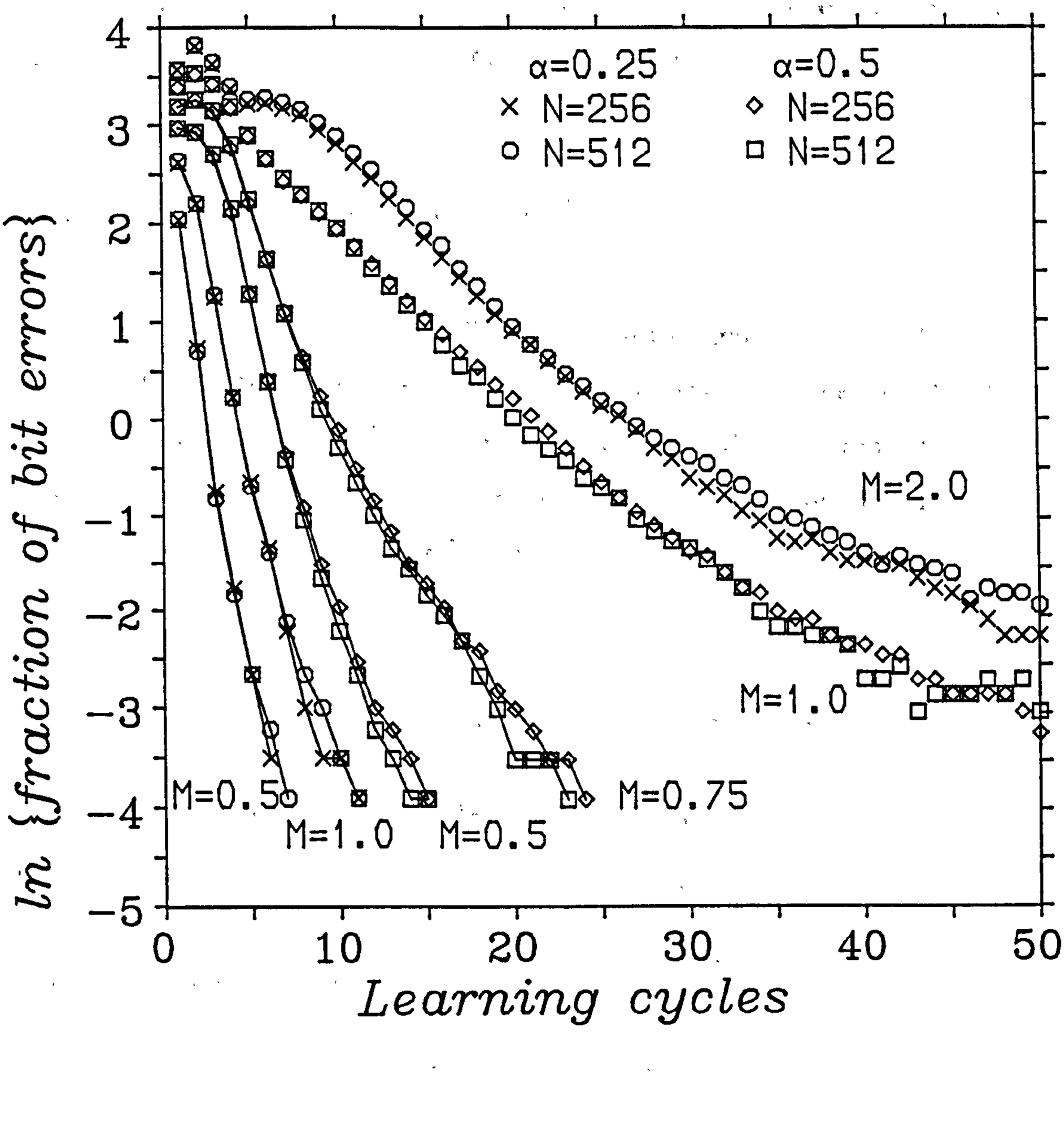

Figure 4.7: Learning rate

is given in figure 4.8. One 'learn' consists of a sweep through a nominal state if it has at least one spin wrongly stored. So, during a learning cycle, a nominal pattern contributes one to the total number of learns only if it has at least one spin wrongly stored at that time. The number of such learns required is seen to grow rapidly as M approaches the largest values—$M = 2$ at $\alpha = 0.25$ and $M = 1$ at $\alpha = 0.5$— attempted here. These values of M may thus be near the optimum values possible (at least for symmetric T_{ij}).

4.5.2 The Optimal Value of M

If the synaptic connections are allowed to assume general asymmetric values, then Gardner (1987, 1988) has shown that it is possible to calculate the maximum possible spin to local field alignment for a general set of connections, whose only restriction is that they are normalised to prevent them being arbitrarily rescaled *a la* § 4.4. This is done by enforcing

$$\sum_{j \neq i} J_{ij}^2 = N \tag{4.16}$$

on the connections J_{ij}. This is an alternative precaution to one such as (4.12).

The optimal value of the spin to local field alignment K, i.e.,

$$\frac{1}{\sqrt{N}} S_i^r \sum_{j \neq i} J_{ij} S_j^r > K \tag{4.17}$$

was then shown to satisfy

$$\frac{1}{\alpha} = \frac{1}{\sqrt{2\pi}} \int_{-K}^{\infty} (t + K)^2 e^{-\frac{1}{2}t^2} dt. \tag{4.18}$$

The formalism of § 4.4 can be cast into a similar form to expose the relationship between the bounds K and M. First of all, the connections T_{ij} must be rescaled to conform to (4.16). Define new connections

$$R_{ij} \equiv \frac{T_{ij}}{\sqrt{N^{-1} \sum_{j=1}^{N} T_{ij}^2}}, \tag{4.19}$$

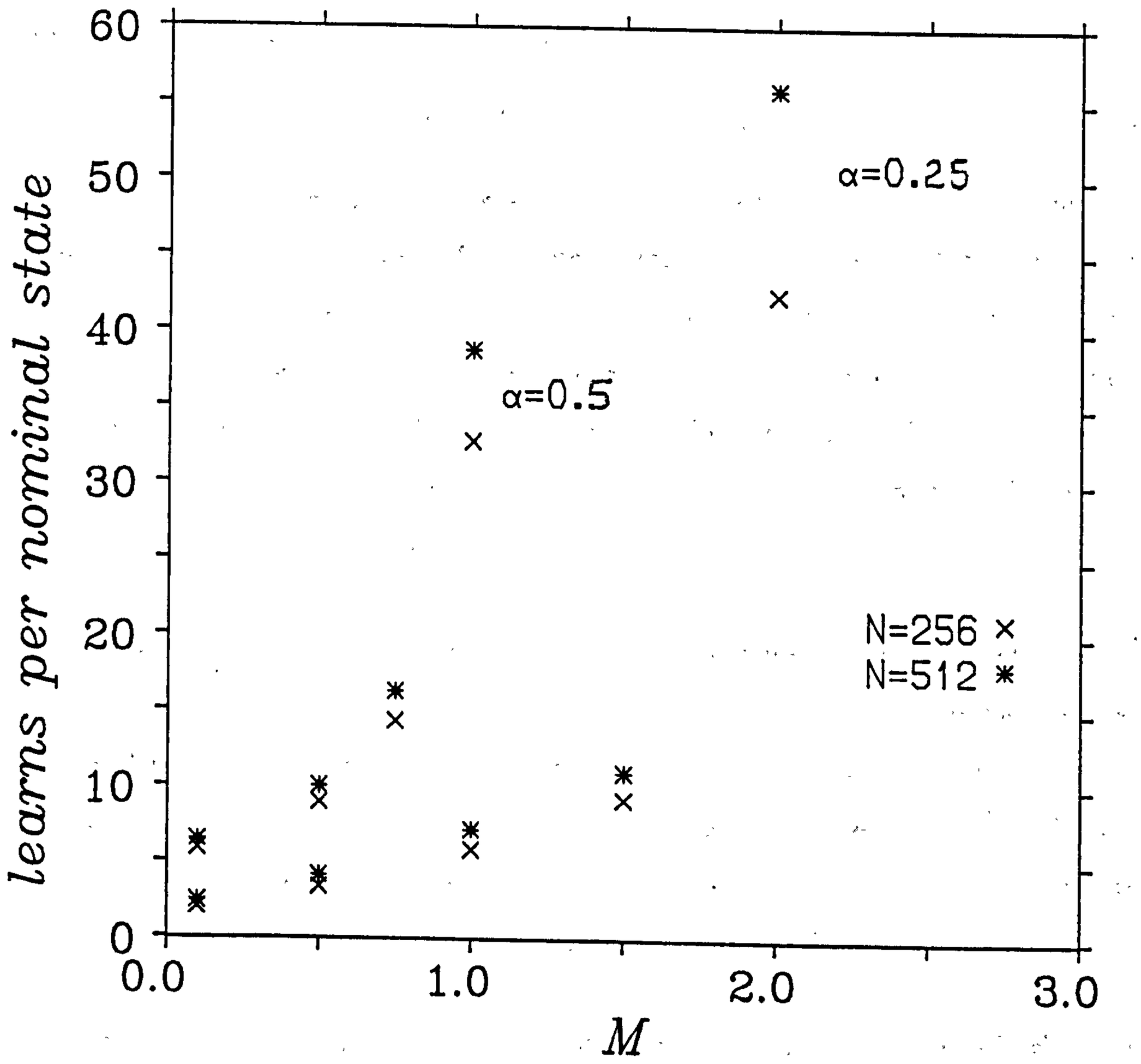

Figure 4.8: The number of learns required per nominal vector

which then satisfy (4.16). (Recall that the neural update dynamics are invariant under any such rescaling.)

Then (4.5), which by virtue of (4.12) is

$$S_i^\tau \sum_{j=1}^{N} T_{ij} S_j^\tau > M \frac{1}{N} \sum_{j=1}^{N} |T_{ij}| \sqrt{N} \tag{4.20}$$

becomes

$$\frac{1}{\sqrt{N}} S_i^\tau \sum_{j=1}^{N} R_{ij} S_j^\tau > M \frac{1}{N} \frac{\sum_{j=1}^{N} |T_{ij}|}{\sqrt{\frac{1}{N}\sum_{j=1}^{N} T_{ij}^2}}. \tag{4.21}$$

Thus the formalism employed in § 4.4 is equivalent to (4.16,17) with

$$K \equiv M \frac{1}{N} \frac{\sum_{j=1}^{N} |T_{ij}|}{\sqrt{\frac{1}{N}\sum_{j=1}^{N} T_{ij}^2}}, \tag{4.22}$$

except, of course, for the added restriction of symmetry on the connections T_{ij}. The relation between K and M becomes even clearer due to the fact that, after completion of the learning algorithm (4.14), the distribution of the T_{ij} remains Gaussian and of zero mean—as is evidenced by the histograms in figures 4.9 and 4.10, which show the final distribution of the absolute value of the connections T_{ij} at $\alpha = 0.25, M = 2.0$ and $\alpha = 0.5, M = 1.0$. Best-fit Gaussian forms are provided on these histograms.

Note that the width of the final $|T_{ij}|$ distribution is *narrower* by a factor of $\sqrt{2}$ for $N = 512$ than for $N = 256$. This is consistent with the expectation that the width of the integer synapse distribution NT_{ij} should be proportional to the square root of the number of patterns being learnt, i.e., to $\sqrt{N\alpha}$. This is because the number of modifications to a typical T_{ij} should be proportional to $p = N\alpha$ and these modifications are approximately random.

Hence each T_{ij} in (4.22) is a random variable following a Gaussian distribution of zero mean, and thus the ratio multiplying M on the right-hand side of (4.22) is, for large N, the ratio of the absolute value to the width of a Gaussian variable of zero mean, which is

$$2 \frac{\int_0^\infty z e^{-z^2/2\sigma^2} dz}{\int_{-\infty}^{\infty} z^2 e^{-z^2/2\sigma^2} dz} = \sqrt{\frac{2}{\pi}}.$$

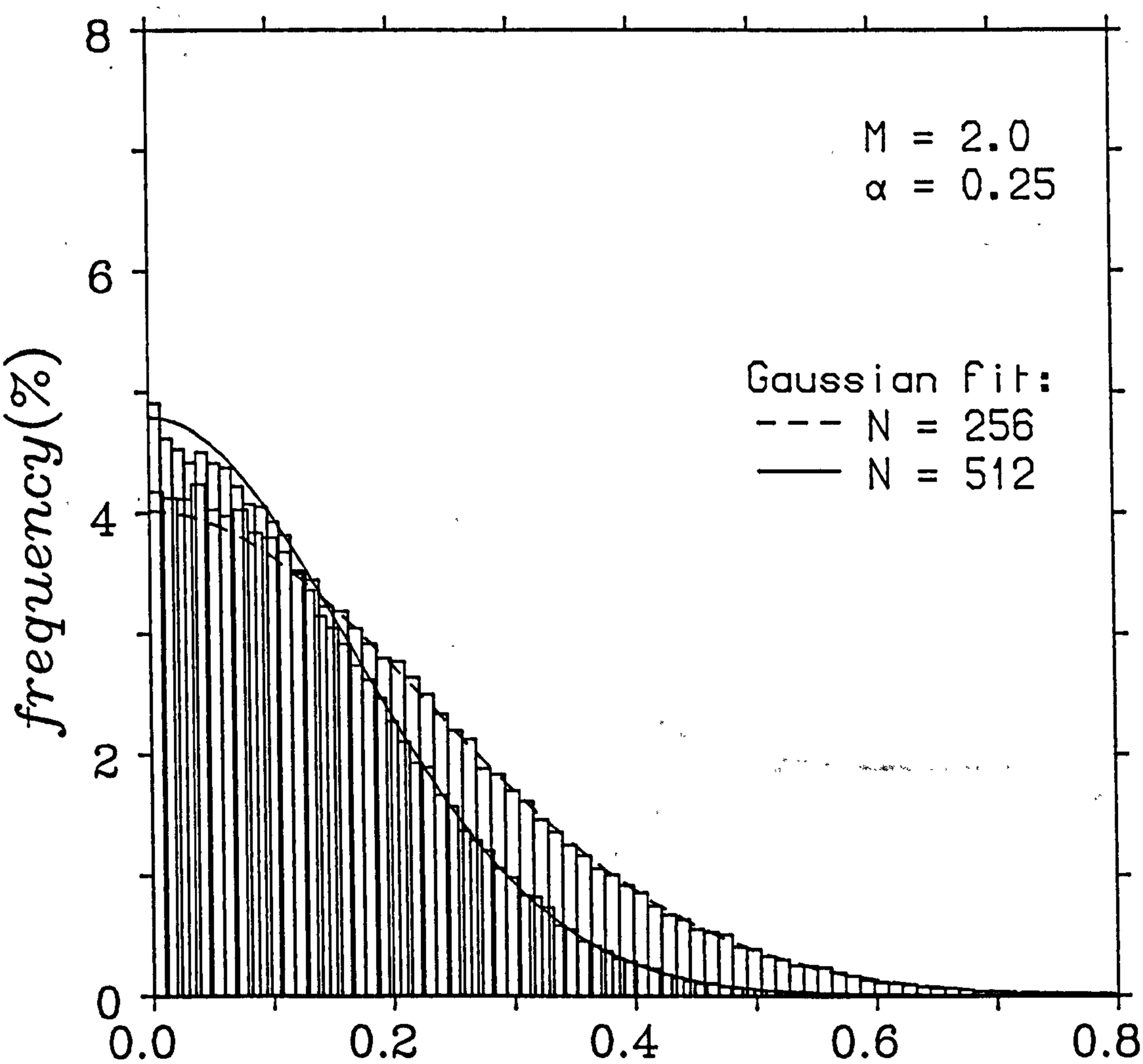

Figure 4.9:Distribution of connections after learning: $\alpha = 0.25$

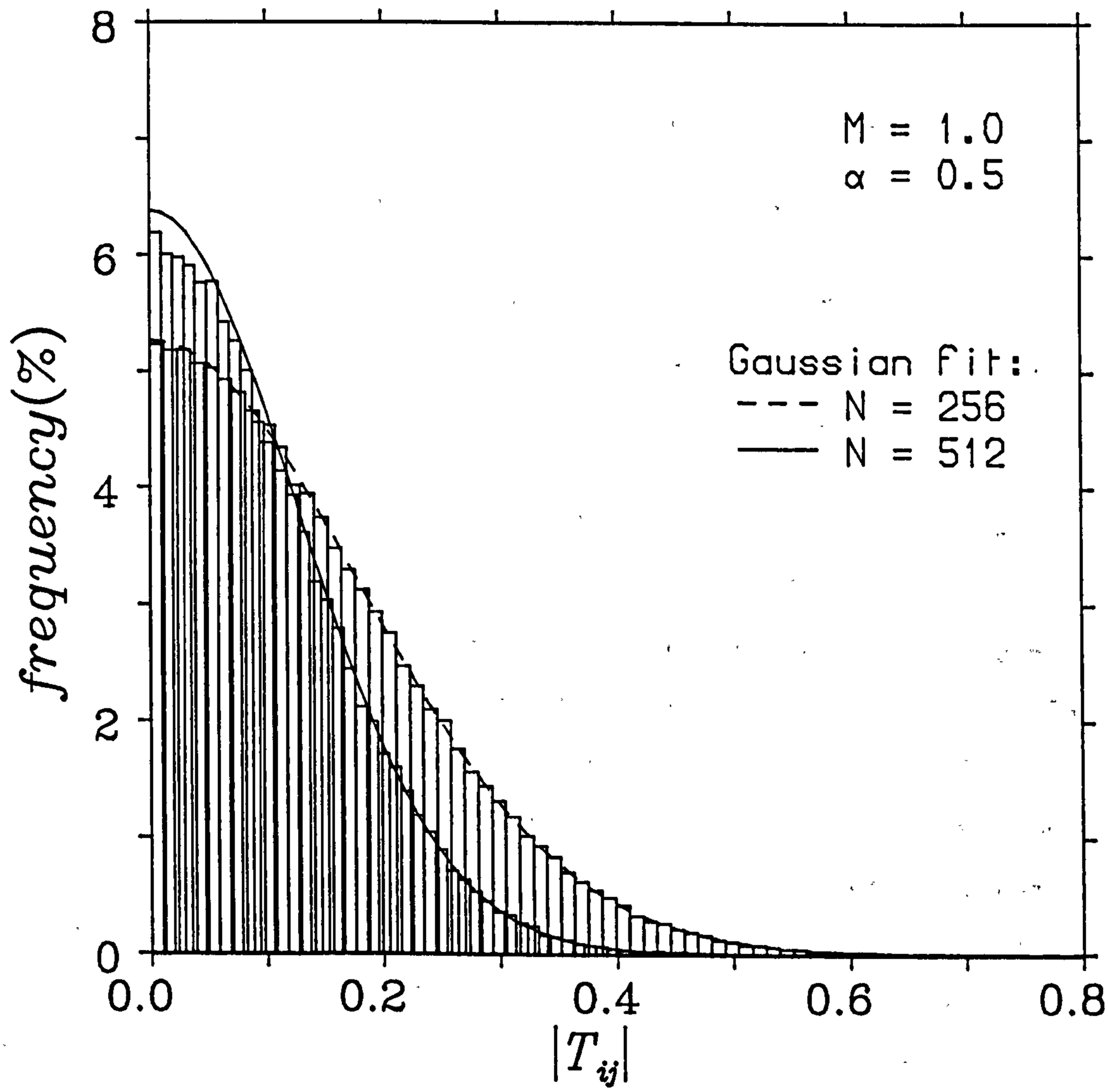

Figure 4.10: Distribution of connections after learning: $\alpha = 0.5$

Thus

$$M \equiv K\sqrt{\frac{\pi}{2}}. \tag{4.23}$$

Now, from (4.18), the optimal values of $K(\alpha)$ for $\alpha = 0.25$ and 0.5 are

$$K(0.25) \simeq 1.74, \quad K(0.5) \simeq 1.04,$$

so that the corresponding values of $M(\alpha)$ for asymmetric connections are

$$M(0.25) \simeq 2.18, \quad M(0.5) \simeq 1.30.$$

These lie above the largest values $M(0.25) = 2$ and $M(0.5) = 1$ which were attempted in the numerical work of this section for which learning became rapidly more difficult (figure 4.8).

After completion of the learning algorithm (4.14) the alignment of each spin with its local field in any nominal pattern satisfies (4.5), i.e, $S_i^\tau h_i > B$. In fact the majority of the alignments are not much greater than B: figure 4.11 shows the distribution of these alignments after learning at $\alpha = 0.5$ for different values of M. Increasing M does not appear to significantly alter the shape of the final distribution, but only has the effect of shifting it further along the $S_i^\tau h_i$ axis. The curve for each value of M rises sharply from zero to a maximum just after B (which is the lower bound on the alignment) and then tails off more slowly along the axis. It may be that as $N \rightarrow \infty$ the increase of these curves at B approaches a discontinuous jump to the maximum. Analysis of larger systems would be required to expose any such trend and possible scaling behaviour of these distributions. The curve for $M = 0$, corresponding to the algorithm (4.1,2), confirms the fears expounded at the beginning of § 4.4, i.e., that despite ensuring perfect spin alignments in every nominal pattern, the resultant alignments are mostly grouped around a value not much greater than zero. It is however interesting to note that while in figure 4.4 the shape of the graphs and the trend from $N = 256$ to 512 indicate that non-zero content-addressability will probably be achieved for $M = 0.5$, but not for $M = 0$, the

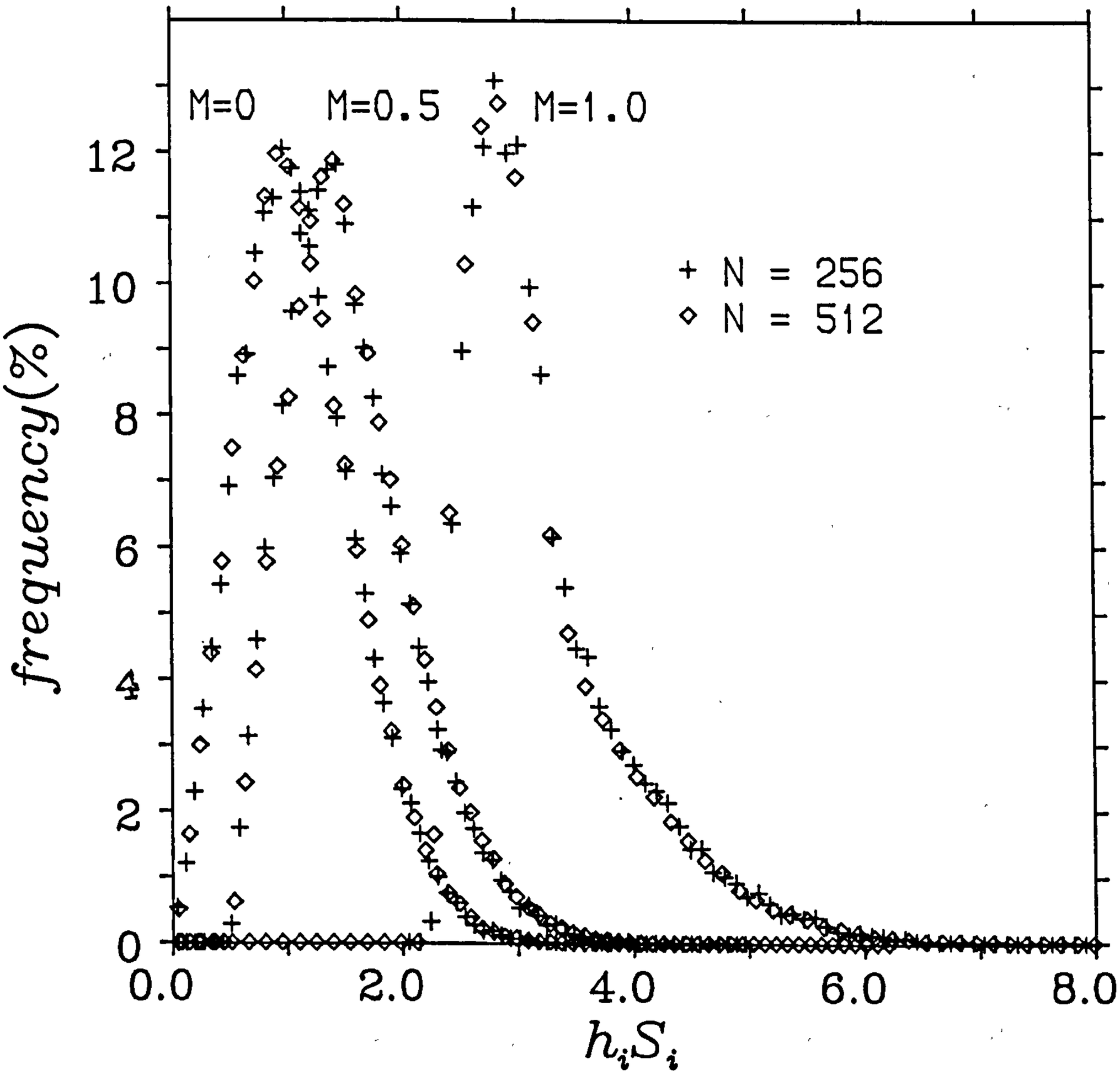

Figure 4.11: Final distribution of alignments of spin and local field

final distributions in figure 4.11 indicate that the corresponding alignments are only slightly greater for $M = 0.5$ than for $M = 0$.

4.6 Discussion

The results presented in § 4.3 have shown that, despite achieving perfect storage of all $N\alpha$ nominal patterns for α up to at least 1.0, the 'naive' learning algorithm (4.1,2) only provides appreciable content-addressability for $\alpha \leq 2.0$ at $N = 512$. The solution it finds for higher values of α look like being of no use for large N.

However, if strong stability of the spins in each nominal configuration is enforced by generalising (4.1,2) to (4.14) then it has been shown in § 4.5 that good content-addressability in addition to perfect storage can be achieved for non-zero values of M in (4.14) for α up to at least 0.5. The trend for increasing N suggests qualitatively that this will be the case as $N \rightarrow \infty$, albeit that the quantitative finite-size scaling analysis of § 4.5 may not be so accurate as that for the Hopfield model in § 3.3 due to the smaller system sizes N utilised.

It has been customary over the past few years to measure the performance of a neural memory in terms of the ratio of the number of patterns it can store (to within some tolerable accuracy) to the number of neurons in the network; i.e., the 'storage ratio' α. However, this ignores the typical size of connections T_{ij} required for the storage. A more appropriate measure which takes this into account would be the *relative storage capacity* —the number of bits stored to the number of bits of synaptic information used. This would be a much more relevant measure for hardware implementation since the number of bits needed to construct the synaptic connections is directly related to the amount of hardware required to build such a network. The number of bits required for the synapses T_{ij} after completion of the learning algorithm (4.14) is larger than for the Hop-

field model since a wider distribution of T_{ij} is produced (and in fact the width increases as M increases). However, the distribution remains Gaussian and the width of the integer synapse distribution NT_{ij} scales as $\sqrt{N}$, as it does for the Hopfield model. Hence the number of bits of synaptic information required will be $O(N^2 \ln \sqrt{N})$ in both cases, as there are $O(N^2)$ connections, each of $O(\sqrt{N})$, hence each comprising of $O(\log_2 N)$ bits $\equiv O(\ln \sqrt{N})$ bits. Therefore, since they both store $O(N^2\alpha)$ bits of information, their relative storage capacities—the number of bits of stored information per bit of synaptic information—will both be of the same order, namely $O(1/\ln \sqrt{N})$. Hence this learning algorithm is superior even in terms of relative storage capacity since it stores all $N^2\alpha$ bits exactly and provides a larger region of attraction around each nominal pattern. Nonetheless, the Hopfield model has been shown in § 3.3 to exhibit reasonable content-addressability at lower values of α, and even at higher values (such as α=0.5) it still provides a good starting point for the learning algorithms.

Comparison by numerical simulation of this type of algorithm with ones which involve iteration from noisy versions of nominal patterns would be required in order to gauge their relative merits and failings. Algorithms such as (4.14) involve only one iteration sweep per nominal state during learning and implicitly create content-addressability by enforcing at least a minimum stability on the spins in each nominal configuration. The 'learning with noise' algorithms, which install content-addressability explicity, test more than one noisy pattern per nominal pattern. The number of possible states differing from a nominal state in a given amount of sites (i.e., for a given degree of noise) increases exponentially in the number of sites which differ, although the number of states required to train on to achieve finite content-addressability may not necessarily suffer this exponential growth. This type of algorithm can provide the opportunity of creating anisotropic basins of attraction by choosing an appropriate set of noisy patterns to train on. By its very nature, it also attempts to recover nominal patterns from noisy versions in one sweep, although again only direct comparison

would reveal which type of algorithm provided fastest recall after the learning phases had been completed.

Simulations of much larger systems than $N = 256$ or 512 would be necessary to determine whether the basins of attraction continue to exhibit scaling behaviour for non-zero M, as the trend for increasing N suggests in § 4.5. Nevertheless, the numerical work of this chapter has demonstrated that memory with a high storage capacity (α up to at least 0.5) and near-optimal content-addressability can be achieved by a deterministic, local algorithm such as (4.14), and thus that it is unnecessary to resort to more computationally expensive stochastic algorithms as a means to this end. It would also be possible to tailor this algorithm to deal with more specialised cases which may be of interest, such as, e.g., networks with 'windowed' connections (Canning and Gardner 1988) or 'clipped' synapses (Parisi 1986, Sompolinsky 1987). The work presented here has dealt solely with random uncorrelated nominal bit patterns, but it has been shown (Gardner *et al.*1987[a,b]) that learning speed and performance can be improved when dealing with the storage of correlated patterns.

Chapter 5

Optimisation with Neural Networks

5.1 Introduction

The work presented thus far has concerned models of neural networks functioning as associative content-addressable memories. This, however, is only one of the many powerful capabilities of the nervous system. For example, the perceptual tasks demanded in image and speech recognition, which are constantly carried out by the nervous system, involve the processing of massive amounts of sensory data in a relatively short time. Indeed the time scale (fractions of a second) in which these tasks are completed is very impressive not only in view of the huge amount of information that must be processed, but also considering the response times (milliseconds) of the biological components (neurons) available in the nervous system. This is put even more into perspective when one considers the very fast components (of the order of nanoseconds) at the disposal of modern electronic hardware technology and the fact that there is as yet no computational tool constructed which has come anywhere near emulating such processing powers.

A great deal of the tasks which have to be confronted in such biological per-

ception can be formulated (at least partly) as optimisation problems—searching for the 'best' solution—, as indeed can many problems which arise in science and engineering or industry and commerce. One has a set of variables defined by the problem and seeks to find the values which minimise a *cost function* of the variables—a function which embodies the criteria of a 'good' solution to the problem (and which may also incorporate any constraints placed on the variables). The lower is the value of the cost function, the better is the solution. The solution which globally minimises the cost function is the 'best' or optimal solution. Indeed, the learning problems dealt with in the previous chapter could also be envisaged as optimisation problems: the variables to be chosen are the synaptic connection strengths and a cost function could be constructed containing terms which are lowest when each nominal state is accurately stored and has a large region of attraction.

Many of the optimisation problems encountered in the real world are very difficult to solve not only due to the large number of variables that may be involved (as is typically the case in perceptual tasks), but also because there can exist a very large number of solutions which are all local minima of the cost function but not necessarily deep minima. Thus, even though it may be sufficient to merely find a near-optimal solution (one of higher but comparable cost to the global minimum), the complexity (many-valleyed nature) of the cost function as it varies over the phase-space of variables can prohibit the use of simple deterministic techniques such as, e.g., gradient methods which search for a solution by only accepting changes in the variables which reduce the cost function. The variables of the problem do not have to be real-valued for this behaviour to occur. On the contrary, this can easily manifest itself in logical problems, where each variable can only assume two values, nominally 1 (true) and 0 (false). This is akin to the Ising spin-glass: the logical variables are the Ising spins (+1 or −1) and the cost function is the Hamiltonian or energy which, as mentioned in § 2.2 is many-valleyed, with the ground states ('near-optimal' solutions) ly-

ing amongst a plethora of local minima. This analogy prompted Kirkpatrick and co-workers (Kirkpatrick *et al.* 1983) to propose optimisation by *Simulated Annealing* (SA) as a general method for minimising cost functions of logical problems. This will be described in § 5.3.4. Such stochastic techniques as SA are powerful tools but they tend to be computationally expensive and slower to execute than deterministic methods.

5.2 The Model of Hopfield and Tank

The underlying philosophy of the seminal paper of Hopfield and Tank (1985) is that, given the computational power and speed of which real biological networks of neurons are capable, it should be possible to constuct simplified models of neural networks which can emulate their feats as long as the essential ingredients responsible for their power are maintained. The model retains two key features which manifest themselves in real neural networks: they have a high degree of parallelism as each neuron simultaneously and continually modifies its state in response to the net incoming signal from other neurons; and all the neurons are analogue devices. Undoubtedly the former characteristic (massive parallelism) is largely responsible for the overall speed with which biological information processing systems can operate. Hopfield and Tank claim that it is the latter feature—the non-linear analogue nature of the neurons—which endows the network with its computational power, its ability to find good solutions to problems with many-valleyed cost funtions.

The technique is applicable to logical optimisation problems where one attempts to globally minimise a cost function that can be expressed in the form

$$E(\underline{V}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} V_i\theta_i, \tag{5.1}$$

with each $V_i = 0$ or 1 only. The discrete problem is embedded in a continuous

space by assigning each logical variable ($V_i \in \{0,1\}$) to be the analogue firing rate of a neuron ($V_i \in]0,1[$).

Of particular importance for practical considerations is that it is straightforward to implement the network using standard electronic components. Each neuron is modelled by a voltage amplifier with a non-linear response function: the jth amplifier produces an output voltage V_j which depends on its input voltage u_j:

$$V_j = g(u_j), \tag{5.2}$$

where $g(u)$ is typically a sigmoidal function, e.g.,

$$g(u) = \frac{1}{1+e^{-gu}}. \tag{5.3}$$

The parameter g is the 'gain' which measures the steepness of the response through $u = 0$.

At its input each amplifier is connected to earth through a resistance ρ_i and capacitance C_i. The synaptic connection from the output of the jth neuron into the input of the ith is realised by a conductance $|T_{ij}|$. If $T_{ij} < 0$ then the connection is made between neuron i and j through an inverter at the output of neuron j. Otherwise a normal connection is made. (The connection architecture—the composition of the matrix $\{T_{ij}\}$—is at this point unspecified and is determined by the cost function in question. The only constraint is that it be symmetric: $T_{ij} = T_{ji}$.) Each amplifier also is supplied with an external input current θ_i. It can then be shown that the equation of motion governing the time evolution of the circuit is

$$C_i \frac{du_i}{dt} = \sum_{j=1}^{N} T_{ij} V_j - \frac{u_i}{R_i} + \theta_i, \tag{5.4}$$

where

$$\frac{1}{R_i} = \frac{1}{\rho_i} + \sum_{j=1}^{N} |T_{ij}|, \qquad V_j = g(u_j).$$

For the sake of simplicity, although it is not necessary, it is convenient to assign the same characteristics to each amplifier: $C_i = C$; $\rho_i = \rho$; and so $R_i = R$. Then,

redefining $T_{ij} \equiv T_{ij}/C$ and $\theta_i \equiv \theta_i/C$, (5.4) becomes

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} + \sum_{j=1}^{N} T_{ij} V_j + \theta_i, \qquad \tau \equiv RC. \tag{5.5}$$

Now consider the function

$$L(\underline{V}) \equiv -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij} V_i V_j - \sum_{i=1}^{N} \theta_i V_i + \frac{1}{\tau}\sum_{i=1}^{N} \int^{V_i} \frac{1}{g} \ln\left[\frac{1-t}{t}\right] dt. \tag{5.6}$$

Its time-derivative is, due to the symmetry $T_{ij} = T_{ji}$,

$$\begin{aligned} \frac{dL}{dt} &= -\sum_{i=1}^{N} T_{ij} V_j \frac{dV_i}{dt} - \sum_{i=1}^{N} \theta_i \frac{dV_i}{dt} + \frac{1}{\tau}\sum_{i=1}^{N} \frac{1}{g} \ln\left[\frac{1-V_i}{V_i}\right] \frac{dV_i}{dt} \\ &= -\sum_{i=1}^{N} \left\{ T_{ij} V_j + \theta_i - \frac{u_i}{\tau} \right\} \frac{dV_i}{dt} \frac{du_i}{dt} \\ &= -\sum_{i=1}^{N} \left(\frac{du_i}{dt}\right)^2 \frac{dV_i}{du_i}. \end{aligned} \tag{5.7}$$

But V_i is a monotonic increasing function of u_i, so $dV_i/du_i \geq 0$. Thus $dL/dt \leq 0$. Hence L is a Lyapunov function for the network: the network will always converge to a stable state (in which all the outputs of the neurons remain constant) which is a local minimum of $L(\underline{V})$. But from (5.6) and (5.1),

$$L(\underline{V}) = E(\underline{V}) + \frac{1}{\tau}\sum_{i=1}^{N} \int^{V_i} \frac{1}{g} \ln\left[\frac{1-t}{t}\right] dt \tag{5.8}$$

so that $L(\underline{V})$ is equivalent to $E(\underline{V})$ up to corrections of order g^{-1}. Therefore, for large gain g (steep response function (5.2)), the analogue network will evolve to a stable state $\underline{V}^*$ which is a local minimum of the cost function $E(\underline{V})$ to be minimised, and which is a valid solution for the discrete problem (as $g \gg 1$ implies that each $V_i \to 0$ or 1).

Of course, at this stage there would appear to be no guarantee that $\underline{V}^*$ will correspond to the optimal minimum or even a near-optimal minimum of E: its precise value will depend upon the choice of initial state $\underline{V}^0$ of the network. However, the reasons why deep minima of E should be found by the network were outlined by Hopfield and Tank as follows. The state space of this analogue circuit is the interior of the N-dimensional hypercube defined by $V_i = 0$ or 1, and

the state space of the problem to be solved are the 2^N corners of this hypercube. The network will evolve from an initial configuration $\underline{V}^0$ which lies in the interior of the hypercube to a configuration at a corner which is a minimum of E. On following its trajectory, the state of the system will thus avoid other minima of $E(\underline{V})$ which are associated with the discrete problem (at the corners of the hypercube). The larger the region of attraction around a minimum, the more likely that the trajectory will be caught by it. So a deep minimum of E is most likely to be found, assuming that the cost function is of the type where the deepest minima tend to have the largest regions of attraction. Therein lies the mechanism whereby local minima of the discrete problem can be avoided by a determinstic search through a continuous space obtained by allowing the discrete variables to assume analogue values.

The problem to which Hopfield and Tank applied this analogue neuron minimisation technique was the Travelling Salesman problem (TSP), a classic case of a difficult optimisation. A set of n cities is specified with their geographical locations in a plane and it is required to choose the shortest path (tour) which visits each city precisely once and returns to the starting city. The only way to solve this problem exactly is to compute the lengths of all possible tours, of which there are $(n-1)!/2n$ (the factor of $2n$ arises due to the degeneracy in the n choices of starting city and in the 2 choices of direction). This number becomes prohibitively large even for a moderate number of cities (e.g. $N \sim 50$). It is doubtful whether there exists any algorithm that can provide the exact or near-optimal answer in a computer time which does not grow faster than any finite power of the number of cities: i.e., this would be an example of an 'NP-complete' problem (execution of any algorithm which finds a solution requires a computational time that grows faster than any power of the size of the problem: a *non-polynominal* number of steps are involved).

Hopfield and Tank cast the TSP in a form amenable to the analogue neural network method in the following manner. For n cities, a network of $N = n^2$

neurons is required, each city being assigned a vector of n neurons, only one of which may have a non-zero state. The neuron with the non-zero value (of 1) designates the position of that city on the tour. There are n such vectors, thus constituting an $n \times n$ matrix of neural states. For a valid tour to be encoded in this manner, this matrix must be a permutation matrix, i.e., with only one non-zero entry in each row (a city can only appear once on the tour) and likewise in each column (only one city can be visited at any one instance). To ensure a valid solution, the cost function contains terms which are minimised if and only if the matrix is of this form. The cost function of course also contains a term proportional to the length of the tour so that it attains its global minimum at the shortest possible tour. Good performances were reported by Hopfield and Tank for $n = 10$ and 30, although they found that the choice of cost function parameters was more sensitive in the $n = 30$ case. Subsequent work (Wilson and Pawley 1987) suggests that Hopfield and Tank were fortunate in their choice of parameters even for $n = 10$ and that the network does not perform well, nor does it scale well—simulations of $n = 64$ were also attempted.

Nevertheless, it seems that the network does have the capability of producing results very quickly: stability was achieved within a few neural time constants τ, the response time of each neuron in (5.5).

5.3 Binary Image Restoration using Analogue Neurons

The problems (encountered by Wilson and Pawley) that the network has in finding good solutions to the TSP would seem to be largely due to the cumbersome nature of the cost function employed. An already complex problem is exacerbated out of the necessity of formulating a cost function of the form (5.1) containing logical variables, with the result that solutions can arise which do

not correspond to valid tours, in addition to the possibility of obtaining valid tour solutions of long path length. Also, as noted by Wilson and Pawley, the network has no way of deciding between the $2N$-fold degeneracy of solutions.

The optimisation problem to which the analogue neural network technique is applied here is that of restoring binary images which have been corrupted by noise—how to choose the image which is most likely to correspond to the original uncorrupted image, given the observed image. A binary image is one in which each pixel has either one of two intensities: black or white (nominally denoted by 0 or 1, respectively). This is a problem which maps more naturally than the TSP onto a neural network. The binary variables over which the optimisation takes place are simply the pixel intensities of the restored image, each neuron being assigned to one pixel. Hence the one-one mapping from pixels to neurons obviously preserves the topology of the problem: neighbouring pixels give rise to neighbouring (interconnected) neurons. Also, unlike the TSP, any of the 2^N corners of the N-dimensional hypercube (for N pixels) which constitute the phase-space are all valid solutions (as they are all binary images).

Restoration, or enhancement, of noisy images plays a crucial role in the early stages of vision perception. In most real-world situations the modules of *early vision* (the set of processes which recover the physical properties of three-dimensional surfaces from two-dimensional images) have to deal with images which have become corrupted by some noise processes. Just like many of the problems ocurring in the field of computer vision, image restoration can be formulated in the framework of optimisation theory. The cost functions involved are invariably very complicated and highly non-convex due to the inherent difficulties of dealing with the inverse problems which pervade computer vision: those of recovering surfaces from images—the direct problem, one of classical optics or computer graphics, is to obtain the projected image of three-dimensional objects. These inverse mappings from images to surfaces are usually unstable or ill-defined due to the information lost in the projection and/or the noise distortion. Thus to

ensure an unambiguous result (unique solution), it is often necessary to *regularise* the problem by using natural constraints (general *a priori* physical assumptions about the physical world) which, within an optimisation framework, corresponds to introducing appropriate terms into the cost function (hence adding to its complexity).

5.3.1 The Algorithm of Geman and Geman

The method proposed by Geman and Geman (1984) for restoring corrupted grey-level images is based upon a Bayesian approach. A cost function of the restored pixel intensities is constructed which assumes its global minimum when the retored image is the maximum *a posteriori* (MAP) estimate of the uncorrupted image, given the observed noisy image. In other words, given the observed data $\underline{D} \equiv \{D_1, D_2, \ldots, D_N\}$ (where D_i represents the intensity of the ith pixel in an image of N pixels), out of all the possible interpretations $\underline{I} \equiv \{I_1, I_2, \ldots, I_n\}$, the MAP estimate is that interpretation which maximises the conditional probability $P(\underline{I}|\underline{D})$—it is the image which was most likely to have been corrupted by the noise processes to produce the observed image $\underline{D}$. However, as mentioned above, the inverse mapping from the data to the interpretation is generally unstable or ill-defined. This problem can be overcome by the Bayesian approach which invokes Baye's rule to express the optimisation problem as

$$\max_{\underline{I}} \left\{ \frac{P(\underline{D}|\underline{I})P(\underline{I})}{P(\underline{D})} \right\} \tag{5.9}$$

which is just equivalent to

$$\max_{\underline{I}} \{P(\underline{D}|\underline{I})P(\underline{I}))\} \tag{5.10}$$

since the unconditional probability $P(\underline{D})$ of observing the data is independent of $\underline{I}$ and so plays no role in the optimisation procedure.

Now only the direct mapping from interpretation $\underline{I}$ to data $\underline{D}$ appears in (5.10). It does so through the first factor $P(\underline{D}|\underline{I})$, the probability of obtaining the

observed data given the interpretation. This simply depends upon the noise processes responsible for the corruption of the image. As with Geman and Geman, it will be assumed here that, to a good approximation, the noise processes amount to the addition of white Gaussian noise independently at each datum i, in which case

$$P(\underline{D}|\underline{I}) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\sum_{i=1}^{N} \frac{(D_i - I_i)^2}{2\sigma^2}\right\}, \tag{5.11}$$

where the noise has standard deviation σ and zero mean.

The other term appearing in (5.10) is $P(\underline{I})$, the *a priori* probability of the interpretation. Its evaluation requires specification of the type of possible clean images which could be observed in the absence of noise. Thus $P(\underline{I})$ embodies the prior model of uncorrupted images, and its presence in (5.10) acts as a regulariser for the problem—it is based on *a priori* knowledge (assumptions of the distribution of clean images) and so restricts the class of admissable solutions.

In the Geman and Geman algorithm the specification of $P(\underline{I})$ is achieved through the assumption that any clean image can be modelled as the realisation of a discrete two-dimensional Markov Random Field (MRF). This is a generalisation of a Markov chain to two dimensions and simply means that each pixel intensity is treated as a random variable whose value only depends on those pixel intensities within some finite neighbourhood:

$$P(I_i|I_j, j = 1, 2, \ldots, N) = P(I_i|I_j, j \in \mathcal{G}_i) \tag{5.12}$$

where $\mathcal{G}_i$ denotes the set of neighbouring pixels of pixel i.

Through the equivalence between an MRF and a Gibbs distribution (Geman and Geman 1984, Besag 1972), the prior probability of the whole image may be concisely expressed as

$$P(\underline{I}) = \frac{e^{-U(\underline{I})}}{Z} \tag{5.13}$$

where Z is the normalising *partition function* and $U(\underline{I})$ is the *prior potential* or *energy* of the image and is dependent on the model chosen. Images which

are most likely in the context of a particular model would then be assigned the lowest energies.

Now, noting that ln is a monotonic increasing function, (5.10) is equivalent to

$$\max_{\underline{I}} \{\ln P(\underline{D}|\underline{I}) + \ln P(\underline{I})\} \tag{5.14}$$

and using (5.11) and (5.13), the problem to be solved is then

$$\min_{\underline{I}} \left\{ U(\underline{I}) + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (D_i - I_i)^2 \right\}, \tag{5.15}$$

where irrelevant constant terms have been dropped. Thus it is required to minimise a cost function containing a regulariser $U(\underline{I})$ and a least squares estimator, but which is typically highly non-convex.

5.3.2 Analogue Neural Network Implementation

Up until now the description of the Geman and Geman algorithm has been for general grey-level images, culminating in the need to minimise a cost function of multi-valued variables. For the technique of analogue neural network (henceforth ANN) minimisation to be applicable however, the cost function must be one of binary variables, as was explained in § 5.2. This will obviously be the case when the images to be restored are binary. In that case the noise term $P(\underline{D}|\underline{I})$ needs some modification. The Gaussian white noise N_i at site i will still be added to the original intensity I_i, but D_i, the observed intensity, will now be obtained by truncating the net intensity to $\theta(I_i + N_i - \frac{1}{2})$, where $\theta(x)$ is the usual threshold function. Thus

$$P(\underline{D}|\underline{I}) = \prod_{i=1}^{N} P(D_i|I_i) \quad \text{with} \quad P(D_i|I_i) = \begin{cases} p & \text{if} \quad D_i \neq I_i \\ 1-p & \text{if} \quad D_i = I_i \end{cases} \tag{5.16}$$

where $p = P(N_i > \frac{1}{2}) = P(N_i < \frac{1}{2}) = \text{erfc}(1/\sigma 2\sqrt{2})$. Hence specifying σ is just equivalent to specifying the probability p that a pixel intensity will 'flip' due to the noise.

Taking the logarithm of (5.16) then implies

$$\ln P(\underline{D}|\underline{I}) \equiv \sum_{i=1}^{N} \ln P(D_i|I_i) = \sum_{i:D_i \neq I_i} \ln p + \sum_{i:D_i = I_i} \ln(1-p). \tag{5.17}$$

Since I_i and $D_i \in \{0,1\}$, this may be written

$$\ln P(\underline{D}|\underline{I}) = \sum_{i=1}^{N} \{[(2D_i - 1)I_i - D_i + 1]\ln(1-p) + [(1-2D_i)I_i + D_i]\ln p\}, \tag{5.18}$$

which is of a form consistent with (5.1) since the variables $\underline{V}$ to be minimised over correspond to $\underline{I}$ here.

It now remains to consider the form of $\ln P(\underline{I})$ in (5.14). This is determined through the particular choice of prior potential $U(\underline{I})$ in (5.13). In general,

$$U(\underline{I}) = \sum_{c \in \Gamma} V_c(\underline{I}), \tag{5.19}$$

where Γ is the set of 'cliques' of the neighbourhood system of the MRF: a clique being a set of pixels, each of which is a neighbour of all the other pixels in that clique. For simplicity, the clique potentials $V_c(\underline{I})$ can be restricted to pairwise interactions:

$$U(\underline{I}) = \sum_{i=1}^{N} \sum_{j \in \mathcal{G}_i} V_2(I_i, I_j), \tag{5.20}$$

where $\mathcal{G}_i$ is the set of neighbours of pixel i. Then, in order for $U(\underline{I})$ to have a form consistent with the ANN cost function (5.1), a straightforward Ising-like interaction can be used as in (Geman and Geman 1984 and Murray *et al.* 1986):

$$V_2(I_i, I_j) = \begin{cases} -A & I_i = I_j \\ +A & \text{otherwise} \end{cases} \quad (A > 0), \tag{5.21}$$

since for binary variables this is just the closed form

$$V_2(I_i, I_j) = -A(2I_i - 1)(2I_j - 1). \tag{5.22}$$

Hence this produces

$$\begin{aligned} \ln P(\underline{I}) &= -\ln Z - U(\underline{I}) \\ &= -\ln Z + \sum_{i=1}^{N} \sum_{j \in \mathcal{G}_i} A(2I_i - 1)(2I_j - 1) \end{aligned} \tag{5.23}$$

and so, incorporating this and (5.18) into (5.14), the cost function to be minimised is

$$E(\underline{I}) = -\sum_{i=1}^{N}\sum_{j\in\mathcal{G}_i} A(2I_i-1)(2I_j-1) - \sum_{i=1}^{N}(2D_i-1)\ln(p^{-1}-1)\,I_i. \qquad (5.24)$$

The first term is the prior potential $U(\underline{I})$, while the second represents the faithfulness to the data. Up to an additive constant, $E(\underline{I})$ is identical to the Hamiltonian of a set of Ising spins $S_i \equiv 2I_i - 1$ with ferromagnetic coupling constant $2A$ and subject to a random field $\pm\frac{1}{2}\ln(p^{-1}-1)$ at each lattice site (pixel). The size of the parameter A determines the granularity of the images in the prior distribution (5.13). The larger is A, the larger will be the piecewise constant patches (of black or white) in the images. In fact the role of A is analogous to an inverse temperature parameter in the Gibbs distribution in (5.13).

The weight of the second term depends on the amount of noise present (the size of p). Two limiting cases should be observed. The limit of no noise ($p \to 0$) has $\ln(p^{-1}-1) \to \infty$ and the data-dependent term in E dominates, so that the minimum is achieved when the restored image is identical to the observed data ($\underline{I} \equiv \underline{D}$). In the other limit of maximum noise ($p \to \frac{1}{2}$), all information is lost as the data and original image become totally uncorrelated: $\ln(p^{-1}-1) \to 0$ and the data term plays no part in E.

The cost function (5.24) is now in a form consistent with the ANN cost function (5.1) with

$$T_{ij} = \begin{cases} 8A & j \in \mathcal{G}_i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \theta_i = -4An(\mathcal{G}_i) + (2D_i-1)\ln(p^{-1}-1), \qquad (5.25)$$

where $n(\mathcal{G}_i)$ denotes the number of pixels in the neighbourhood of pixel i.

Note that the connections are indeed symmetric since $j \in \mathcal{G}_i \Leftrightarrow i \in \mathcal{G}_j$, and they are also local (to within $\mathcal{G}_i$).

The ANN implementation then consists of assigning a neuron to each pixel. The (analogue) firing rate I_i of the ith neuron represents the pixel intensity of the

ith pixel and is a sigmoidal response function of its input potential u_i:

$$I_i(u_i) = \frac{1}{1 + e^{-gu_i}}, \tag{5.26}$$

where the equation of motion for u_i is, by virtue of (5.5),

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} + 8A \sum_{j \in \mathcal{G}_i} I_j - 4An(\mathcal{G}_i) + (2D_i - 1)\ln(p^{-1} - 1). \tag{5.27}$$

5.3.3 Numerical Simulations

The local nature of the connections T_{ij} in (5.25)—each being restricted to lie within a neighbourhood $\mathcal{G}_i$ of pixel i—means that in a numerical simulation of the ANN implementation (5.27) more than one pixel may be updated at a given instant. All pixels which are not mutual neighbours with respect to the neighbourhood system $\mathcal{G}_i$ are independent of each other's state at a given instant and so all may be updated concurrently. This opportunity of parallelism was exploited by simulating the ANN on the ICL DAP. Images of 64×64 pixels (i.e., $N = 4096$) were processed so that one pixel was assigned to one DAP processing element. The simple neighbourhood system of $\mathcal{G}_i$ being only the nearest neighbours of pixel i was used. (Thus in (5.27) $n(\mathcal{G}_i) = 2, 3$ or 4 for a corner, side or interior pixel site respectively). This meant that when a pixel was being updated, the p.e. assigned to it needed only to communicate with its four nearest-neighbour p.e.'s, which could be accomplished by simple SHIFT operations on the DAP. (Details of the DAP code are provided in appendix C.) For this choice of $\mathcal{G}_i$ it was possible to update half of the pixels simultaneously. Imagining the pixel sites each being coloured black or white in a chequerboard fashion, then any white site need only communicate with black sites and vice versa. Thus all $\frac{1}{2}N$ white sites may be updated in parallel, then all $\frac{1}{2}N$ black sites. (This was achieved in the DAP code by constructing an appropriate chequerboard logical mask—see appendix C.)

A good starting point for the network is obviously the observed image itself, $\underline{D}$.

However, simply initialising $\underline{I}_0 = \underline{D}$ would result in unbounded initial potentials u_i in (5.26) as $u_i = g^{-1} \ln \{ (1 - I_i)/I_i \}$ and $D_i = 0$ or 1. Instead $\underline{I}$ was initialised at $I_i = | D_i - \delta |$, where δ followed a Gaussian distribution of mean μ and small width.

In order to simulate the differential equation (5.27) properly, the neural potentials $u_i(t)$ must be updated at small enough time intervals δt so that it may be approximated by a linearisation:

$$u_i(t + \delta t) = u_i(t) + \frac{du_i}{dt}\delta t \tag{5.28}$$

where $\delta t \ll \tau$, the characteristic response time of the neurons. In the simulations here, as with Hopfield and Tank (1985), without loss of generality, τ was taken to be 1. A value of $\delta t = 10^{-3}$ was then deemed to be small enough—smaller values of δt did not alter the results significantly, but merely necessitated a larger number of iterations to produce a similar outcome. The network was considered to have reached stability if the change δI_i in each neural firing rate satisfied $|\delta I_i| < 10^{-6}$ after a complete updating sweep. This tolerance was considered appropriate since $\delta I_i = (dI_i/du_i)(du_i/dt)\delta t$, i.e., $\delta I_i \propto \delta t$ and $10^{-6} \ll \delta t$.

Figure 5.1 depicts the performance of the network in restoring an original image (top) of concentric rings which had been distorted (centre) with 25% noise ($p = 0.25$). For this restoration (bottom), as with all the other results which appear in this work, $g = 10$ and $A = 2$. Figure 5.2 similarly depicts an original, (25%) distorted and restored image of an 8×8 chequerboard pattern.

It was found that the choice of the parameter μ in initialising $\underline{I}$ had an effect on the cost of solutions that were found. Figure 5.3 shows the cost discrepancy of the ANN solutions (averaged over 25 runs) applied to the concentric rings and chequerboard images under 20% distortion for different values of μ.

It is seen that lower-cost solutions are found as μ is increased. Thus the further

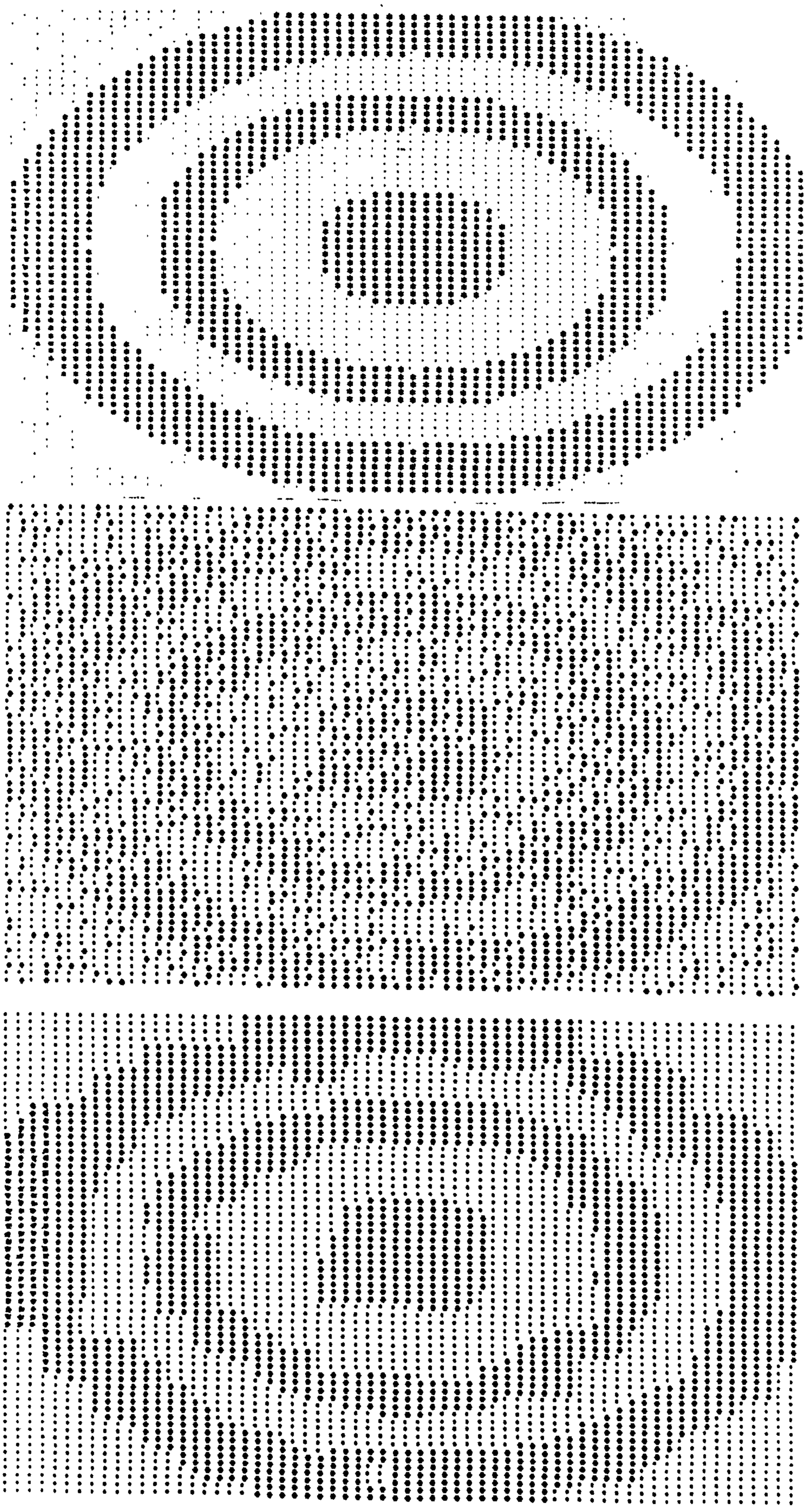

Figure 5.1: ANN restoration of an image with concentric rings

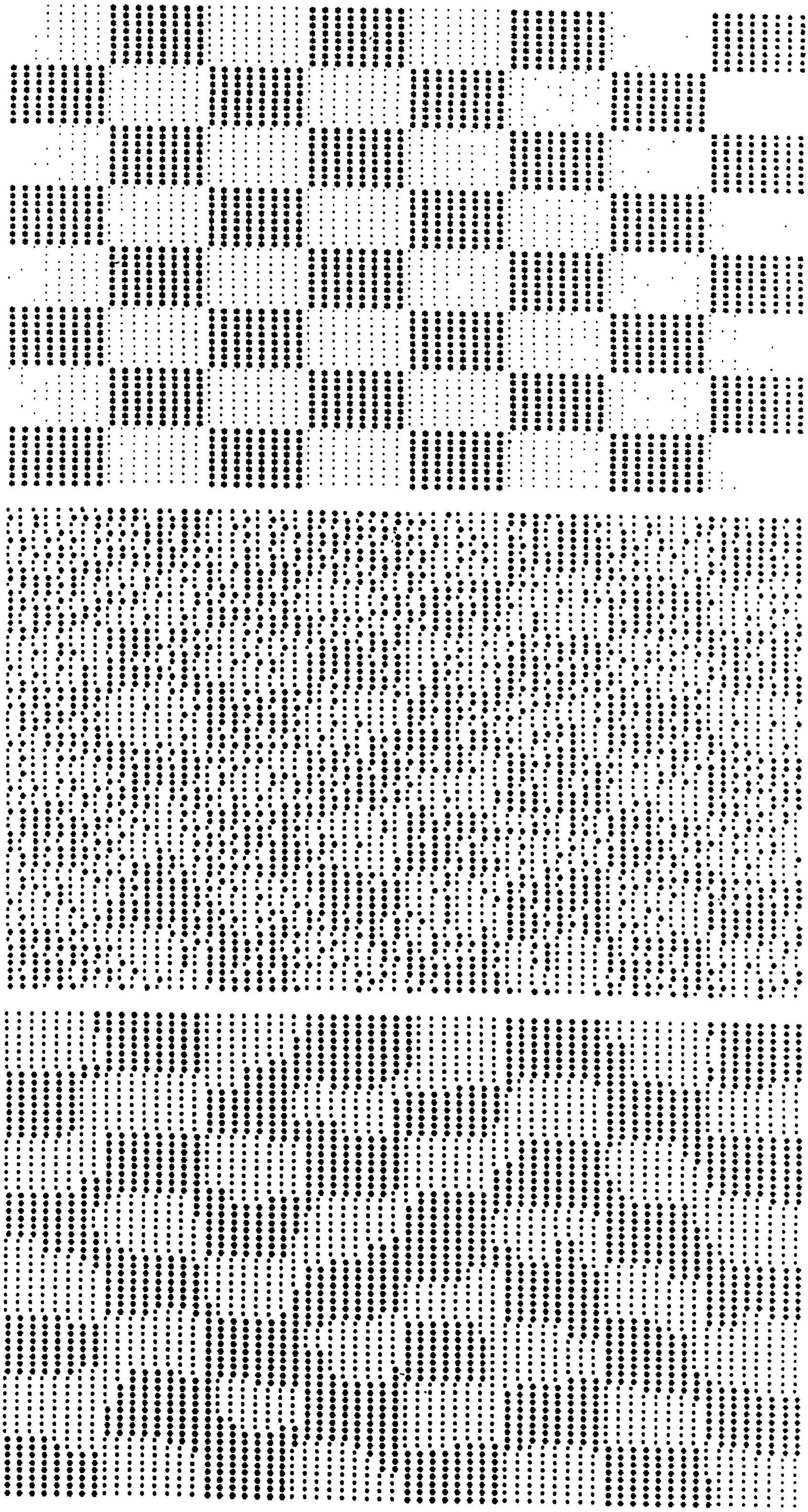

Figure 5.2: ANN restoration of an image of a chequerboard

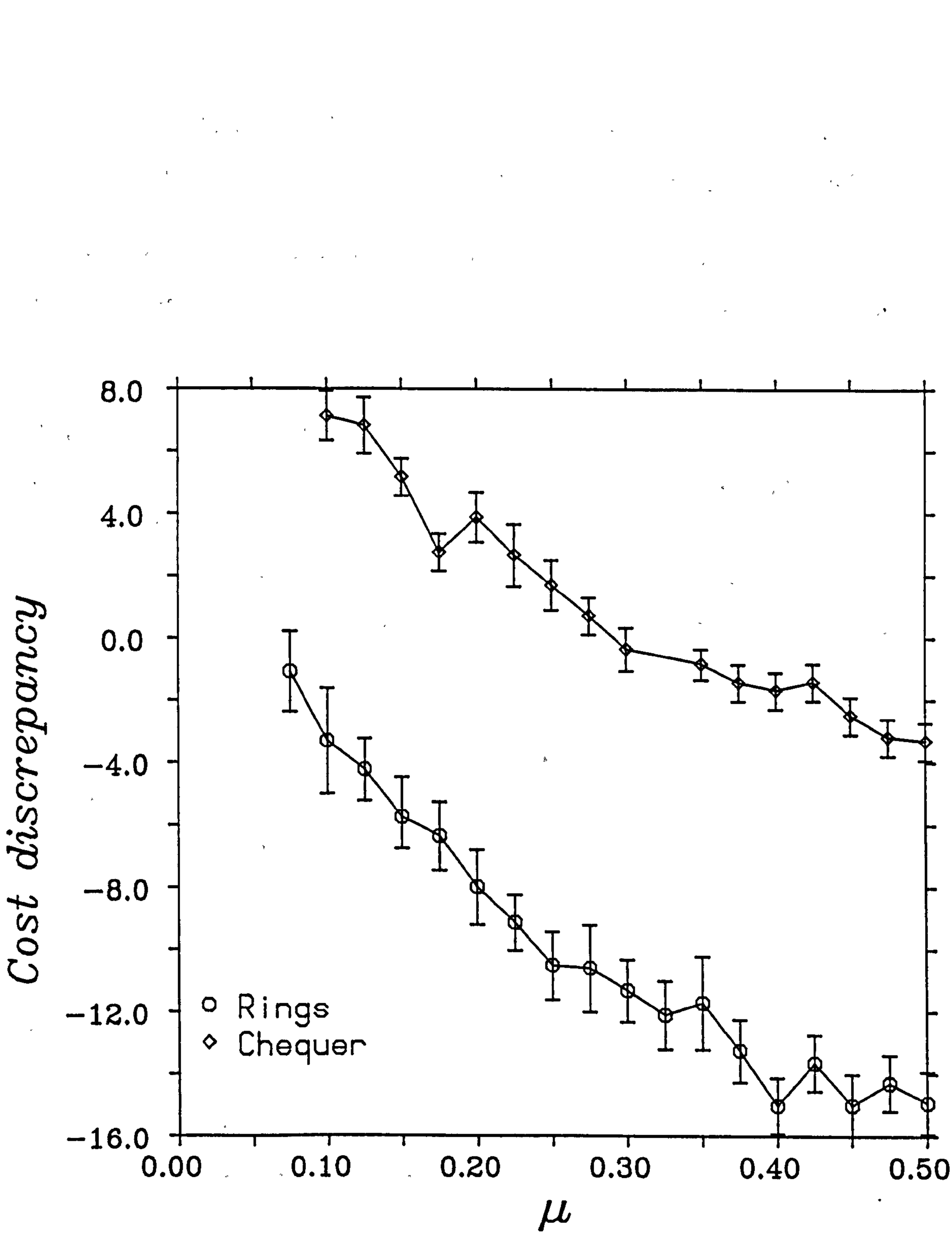

Figure 5.3: Effect of the parameter μ on the ANN minimisation

the initial configuration $\underline{I}^0$ is (away from the corner $\underline{D}$) towards the centre of the hypercube of phase-space, the deeper is the minimum that is found.

Increasing μ also tends to improve the performance of the ANN restoration in reducing the number of errors from the (20%) distorted images, as can be seen from figure 5.4. An optimal value of μ around 0.4 is suggested from this graph.

The quality of the ANN restoration was compared to that of a simple deterministic majority-rule (MR) scheme, where each pixel continually adopts the intensity of the majority of its neighbours—or remains unchanged if exactly half of them have the same intensity—until the image stabilises. Unlike the cost function approach, the MR method loses any memory of the observed data and only uses it as a starting point.

A third restoration method was also applied, that of performing a gradient descent (GD) on the cost function, where the variables $\underline{I}$ were restricted to be binary (0 or 1) at every iteration. This was achieved by updating each neuron according to

$$I_i = \begin{cases} 1 & \text{if } \Sigma_j T_{ij} I_j + \theta_i > 0 \\ 0 & \text{otherwise} \end{cases} , \tag{5.29}$$

where T_{ij} and θ_i are as in (5.25). This ensures that the cost of each successive state of the network decreases until a stable configuration is reached which is a (local) minimum of E. This corresponds to moving from corner to corner of the N-dimensional hypercube of phase-space until a local minimum of E is encountered. In fact, it is also equivalent to applying Besag's Iterated Conditional Modes (ICM) procedure (Besag 1986), which at each iteration chooses the intensity that has maximum conditional probability given the observed data and the current reconstruction elsewhere; that is, it chooses the mode of this conditional probability distribution. To demonstrate this equivalence, firstly recall that the posterior conditional probability of a restored image $\underline{I}$ given the data $\underline{D}$ can be

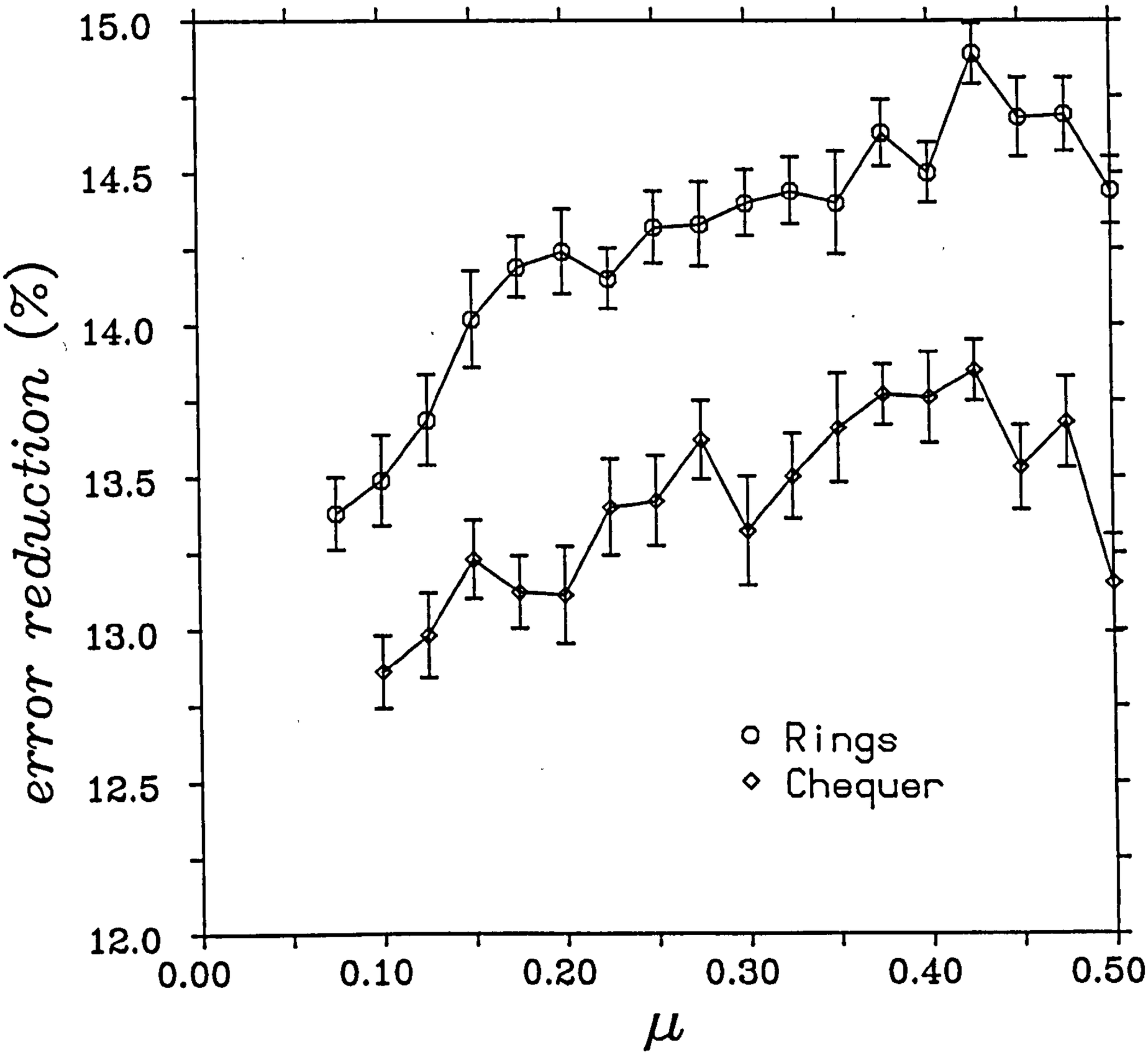

Figure 5.4: Effect of the parameter μ on the image restoration

expressed as

$$P(\underline{I}|\underline{D}) = \frac{1}{Z}e^{-E(\underline{I})}, \tag{5.30}$$

where

$$E(\underline{I}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}I_iI_j - \sum_{i=1}^{N}\theta_iI_i, \tag{5.31}$$

with T_{ij} and θ_i as given in (5.25). So the conditional probability that I_i assumes the value x_i given the data $\underline{D}$ and the current reconstruction elsewhere ($I_j = x_j, j \neq i$) is

$$P(I_i = x_i|\underline{D}\text{and } I_j = x_j, j \neq i) = \frac{1}{Z}\exp\left\{x_i\left(\sum_{j=1}^{N} T_{ij}x_j + \theta_i\right) + \xi\right\}, \tag{5.32}$$

where ξ is independent of x_i. Thus the choice of x_i which maximises this probability is

$$x_i = \begin{cases} 1 & \text{if } \Sigma_j T_{ij}I_j + \theta_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.33}$$

(since $x_i = 0$ or 1 only), which is equivalent to (5.29).

Figure 5.5 compares the success of the three methods in restoring distorted versions of the concentric rings image for various noise levels p by measuring the percentage reduction in errors achieved by the restorations. Each result was obtained by averaging over 25 simulations.

The GD method performs slightly better that the MR method, but the ANN restoration is significantly superior to both of these.

The cost discrepancy was also measured from these simulations and it is clear from figure 5.6 that the ANN method consistently finds lower-cost solutions that the corresponding GD solutions.

A direct visual comparison of the typical restorations obtained by the three methods is provided in figure 5.7. An original image (top left) has undergone 30% noise distortion (top right). The bottom three images are the restorations: ANN (left), GD (centre) and MR (right).

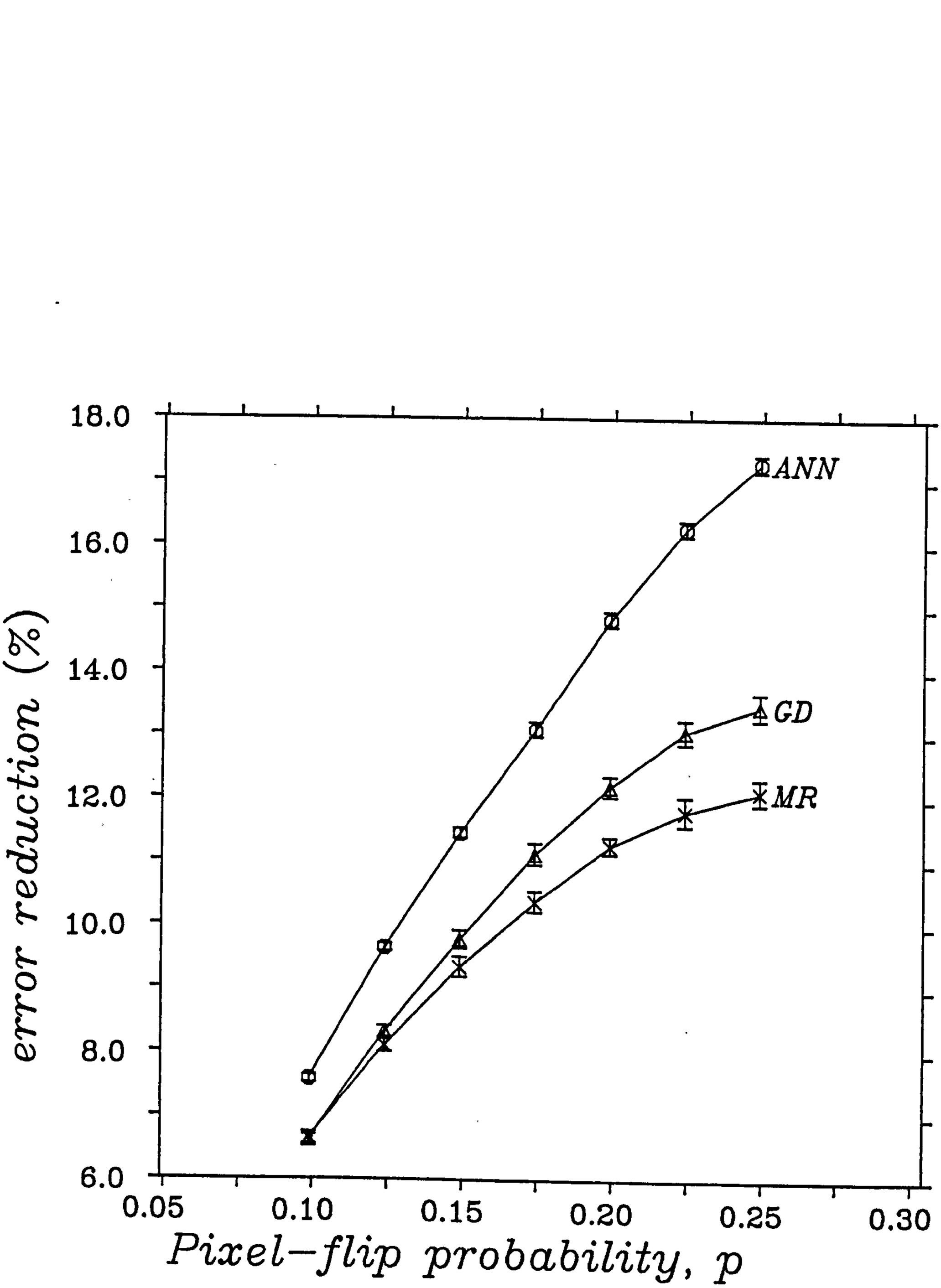

Figure 5.5: Comparison of the ANN, GD and MR restorations

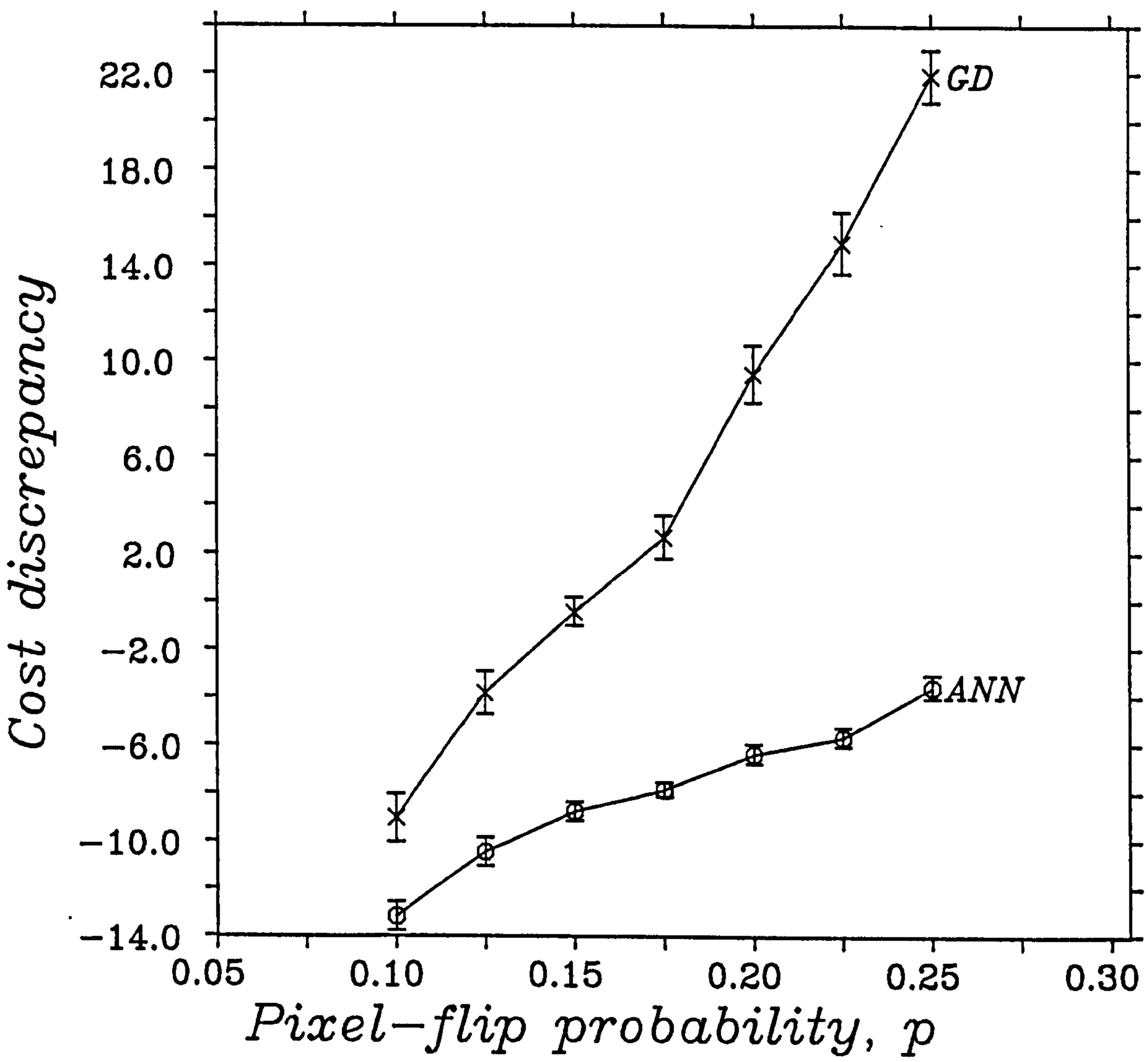

Figure 5.6: Comparison of the ANN and GD minimisations

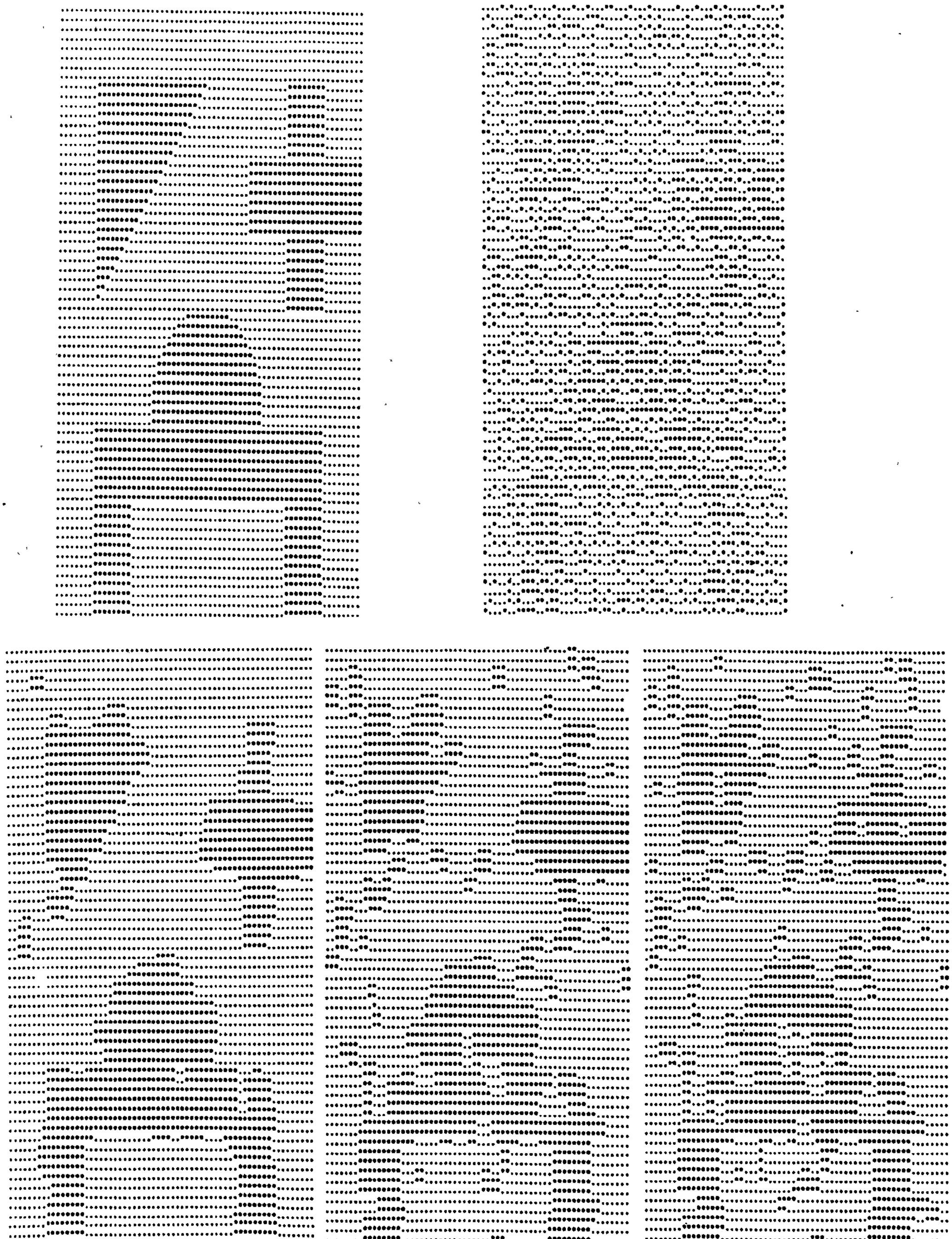

Figure 5.7: ANN, GD and MR restorations of a 30% corrupted image

5.3.4 Comparison with Simulated Annealing

Purely in terms of a minimisation tool, the ANN network has been shown in § 5.3.3 to be superior to a gradient descent on the cost function (5.24). To further gauge its performance, comparison was made with a renowned powerful stochastic method for minimising combinatorial cost functions: *Simulated Annealing* (SA) (Kirkpatrick *et al.* 1983).

Unlike gradient methods, which have no way of escaping from local minima, SA allows uphill moves (with respect to the cost function), but in a controlled manner. It is based on the algorithm of Metropolis *et al.* (1953) for generating states of a system in equilibrium at a 'temperature' T; that is, it generates configurations c of a system distributed with a probability according to Gibb's law:

$$\text{Prob}(c) = Z^{-1} e^{-E(c)/T} \tag{5.34}$$

where $E(c)$ is the cost (or 'energy') of the configuration c. The larger the value of T, the more likely that uphill moves are accepted, while $T = 0$ essentially corresponds to a gradient descent. The process begins by equilibrating at high T where many uphill moves are accepted and a coarse search of phase-space is effectively carried out. The temperature is gradually lowered according to some 'annealing schedule' which ideally should be slow enough in order that thermal equilibrium is maintained. As T is lowered, the sizes of energy (cost) barriers over which the system is able to climb decreases, until near $T = 0$ the system will (hopefully) be in a low-lying state.

The SA procedure can be roughly described as

1. Choose an inital configuration c and an initial temperature T_0.

2. Propose a configuration change c to c' and determine the corresponding change in cost it would incur: $\delta E \equiv E(c') - E(c)$.

3. If $\delta E \leq 0$, accept the change. If $\delta E > 0$, accept it with probability $\exp(-\delta E/T)$.

4. Repeat 2 and 3 until the average cost at temperature T reaches equilibrium.

5. Decrease the temperature according to the annealing schedule and repeat steps 2 through 4.

Geman and Geman (1984) have shown that if T is lowered according to

$$T = T_0 \frac{\ln 2}{\ln(k+1)},$$

where k denotes the iteration cycle, then the algorithm should converge to the minimal energy. Unfortunately, they found that their theoretical value of T_0 would necessitate a prohibitive amount of computer time, but however that a lower value of T_0 was satisfactory. Kirkpatrick *et al.* (1983) also achieved success using a quicker cooling schedule. In the simulations carried out here the temperature was reduced linearly, but by small amounts.

The initial configuration of step 1 was the observed image itself: $\underline{I}_0 = \underline{D}$. The proposed configuration change in step 2 simply consisted of choosing a pixel i and considering the change in cost δE_i if intensity I_i were flipped to $1 - I_i$:

$$\delta E_i = \left\{ 2 \sum_{j \in \mathcal{G}_i} A(2I_j - 1) + (2D_i - 1)\ln(p^{-1} - 1) \right\} (2I_i - 1). \qquad (5.35)$$

Once again, due to the local nature of this change, more than one pixel change could be proposed simultaneously.

Comparison of the SA and ANN algorithms in minimising the cost function (5.24) was determined by applying them to distorted versions of the chequer-board image in figure 5.2.

5.3.5 Numerical Results

Both algorithms were implemented on a Meiko Computing Surface (see Bowler *et al.* 1987), the principle component of which is the INMOS transputer—a VLSI chip which at present comes in two varieties. The T414 contains a 32-bit integer processor which can execute 10 million instructions per second (10MIP). The chip contains 2 Kbytes of memory and is capable of communication with other transputers via its four hardware links, each of which supports a band-width of 20 Megabits per second. In addition to these on-chip facilities, each transputer can also access up to 4 Gigabytes of off-chip memory at a rate of 25 Megabytes per second. The second type of transputer is the T800, which has an integrated 64-bit floating-point unit and 4 Kbytes of on-chip memory as well as all of the functionalities of the T414.

The Computing Surface is a reconfigurable array of transputers: the user can specify a particular connection architecture consistent with the maximum of four hardware communication links on each transputer.

The programming language of the transputers is the high-level language Occam, which provides for processes that can be executed concurrently (if they do not share any variables) and can communicate with each other through uni-directional 'soft' channels. An Occam program is distributed across the array of transputers, processes (one or more) being allocated to transputers and soft channels being assigned to 'hard' channels (physical links)—each hard link can implement two uni-directional soft channels.

This array of transputers is an example of a MIMD (Multiple Instruction stream, Multiple Data stream) machine: concurrently each transputer can process its own data—multiple data stream—but, unlike SIMD machines, where every processor executes identical instructions, every transputer can in principle be executing its own individual process on the data—multiple instruction stream.

The form of parallelism employed was that of *geometric parallelism* (i.e., processor to data locality was preserved). This was achieved by configuring T 'slave' transputers plus a 'master' transputer in a ring and assigning a band of $n = N/T$ pixel rows to each slave transputer, where the image contained N rows of pixels. (Then transputer 0 held rows 0 through $n-1$, transputer 1 held rows n through $2n-1,\ldots,$ transputer $T-1$ held rows $n(T-1)$ through $nT-1$.) In this way all the pixels needing to be accessed by any processor were either held on that processor or on one of its two neighbouring processors in the ring. Further details of the implementation are provided in Appendix D.

The results presented in figure 5.8 show the cost discrepancy of the solutions found by the SA algorithm from various initial temperatures T_0 applied to 20% and 25% distorted images. A linear annealing schedule was used, the temperature being lowered by $T_0/1000$ at each iteration (this being thought slow enough to maintain the image near thermal equilibrium).

Solutions of lower cost are found for higher starting temperatures. Comparison with the typical cost of ANN solutions would suggest that the SA algorithm is a more powerful minimisation technique, given a high enough T_0, but that the ANN is superior to SA from smaller values of T_0. Both are seen to be roughly equivalent (at least for this problem) around $T_0 = 0.3$.

However, the better minimisations of the SA solutions for increasing T_0 does not yield correspondingly better image restorations. On the contrary, much poorer results are produced—figure 5.9. (A negative value of the percentage

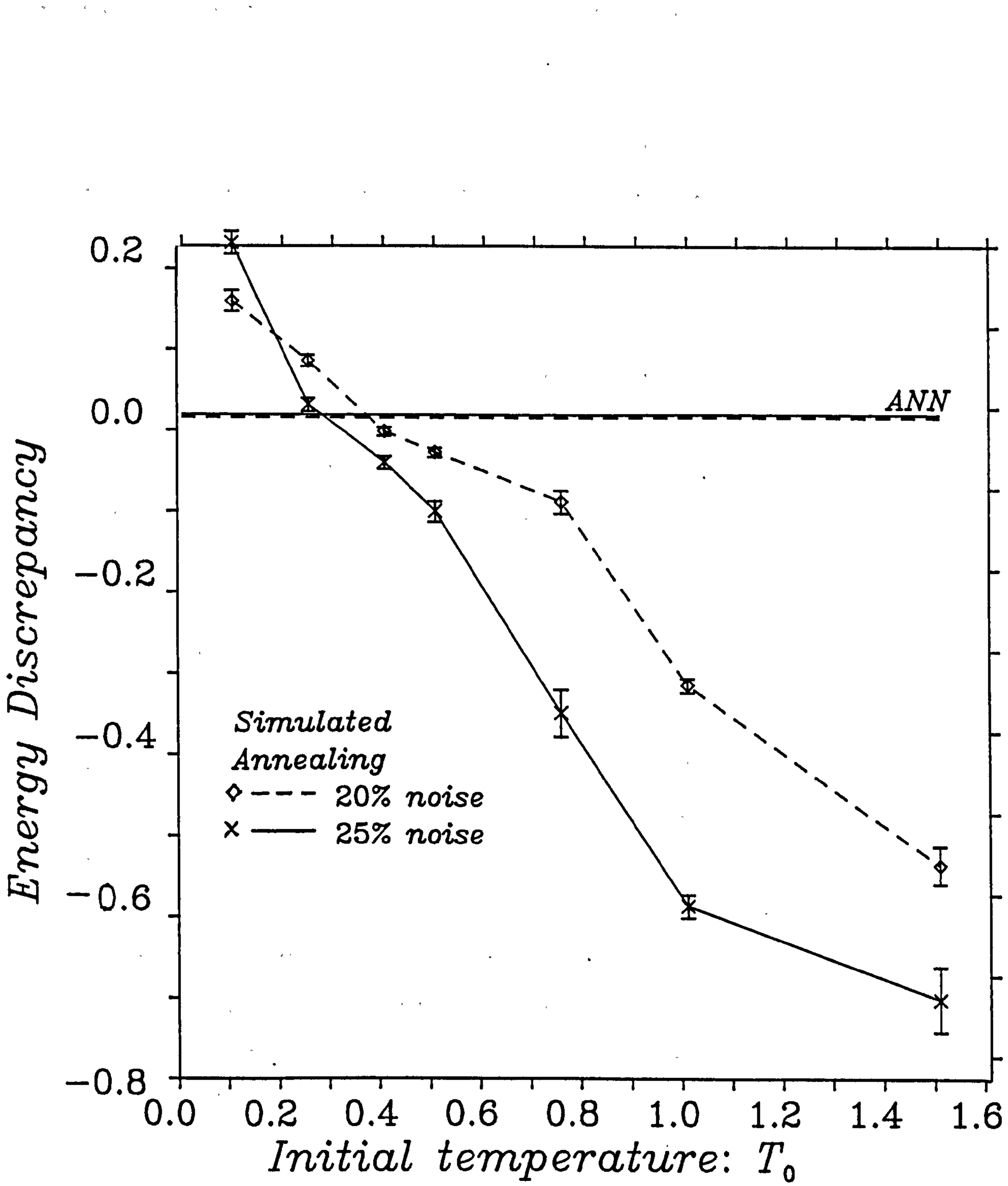

Figure 5.8: Comparison of SA minimisation with ANN

error reduction indicates that the restored image contains more errors than the noisy image.)

The reason for this is that the deep minima which SA finds for larger values of T_0 correspond to images which contain large regions of the same intensity. A typical example is shown in figure 5.10, which depicts the (25%) noisy chequerboard pattern and the SA restoration from $T_0 = 1.0$.

The fact that these minima of lower cost than the original image do not correspond to good restorations is a fault of the cost function itself. An ideal cost function would not have any minima of lower (or even comparable) cost than the original image. This is clearly not the case here.

5.4 Discussion

In terms of minimising the cost function (5.24), the ANN method has been shown to be superior than a gradient descent since it consistently finds solutions of significantly lower cost than the corresponding GD solutions. The results of § 5.3.4 indicate that the ANN minimisation is roughly as powerful as SA from temperatures around $T_0 = 0.3$. Annealing from higher temperatures found deeper minima than the ANN. Hence, although itself only a deterministic procedure, by modelling the discrete variables as analogue variables, the ANN can avoid minima of higher cost than found by a gradient descent of the discrete problem, and instead finds solutions of lower cost.

As for how good the corresponding restorations will be, depends on how well the global minimum (or at least the deepest minima) of the cost function faithfully represents the original image. The SA algorithm in § 5.3.4 was able to find minima lying deeper than the original image, but which were poor restorations.

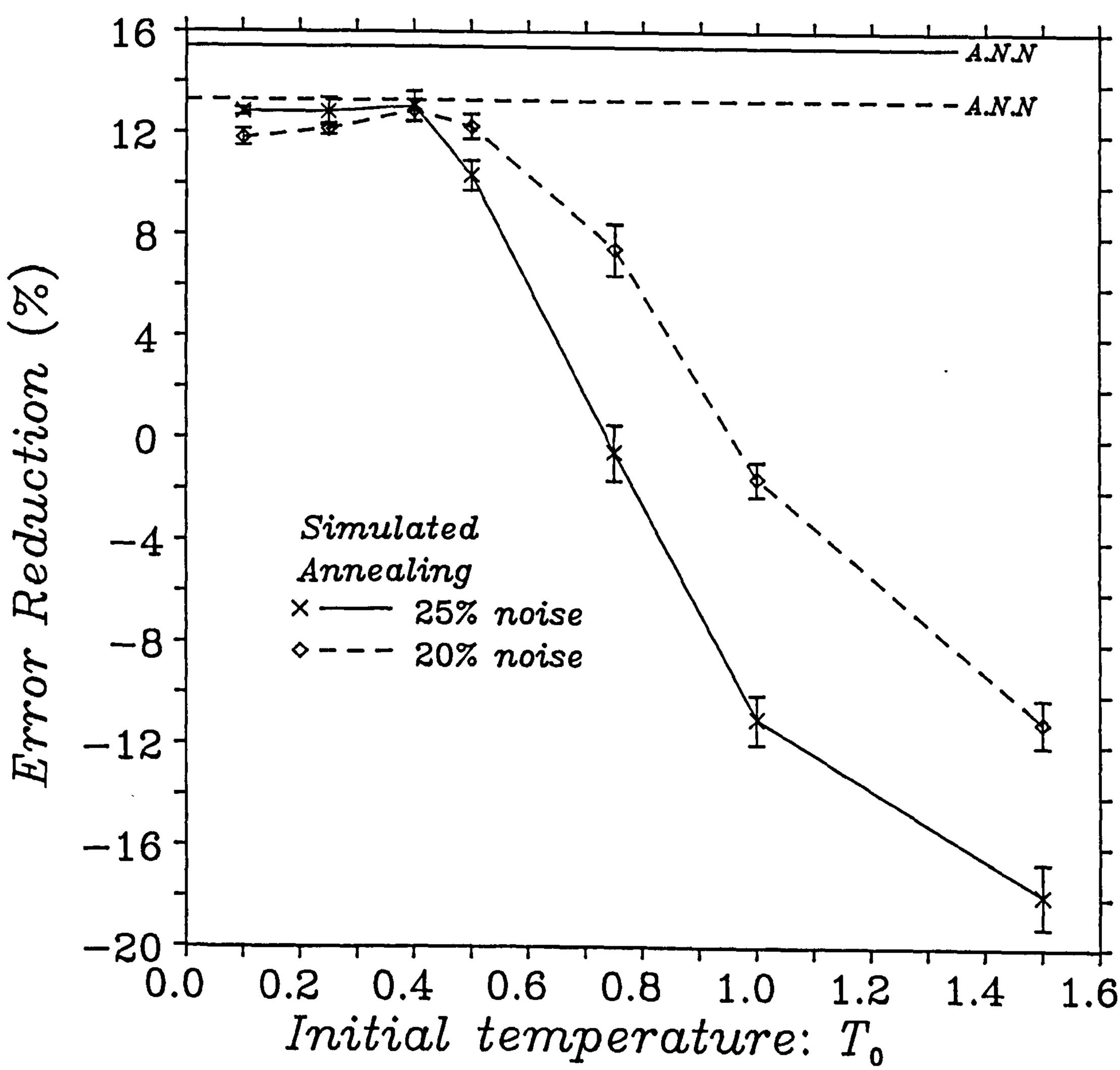

Figure 5.9: Comparison of SA restorations with ANN

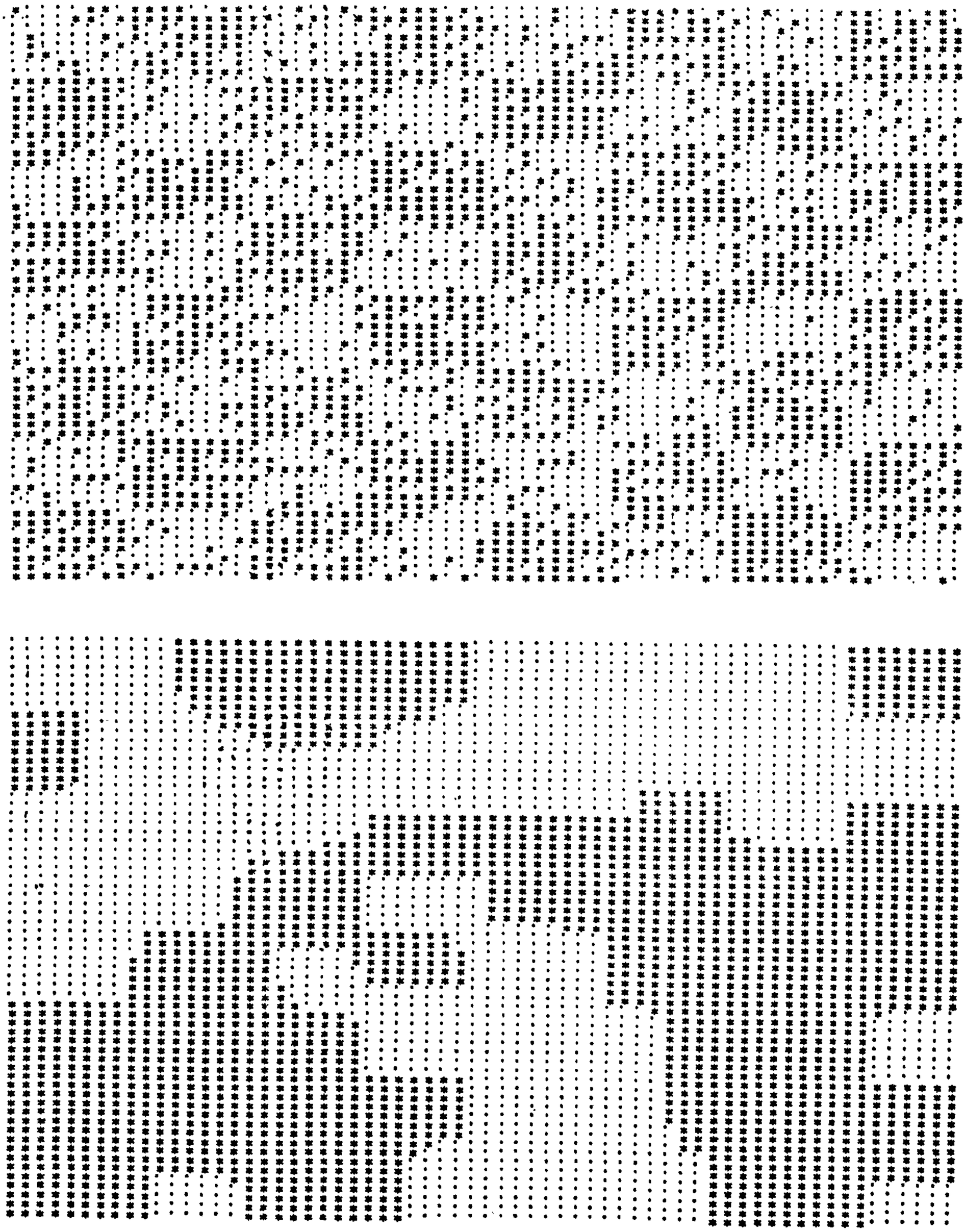

Figure 5.10: A typical SA image restoration ($T_0 = 1$)

However, the ANN solutions were of comparable cost to the original images and relatively good restorations. They were certainly of a higher quality than those achieved by a gradient descent on the same cost function and by the majority-rule method. Thus for good restorations it may be unnecessary to resort to highly ergodic stochastic searches such as SA from a high starting temperature, which only succeed in finding minima (although deep) which are far removed from the original image. As pointed out by Besag (1986), uncovering these really low-lying states can make the restorations susceptible to the undesirable large-scale effects of the prior distribution $P(\underline{I})$ which tend to produce images with large granularity. Also, when T_0 is high, many of the proposed pixel flips have a better chance of occuring (be they favourable or unfavourable with respect to reducing the cost function) and in doing so, the advantage of the good starting point—the observed image itself—is effectively lost.

The problem of the existence of deep minima which do not correspond to good interpretations could only be remedied by improvements in the underlying cost function. The fact that these 'bad' low-cost solutions tend to correspond to images containing large continuous patches would suggest that the cost-function might benefit from the inclusion of interstitial line processes (Geman and Geman 1984, Murray *et al.* 1986) which help favour discontinuities (edges) in the prior images. These are unobserved variables which reside between neighbouring pixels and when switched 'on' incur a cost, but also eliminate the interaction (5.21) between the two pixels. An extra term must be introduced in the cost function which is responsible for the interactions amongst these neighbouring line processes in order to encourage formation of the correct type of edges. Inclusion of these line processes, although binary variables themselves, would render the cost function incompatible with the quadratic form (5.1) required for the ANN method to be applicable. The method could be generalised to deal with a non-quadratic cost function $E(\underline{V})$ by noting that the equation of motion (5.5) for

the neural potentials is a special case of

$$\frac{du_i}{dt} = -\frac{u_i}{\tau} - \frac{\partial E}{\partial V_i} \tag{5.36}$$

and that $L(\underline{V})$ in (5.8) is still a Lyapunov function for the system. However, one of the potential merits (if not *the* potential merit) of the ANN is the ability to be implemented in hardware form: a requirement to which the original formulation (5.1) and (5.5) easily conforms, but this may not be the case for a general $E(\underline{V})$ in (5.36).

For restoration of grey-level images via the Geman and Geman algorithm to be amenable to an ANN, the cost function must still be one of binary variables. This could be accommodated if each grey-level pixel intensity I_i were represented by L bits $I_i^l (l = 0, 1, \ldots, L-1)$ so that 2^L grey levels would be allowed for:

$$I_i = \sum_{l=0}^{L-1} 2^l I_i^l. \tag{5.37}$$

To retain the same form of pairwise interactions (5.21), a product over the L bits could be used:

$$V_2(I_i, I_j) = -A \prod_{l=0}^{L-1} \left\{1 - (I_i^l - I_j^l)\right\} \equiv \begin{cases} -A & \text{for } I_i = I_j \\ 0 & \text{otherwise} \end{cases} \tag{5.38}$$

and the data term in the new cost function would be

$$\frac{1}{2\sigma^2(2^L - 1)} \sum_{i=1}^{N} \{D_i - I_i\}^2,$$

where the noise is Gaussian of zero mean and variance σ^2.

The cost function would no longer be quadratic in the variables I_i^l, so the general equation of motion (5.36) would have to be used:

$$\frac{du_i^l}{dt} = -\frac{u_i^l}{\tau} - \frac{\partial E}{\partial I_i^l} \quad \text{with} \quad I_i^l = \frac{1}{1 + e^{-gu_i^l}}. \tag{5.39}$$

A network of NL neurons would now be required to process an image of N pixels of grey-levels $0, 1, \ldots, 2^L - 1$. But, as mentioned above, such a network would

not have as straightforward a hardware implementation (if at all) as the one for binary images.

Regarding the speed with which the ANN restorations were accomplished, the simulations typically required from 1000 to 3000 sweeps to reach stability (i.e., $\delta I_i < 10^{-6}, \forall i$). This is certainly slower in computer time than the 10 or so sweeps required by the GD and MR methods (although not slow when compared to the SA algorithm). However, by virtue of the circuit implementation of the ANN that is being simulated here—through the differential equation (5.5)—the 'real' time to stability should be measured in terms of the characteristic response time τ of the neurons. Each updating cycle represented a time increment of $\delta t = 10^{-3}\tau$ in (5.5), as $\tau = 1$. Thus, the effective time for the ANN restorations was 1 to 3 time constants. This is of the same order of magnitude (a few time constants) as was found by Hopfield and Tank (1985) in their TSP ANN, and could therefore correspond to very rapid image restorations, given that the response times of electronic components could perhaps be of the order of microseconds or even less.

In contrast to the TSP network, this network for restoring binary images would be particularly straightforward to construct. From (5.25), connections would only have to be made between neighbouring amplifiers and these would all have the constant value of $8A$. (Recall that this would correspond to conductances of $8A$.) The TSP network, on the other hand, is fully connected, and some of the connections have values which depend upon the particular realisation of the problem under consideration—they involve the inter-city distances. However, here all the problem-dependent variables, namely the observed pixel intensities (D_i) and the estimate of the noise (p), appear only in the externally supplied bias currents θ_i in (5.25). This is more convenient as bias currents could be more readily altered than could conductances of hardware interconnects.

That this network does not encounter as many problems as its TSP counterpart

should not be too surprising. Firstly, the mapping from problem variables to neural variables is not as cumbersome: it is simply a one-one mapping from pixel intensity to neural activity. Hence the complexity of the problem is not increased, unlike in the TSP where each of the N cities is assigned N neurons. Secondly, the mapping also preserves the locality of the problem which arises out of the local interactions in the prior and from the conditional independence of the noise processes. In contrast, the TSP network is a fully connected one. Thirdly, there are fewer parameters to deal with in the associated cost function, and they are of a more intuitive nature: one of them (p) is the estimate of the amount of noise present and appears in the term which determines the faithfulness to the data; the other (A) determines how much continuity is encouraged across the restored image. The larger number of parameters and of terms which appear in the TSP network's cost function arises out of the necessity to enforce a permutation matrix solution—a valid tour—on the neural firing rates. Such a limitation of phase-space does not need to be imposed in the problem here. Fourthly, the TSP network suffers from its inability to distinguish amongst a ($2N$-fold) degeneracy in its solutions. Finally, a good starting point is available to the image restoration network: namely, the observed data itself.

The success of this method in achieving good image restorations is undoubtedly helped by the fact that it is not necessary to find the really low-lying states of the cost function in order to significantly enhance the corrupted images. Nonetheless, the ANN restorations were much better than those found by a deterministic search (ICM or GD) in the discrete variables and than those achieved by a majority-rule scheme. The ANN minimisation does not seem to be as powerful as simulated annealing (from a high enough temperature) but is nevertheless more powerful than a gradient descent and indeed was found to perform as well as simulated annealing from lower (but still positive) temperatures, despite being a deterministic procedure itself.

Although applied to a simpler problem than the TSP, the work of this last chap-

ter has demonstrated that a network of analogue neurons can be as powerful as some stochastic minimisations (simulated annealing from low temperatures—admittedly with a conservative cooling schedule). It can provide very rapid solutions—the network stabilises in a few neural time constants and this translates to a very fast solution on account of the huge parallelism involved as all the neurons update simultaneously.

Conclusions

It already had been shown that, despite involving extremely simplified representations of neurons, Ising spin neural network models such as the Hopfield model exhibit rich and complex behaviour similar to that of Ising spin glass models. The Hopfield model had also been shown to display critical behaviour with respect to the storage (stability) of patterns. Strong evidence has been provided in chapter 3 for critical behaviour and scaling of the basins of attraction in the model. Evidence for similar phenomena in the learning algorithm model of chapter 4 was also obtained (although smaller system sizes were studied there). Such scaling behaviour shows the importance of studying systems of increasing size in order to predict from any trend the expected behaviour of very large systems (the 'thermodynamic limit'). This consideration was particularly important in chapter 4: contrasting trends of the basins of attraction (and hence the content-addressability) for the naive ($M = 0$) and improved ($M > 0$) algorithms were found.

The learning algorithm was developed in chapter 4 by identifying the limitations of the naive algorithm (based upon reinforcement of unstable spins using the Hebbian rule) which, despite the promising results of ensuring perfect storage of all patterns at values of storage ratio α much higher than is possible in the Hopfield model, fails to install finite basins of attraction at these high values. The solutions for the synaptic connections found by the improved algorithm were shown to exhibit behaviour indicating that they will endow the network

with appreciable regions of attraction in the thermodynamic limit, in addition to providing perfect storage of the patterns. This was achieved for storage ratios as high as $\alpha = 0.5$—well above the limitation $\alpha_2^c \simeq 0.14$ of the Hopfield model. The solutions provided are also superior with respect to the more realistic measure of relative storage capacity—the number of bits of information stored to the number of bits of synaptic information required—as was detailed in the discussion in § 4.6.

Despite involving a search through a very high-dimensional space, the learning achieves its success by using an algorithm that is both deterministic and local—changes to a synapse depend only on the states of the two neurons which it connects. (Admittedly it also requires the calculation of the average modulus of the synaptic connections at each learning cycle, but this is confined to a row average of the connection matrix (4.13).)

In keeping with the tradition of the Ising spin models of chapters 2 and 3, the learning algorithm of chapter 4 maintained the symmetry of the synapses, ensuring the existence of a Hamiltonian (energy) function. This symmetry is not very plausible from a biological point of view, as the way in which one neuron influences another (as described in chapter 1) is very much a uni-directional process. Nevertheless, this symmetry constraint does not seem to limit the performance of the learning algorithm: the results obtained in § 4.5 were close to the optimal solution for general asymmetric connections.

This type of Ising spin model, then, has been shown capable of performing as a content-addressable memory and if implemented in hardware form could provide very fast retrieval of large amounts of data since the 'neurons' could all be changing their states asynchronously. Also, when presented with an input which is a distorted form of one of a large number ($N\alpha$) of stored patterns, assuming the distortion is not large enough that it lies outside the pattern's basin of attraction, by the very act of relaxing into the undistorted pattern, the

network is simultaneously 'considering' all the $N\alpha$ possibilities while 'finding' the desired pattern that most resembles the presented pattern, as opposed to performing some kind of sequential search through its memory.

The analogue neural network of chapter 5 has also been shown capable of recovering good interpretations of distorted binary images. While the model in chapter 4 could recover distorted images (having at least the critical minimum overlap with a stored pattern) without error, it could only do so for patterns which are explicitly stored in the 'memory'. The ANN technique does not rely on storing any patterns in order to restore them, but instead can in principle recover equally well any image which can be approximated by the prior distribution (Markov random field). In particular it could enhance (to the same degree of quality) both an image and a second image that was obtained from a global translation of the first image. However, the memory model of chapter 4 could only achieve this if both images were explicitly stored in it.

The ANN involved a neural network functioning as an optimisation tool. The results of chapter 5 have shown that, in addition to producing image restorations that compare favourably with other determinstic techniques (GD and MR), the ANN method is more powerful than a corresponding gradient descent in terms of minimising the cost function. In fact, the results of § 5.3.4 tend to suggest that, at least for this problem, it is as powerful as simulated annealing—a stochastic algorithm—up to starting temperatures around $T_0 \simeq 0.3$. The solutions found could perhaps correspond to better image restorations if a larger neighbourhood system or an improved cost function were used (although the latter may inhibit possible hardware implementation).

Undoubtedly the time (a few neural time constants) in which the ANN produced its solutions were very fast in view of the possible hardware implementation and the fact that a large degree of parallelism would be involved, with all of the neurons evolving asynchronously.

All of the models considered here have been non-stochastic in nature. The inclusion of noise into the neural dynamics may improve performance further: if, as with the Hopfield model in the FM phase ($\alpha < \alpha_1^c$), there are any spurious minima near the nominal patterns' regions of attraction, the introduction of noise could effect escape from these into the (larger) desired basin; a noise term in the ANN could perhaps improve its minimisation ability.

The work throughout this thesis has involved computationally intensive numerical simulations: the memory models of chapters 3 and 4 were fully connected networks; the one in chapter 5 had local interactions, but the neurons were of an analogue nature and required simulation of a differential equation. The heavy computational demand of simulating large networks of this type in a feasible amount of time was alleviated by exploiting the large degree of parallelism inherent in these models—a common feature amongst a wide range of neural network models. The simulations in chapters 3 and 4 involved both algorithmic and task parallelism, with the boolean neurons being particularly amenable to the bit-processing capabilities of the DAP. Those in chapter 5 had inherent geometric parallelism over a two-dimensional array: suited to both the DAP and to the flexible array of transputers constituting the Computing Surface.

The study of many neural network models and the realisation of their enormous potential in, e.g., optical design (Farhat *et al.* 1986) and silicon implementation (Sivilotti *et al.* 1986) will rely not only on innovation and analysis, but also on their simulation, for which parallel computing should continue to be an invaluable tool.

Appendix A

DAP Implementation of the Hopfield Model

Systems of sizes $N = 512, 1024$ and 2048 were simulated. Since when updating the ith spin only the ith row of the connection matrix is required, to save memory space in the DAP, it was decided to calculate the corresponding row each time it was required instead of storing the whole connection matrix. This was possible due to the (Hebbian) prescription for the T_{ij} which essentially means that each connection is just a sum of p conditional adds of +1 or −1.

For simplicity, only the case of $N = 1024$ will be dealt with here. The similar way in which the other system sizes were implemented should then become clear.

The p nominal patterns were stored in logical matrices thus:

$$V(,,r) = \left[\begin{array}{cccc|c|c|c} {}^{(1)}V_1^r & {}^{(1)}V_{65}^r & \cdots & {}^{(1)}V_{961}^r & & & \\ {}^{(1)}V_2^r & {}^{(1)}V_{66}^r & \cdots & {}^{(1)}V_{962}^r & \text{sim.} & \text{sim.} & \text{sim.} \\ \vdots & \vdots & & \vdots & (2) & (3) & (4) \\ {}^{(1)}V_{64}^r & {}^{(1)}V_{128}^r & \cdots & {}^{(1)}V_{1024}^r & & & \end{array}\right]$$

where ${}^{(s)}V_i^r$ denotes the ith spin of the rth nominal pattern in simulation s. (The number of simultaneous simulations was $4096/N$). ${}^{(s)}V_i^r = 1$ (.TRUE.) or 0 (.FALSE.) denotes ${}^{(s)}S_i^r = 1$ or -1 respectively.

For updating the kth spin, the kth row of synaptic connections was also held in

a matrix (temporarily):

$$Tk(,) = \left[\begin{array}{cccc|c|c|c} {}^{(1)}T_{k,1} & {}^{(1)}T_{k,65} & \dots & {}^{(1)}T_{k,961} & & & \\ {}^{(1)}T_{k,2} & {}^{(1)}T_{k,66} & \dots & {}^{(1)}T_{k,962} & \text{sim.} & \text{sim.} & \text{sim.} \\ \vdots & \vdots & & \vdots & (2) & (3) & (4) \\ {}^{(1)}T_{k,64} & {}^{(1)}T_{k,128} & \dots & {}^{(1)}T_{k,1024} & & & \end{array}\right]$$

where ${}^{(s)}T_{k,j}$ represents the (k,j)th element in the sth simulation.

The DAP-FORTRAN code for constructing this matrix was as follows.

```
c  no. of simulations  NSIMS = 4096/N
c  no. of columns needed per simulation  NCOLS = N/64
       LOGICAL LTEMP1(,), LTEMP2(,), LMASK(,), V(,,512)
       DO 1000 r = 1, p
         LTEMP2(,) = V(,,r)
         LMASK(,) = COLS(1,NCOLS)
c (a matrix whose first NCOLS columns are .TRUE. and
c  the remainder are .FALSE.)
         M = 0
         DO 1001 ISIM = 1, NSIMS
         LTEMP1(LMASK) = MAT(LTEMP2(K+M))
         M = M + N
c  shift LMASK to the East (cyclically) by NCOLS columns
         LMASK = SHEC(LMASK,NCOLS)
1001
1000     LWORK(,,r) = LTEMP2.LEQ.LTEMP1
```

Then `LWORK(,,r)` contains

$$\left[\begin{array}{cccc|c} {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_1^r & {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_{65}^r & \dots & {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_{961}^r & \text{etc.} \\ {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_2^r & {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_{66}^r & \dots & {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_{962}^r & \text{for} \\ \vdots & \vdots & & \vdots & \text{sims.} \\ {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_{64}^r & {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_{128}^r & \dots & {}^{(1)}V_k^r\text{.LEQ.}{}^{(1)}V_{1024}^r & (2),(3),(4) \end{array}\right]$$

where .LEQ. means "logically equal to".

At this stage the APAL (DAP machine code) subroutine A03ADDPLANES was used: `ISUM(,) = A03ADDPLANES(LWORK,p)` , which executes ISUM(i,j) = $\sum_{r=1}^{p}$ LWORK(i,j,r) simultaneously on each processor (i,j), where each logical

variable LWORK(i,j,r) is assigned the integral value 0 (1) if it is .FALSE. (.TRUE.).

Then `Tk(,) = 2 * ISUM(,) - 1` (where the multiplication by 2 was achieved by an `EQUIVALENCE`) produces the desired result since

$$2 * \left\{ \sum_{r=1}^{p} {}^{(*)}V_k^r \texttt{.LEQ.} {}^{(*)}V_i^r \right\} - 1 \equiv \sum_{r=1}^{p} \left(2^{(*)}V_k^r - 1 \right) \left(2^{(*)}V_k^r - 1 \right)$$
$$\equiv \sum_{r=1}^{p} {}^{(*)}S_k^r {}^{(*)}S_i^r =^{(*)} T_{ki}$$

However, the diagonal elements of T_{ij} must be set to zero: `Tk(TDIAG) = 0`, where `TDIAG` is a mask whose .TRUE. elements denote the diagonal elements T_{kk}.

For updating the kth node in the iteration of of the configuration `V(,)`, the local field $h_k \equiv \sum_j T_{kj} S_j$ is required. This was achieved as follows. Firstly `TS(,) = MERGE( Tk, -Tk, V)` produces

$$\texttt{TS(,)} = \left[\begin{array}{cccc|c} {}^{(1)}T_{k,1}{}^{(1)}S_1 & {}^{(1)}T_{k,65}{}^{(1)}S_{65} & \ldots & {}^{(1)}T_{k,961}{}^{(1)}S_{961} & \text{etc.} \\ {}^{(1)}T_{k,2}{}^{(1)}S_2 & {}^{(1)}T_{k,66}{}^{(1)}S_{66} & \ldots & {}^{(1)}T_{k,962}{}^{(1)}S_{962} & \text{for} \\ \vdots & \vdots & & \vdots & \text{sims.} \\ {}^{(1)}T_{k,64}{}^{(1)}S_{64} & {}^{(1)}T_{k,128}{}^{(1)}S_{128} & \ldots & {}^{(1)}T_{k,N}{}^{(1)}S_N & (2),(3),(4) \end{array} \right]$$

where, in `V(,)`, V_i = .TRUE.(.FALSE.) denotes $S_i = 1(-1)$.

Then the DAP-vector `H() = SUMR(TS(,))` contains

$$\left[\sum_{i=1}^{64} {}^{(1)}T_{k,i}{}^{(1)}S_i \quad \sum_{i=65}^{128} {}^{(1)}T_{k,i}{}^{(1)}S_i \quad \ldots \quad \sum_{i=961}^{N} {}^{(1)}T_{k,i}{}^{(1)}S_i \;\middle|\; \text{etc. for (2),(3),(4)} \right]$$

The cascading summation

```
      J = 1
      DO 2000 I=1, LG2NCOLS
c  LG2NCOLS = log2(NCOLS)
        H() = H() + SHLP(H,J)
2000    J = J + J
```

produces

$$\mathrm{H}() = \left[\sum_{i=1}^{N} {}^{(1)}T_{k,i}{}^{(1)}S_i \quad * \quad \ldots \quad * \quad | \quad \text{etc. for (2),(3),(4)}\right]$$

where * denotes irrelevant entries. A matrix HMAT(,) containing the relevant local field terms was then produced from this:

```
      LMASK(,) = COL1(,)
c COL1(,) is a logical matrix whose .TRUE. values denote the first
c column of each simulation s.
      HMAT(,) = MATR(H())
      I = 1
      DO 3000 J=1, LG2NCOLS
      HMAT(,) = MERGE(HMAT, SHEP(HMAT,I), LMASK)
      LMASK(,) = LMASK(,).OR.SHEP(LMASK,I)
3000  I = I + 1
```

Each row of HMAT(,) then contains

$$\left[{}^{(1)}h_k \quad {}^{(1)}h_k \quad \ldots \quad {}^{(1)}h_k \quad | \quad {}^{(2)}h_k \quad {}^{(2)}h_k \quad \ldots \quad {}^{(2)}h_k \quad |(3),(4)\right]$$

Then V(TDIAG) = HMAT(,).GT.0 will update the kth node of the iterated pattern V(,) (at each simulation in parallel).

More than 3×10^8 conditional adds were achieved per second for $N = 2048, \alpha = 0.10$, while the number of single-spin updates per second exceeded 1700 for $N = 512$.

Appendix B

Learning algorithms: DAP Implementation

Only the case of $N = 256$ will be described for simplicity ($N = 64, N = 128$ and $N = 512$ were also possible in the program). The synaptic connections were held in 64 DAP-matrices:

$$T(,,i) = \left[{}^{(1)}\theta_0 \quad {}^{(1)}\theta_1 \quad {}^{(1)}\theta_2 \quad {}^{(1)}\theta_3 \quad | {}^{(2)}\theta_0 \ldots {}^{(2)}\theta_3 \quad | \text{etc. for (3),(4)} \right]^t$$

$(i = 1, 2, \ldots, 64)$ where the 4×64 submatrices are

$${}^{(s)}\theta_n \equiv \begin{bmatrix} {}^{(s)}T_{1,i+64n} & {}^{(s)}T_{2,i+64n} & \cdots & {}^{(s)}T_{64,i+64n} \\ \vdots & \vdots & & \vdots \\ {}^{(s)}T_{193,i+64n} & {}^{(s)}T_{194,i+64n} & \cdots & {}^{(s)}T_{256,i+64n} \end{bmatrix}$$

Four nominal states were held in each matrix `S(,,r)`:

$$S(,,r) = \left[{}^{(1)}\underline{S}^{4r-3} \quad {}^{(1)}\underline{S}^{4r-2} \quad {}^{(1)}\underline{S}^{4r-1} \quad {}^{(1)}\underline{S}^{4r} \quad | \text{etc. for (2),(3),(4)} \right]^t$$

with the nominal state

$${}^{(s)}\underline{S}^x \equiv \begin{bmatrix} {}^{(s)}S_1^x & {}^{(s)}S_2^x & \cdots & {}^{(s)}S_{64}^x \\ \vdots & \vdots & & \vdots \\ {}^{(s)}S_{193}^x & {}^{(s)}S_{194}^x & \cdots & {}^{(s)}S_{256}^x \end{bmatrix}$$

with the same notation for ${}^{(s)}T_{ij}$ and ${}^{(s)}S_i^x$ as in appendix A.

Then the stability of the kth spin of 16 vectors could be tested simultaneously:

```
c Notation:
c           Number of rows per nominal state NROWS = N/64
c           NROWSQ = NROWS ** 2
          ITESTM(,) = MERGE(T(,,k),-T(,,k),S(,,r))
          ITEMP(,) = ITESTM(,)
          DO 1000 J=1,NROWS-1
            ITEMP(,) = MERGE(SHNC(ITEMP,NROWS-1),SHSC(ITEMP,1),LR1V)
c where the .TRUE. elements of LR1V(,) denote the first row of
c each state
1000        ITESTM(,) = ITESTM(,) + ITEMP(,)
          H() = SUMC(ITESTM)
```

Then H() contains

$$[{}^{(1)}h_k \quad {}^{(1)}h_k \quad {}^{(1)}h_k \quad {}^{(1)}h_k \quad | \quad {}^{(1)}h_{k+64} \ldots {}^{(1)}h_{k+64} \quad |$$
$${}^{(1)}h_{k+128} \ldots {}^{(1)}h_{k+128} \quad | \quad {}^{(1)}h_{k+192} \ldots {}^{(1)}h_{k+192} \quad ||$$
$$\text{etc. for (2),(3),(4)}]^t$$

This then allowed the 16 error masks ${}^{(s)}\epsilon_k^{4r-3}, {}^{(s)}\epsilon_{k+64}^{4r-2}, {}^{(s)}\epsilon_{k+128}^{4r-1}$ and ${}^{(s)}\epsilon_{k+192}^{4r}(s = 1,2,3,4)$ to be determined synchronously. These error masks were held in a logical matrix in a similar fashion to the nominal states S(,,r):

$$E(,,r) = \left[{}^{(1)}\underline{\epsilon}^{4r-3} \quad {}^{(1)}\underline{\epsilon}^{4r-2} \quad {}^{(1)}\underline{\epsilon}^{4r-1} \quad {}^{(1)}\underline{\epsilon}^{4r} \quad |\text{etc. for (2),(3),(4)}\right]^t$$

The above was repeated over the loop $k = 1$ to 64, which then meant that one quarter of all of the error masks for the nominal states $4r - 3$ to $4r$ had been determined. To obtain the remaining three quarters, the above code was executed again, after the nominal states and their associated error masks had been 'rotated':

```
S(,,r) = MERGE(SHNC(S(,,r),NROWSQ-NROWS),
                    SHSC(S(,,r),NROWS),RMASK)
E(,) = MERGE(SHNC(E,NROWSQ-NROWS), SHSC(E,NROWS), RMASK)
```

where RMASK(,) is a logical matrix with .TRUE. entries on the first NROWS rows

of each simulation. This then meant that

$$S(,,r) = \left[{}^{(1)}\underline{S}^{4r} \quad {}^{(1)}\underline{S}^{4r-3} \quad {}^{(1)}\underline{S}^{4r-2} \quad {}^{(1)}\underline{S}^{4r-1} \mid \quad \ldots \right]^t$$

and similarly for E(,). Hence ϵ_1^{4r} to ϵ_{64}^{4r}, ϵ_{65}^{4r-3} to ϵ_{128}^{4r-3}, etc. could all be determined as above. After the 'rotating' procedure was repeated twice more, all of the error masks for the four nominal states $4r-3$ to $4r$ had all been determined. The corresponding changes to the T_{ij} were accumulated in a matrix DT(,) until all of the nominal vectors had been accounted for, then the T(,,i) matrices were modified accordingly.

Up to 16000 single-spin updates per second were achieved in the simulations.

Appendix C

DAP Implementation of ANN image restoration

As described in §5.3.3, binary images of 64×64 pixels were restored, each pixel being assigned to one DAP processing element. The neighbourhood pixels of $\mathcal{G}_i$ consisted of the nearest neighbours of pixel i.

The notation used in the DAP code was as follows.

```
c LOGICAL T(,): target (original) image
c LOGICAL D(,): corrupted image (data)
c REAL U(,), I(,): input potentials and firing rates of neurons
c INTEGER NBHD(,): NBHD(i) = number of pixels in neighbourhood
c                  of pixel i
c LOGICAL CHEQUER(,): chequerboard mask; .TRUE. values denote
c                     those neurons currently being updated
```

First of all, the logical mask `CHEQUER(,)`, the neighbourhood matrix `NBHD(,)` and the matrix `THETA(,)` were initialised:

```
 CHEQUER(,) = (ALTR(1).OR.ALTC(1)).AND.
*                  (.NOT.(ALTR(1).AND.ALTC(1)))
 NBHD(,) = 1
 NBHD = NBHD + SHNP(NBHD,1) + SHSP(NBHD,1) +
*          SHEP(NBHD,1) + SHWP(NBHD,1) - 1
 THETA(,) = -4.*A*FLOAT(NBHD) + MERGE(LOGP,-LOGP,D)
```

where LOGP $\equiv \ln(p^{-1}-1)$. NBHD(i) contained $n(\mathcal{G}_i)$, the number of pixels in the neighbourhood of the ith pixel, so THETA(i) held $\theta_i = -4An(\mathcal{G}_i) + (2D_i - 1)\ln(p^{-1}-1)$.

The DAP-FORTRAN code for sweeping through the network was

```
      DO 1 ISWP=1,MAXSWEEPS
        DO 2 IP=1,2
c determine sum of neighbouring intensities
            SUMM(,) = 0.0
            SUMM(CHEQUER) = SHSP(I,1) + SHNP(I,1) +
     *                      SHWP(I,1) + SHEP(I,1)
            SUMM = 8.0 * A * SUMM
c increment potentials U (with TAU=1 here)
            DELTAU(,) = SUMM + THETA
            U(CHEQUER) = U + (DELTAU * DT)
            POTENTIAL(,) = -GAIN * U
c avoid overflow in exponential
            POTENTIAL(POTENTIAL.GT.173.0) = 173.0
            INEW(,) = 1./(1. + EXP(POTENTIAL))
            STABLE(CHEQUER) = ABS(I-INEW).LT.TOLERANCE
            I(CHEQUER) = INEW
c flip CHEQUER mask to deal with other neurons
            CHEQUER = .NOT.CHEQUER
2           CONTINUE
          IF (ALL(STABLE)) GOTO 3
1         CONTINUE
3       NSWEEPS = ISWP
```

DT $\equiv \delta t$, the time increment of the differential equation. The mask STABLE(i) was .TRUE. if and only if neuron i was stable to within tolerance TOLERANCE= 10^{-6}.

More than 100 sweeps through then network were achieved per second.

Appendix D

ANN image restoration on a Meiko Computing Surface

$T+1$ transputers were configured in a ring: a 'master' transputer and T 'slaves'. For the processing of $nT \times m$ images, each slave was allocated a band consisting of n rows of pixels (neurons), with m pixels in each row: slave 0 held rows 0 to $n-1$; slave 1 held rows n to $2n-1$; ...; slave r held rows rn to $(r+1)n-1$; ...; and slave $T-1$ held rows $(T-1)n$ to $nT-1$. The neuron firing rates and potentials were then held locally by each slave in the $n \times m$ arrays `[n][m]REAL32 I` and [n][m]REAL32 U, while the observed data was split up by the master and distributed around the ring to each slave, which similarly held their own n rows locally in `[n][m]INT D`.

Two input and two output channels were defined at each slave `r`:

```
CHAN cw.in -- clockwise input channel from slave r-1
CHAN cw.out -- clockwise output channel to slave r+1
CHAN cc.in -- counterclockwise input channel from slave r+1
CHAN cc.out -- counterlockwise output channel to slave r-1
```

The updating sweep by each slave on its own band of neurons was

```
PROC Update( CHAN cw.in, cw.out, cc.in, cc.out, [][]REAL32 I, u,
             INT sweeps, BOOL stable, VAL [][]INT D )
  [m]REAL32 top.row, bottom.row, top.nbrs.row, bottom.nbrs.row:
  SEQ
    stable := TRUE
    top.row := I[0]
    bottom.row := I[n-1]
    SEQ
      PAR
        ...evolve internal neurons
        ...communicate boundary neuron activites
      ...evolve boundary neurons
    sweeps := sweeps + 1
:
```

Hence the communication of boundary data from each slave to its two neighbours on the ring was overlapped with the updating of the neurons internal to a band (those in rows 1 to $n-2$). Thus the cost of splitting the image amongst the processors was reduced. The transfer of the boundary data between each slave's band of neurons was achieved by

```
SEQ
  IF
    id <> 0 -- if not slave 0 (top band of pixels)
      PAR
        cc.out ! top.row
        cw.in ? top.nbrs.row
    TRUE
      SKIP
  IF
    id <> (T-1) -- if not slave T-1 (bottom band)
      PAR
        cw.out ! bottom.row
        cc.in ? bottom.nbrs.row
    TRUE
      SKIP
```

Each slave held a local stability flag `stable`, which was TRUE only if all the neurons in its own band were all stable (to within a specified tolerance).

After every `out.freq` updates (a number which could be specified) the current state of all the neurons was passed around the ring to the master along with all the local stability flags: this allowed the image to be displayed and its stability to be determined.

The implementation of the simulated annealing algorithm in §5.3.4 involved an identical parallelization to the ANN procedure. Only the details of the `Update` procedure needed to be altered. (In addition, of course, only boolean variables `[][]BOOL I` were required for the pixel intensities.)

References

[Amit 1987] Amit D J 1987 *Heidelberg Symposium on Glassy Dynamics* ed I Morgenstern and J L van Hemmen (Berlin: Springer)

[Amit *et al.* 1985a] Amit D J, Gutfreund H and Sompolinsky H 1985 *Phys. Rev. Lett.* **55** 1530

[Amit *et al.* 1985b] Amit D J, Gutfreund H and Sompolinsky H 1985 *Phys. Rev. A* **32** 1007

[Amit *et al.* 1987a] Amit D J, Gutfreund H and Sompolinsky H 1987 *Ann. Phys., NY* **173** 30

[Amit *et al.* 1987b] Amit D J, Gutfreund H and Sompolinsky H 1987 *Phys. Rev. A* in press

[Besag 1972] Besag J E 1972 *J. Roy. Stat. Soc. B* **34** 75

[Besag 1986] Besag J E 1986 *J. Roy. Stat. Soc. B* **48** 259

[Bowler *et al.* 1987] Bowler K C, Kenway R D, Pawley G S and Roweth D 1987 *An Introduction to Occam 2 and the Meiko Computing Surface* Physics Dept., Univ. of Edinburgh

[Bruce *et al.* 1986] Bruce A D, Canning A, Forrest B, Gardner E and Wallace D J 1986 *Proc. Conf. on Neural Networks for Computing, Snowbird, UT (AIP Conf. Proc. 151)* ed J S Denker (New York: AIP) p65

[Bruce *et al.* 1987] Bruce A D, Gardner E and Wallace D J 1987 *J. Phys. A: Math. Gen.* **20** 2909

[Canning and Gardner 1988] Canning A and Gardner E 1988 *J. Phys. A: Math. Gen.* in press

[Cover 1965] Cover T M 1965 *IEEE Trans. Electron. Comput.* **EC-14** 326

[Diederich and Opper 1987] Diederich S and Opper M 1987 *Phys. Rev. Lett.* **58** 949

[Edwards and Anderson 1975] Edwards S F and Anderson P W 1975 *J. Phys. F* **5** 965

[Farhat *et al.* 1986] Farhat N H, Miyahara S and Lee K S 1986 *Proc. Conf. on Neural Networks for Computing, Snowbird, UT (AIP Conf. Proc. 151)* ed J S Denker (New York: AIP)

[Gardner 1986] Gardner E 1986 *J. Phys. A: Math. Gen.* **19** L1047

[Gardner 1987] Gardner E 1987 *Europhys. Lett.* **4** 481

[Gardner 1988] Gardner E 1988 *J. Phys. A: Math. Gen.* **21** 257

[Gardner *et al.* 1987a] Gardner E, Stroud N and Wallace D J 1987 *Preprint* Edinburgh 87/394

[Gardner *et al.* 1987b] Gardner E, Stroud N and Wallace D J 1987 *Preprint* Edinburgh 87/410

[Gardner *et al.* 1987c] Gardner E, Derrida B and Mottishaw P 1987 *J. Physique* **48** 741

[Geman and Geman 1984] Geman S and Geman D 1984 *IEEE Trans. PAMI* **5** 721

[Hebb 1949] Hebb D O 1949 *The Organisation of Behaviour* (New York: Wiley)

[Hopfield 1982] Hopfield J J 1982 *Proc. Natl. Acad. Sci. USA* **79** 2554

[Hopfield and Tank 1985] Hopfield J J and Tank D W 1985 *Biol. Cyber.* **52** 141

[Kirkpatrick *et al.* 1983] Kirkpatrick S, Gellat C D and Vecchi M P 1983 *Science* **220** 671

[Kirkpatrick and Sherrington 1978] Kirkpatrick S and Sherrington D 1978 *Phys. Rev. Lett.* **B17** 4384

[Krauth and Mézard 1987] Krauth W and Mézard M 1987 *J. Phys. A: Math. Gen.* **20** L745

[Le Cun 1985] Le Cun Y 1985 *Proc. Cognitiva 85, Paris*

[Little 1974] Little W A 1974 *Math. Biosc.* **19** 101

[Little and Shaw 1978] Little W A and Shaw G L 1978 *Math. Biosc.* **39** 281

[Mattis 1976] Mattis D C 1976 *Phys. Lett.* **56A** 421

[McCulloch and Pitts 1943] McCulloch W S and Pitts W A 1943 *Bull. Math. Biophys.* **5** 115

[Metropolis *et al.* 1953] Metropolis N, Rosenbluth A, Rosenbluth M, Teller A and Teller E 1953 *J. Chem. Phys.* **21** 1087

[Mézard *et al.* 1986] Mézard M, Nadal J P and Tolouse G 1986 *J. Physique* **47** 1457

[Minsky and Papert 1969] Minsky M L and Papert S 1969 *Perceptrons* (Cambridge, MA: MIT Press)

[Moore 1984] Moore M A 1984 *Statistical and Particle Physics: Common Problems and Techniques* eds K C Bowler and R D Kenway (SUSSP Publ.: Univ. of Edinburgh)

[Murray *et al.* 1986] Murray D W, Kashko A and Buxton H 1987 *Image and Vision Comp.* **4** 133

[Parisi 1980] Parisi G 1980 *J. Phys. A: Math. Gen.* **13** 1887

[Parisi 1986] Parisi G 1986 *J. Phys. A: Math. Gen.* **19** L617

[Parker 1985] Parker D B 1985 *MIT Sloan School of Management Technical Report TR-47*

[Pöppel and Krey 1987] Pöppel and Krey 1987 *Preprint* Regensburg

[Reddaway 1979] Reddaway S F 1979 *Infotech State of the Art Report: Supercomputers, vol. 2* eds C R Jesshope and R W Hockney p311

[Rumelhart *et al.* 1986] Rumelhart D E, Hinton G E and Williams R J 1986 *Nature* **323** 533

[Sherrington and Kirkpatrick 1975] Sherrington D and Kirkpatrick S 1975 *Phys. Rev. Lett.* **35** 1792

[Sivilotti *et al.* 1986] Sivilotti M A, Emerling M R and Mead C A 1986 *Proc. Conf. on Neural Networks for Computing, Snowbird, UT (AIP Conf. Proc. 151)* ed J S Denker (New York: AIP)

[Sompolinsky 1987] Sompolinsky H 1987 *Heidelberg Symposium on Glassy Dynamics* ed I Morgenstern and J L van Hemmen (Berlin: Springer)

[van Hemmen 1986] van Hemmen J L 1986 *Phys. Rev. A* **34** 3435

[Venkatesh 1986a] Venkatesh S 1986 *Proc. Conf. on Neural Networks for Computing, Snowbird, UT (AIP Conf. Proc. 151)* ed J S Denker (New York: AIP)

[Venkatesh 1986b] Venkatesh S 1986 *PhD thesis* California Institute of Technology

[Wallace 1986] Wallace D J 1986 *Lattice Gauge Theory—A Challenge to Large Scale Computing* ed B Bunk and K H Mutter (New York: Plenum)

[Wilson and Pawley 1987] Wilson G V and Pawley G S 1987 *Biol. Cyber.* in press