Microsimulation of Complex System Dynamics

Automata Models in Biology and Finance

Inaugural-Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von
Filippo Castiglione
aus Catania, Italien

Köln 2001

Berichterstatter: Prof. Dr. Rainer Schrader
Prof. Dr. Dietrich Stauffer

Tag der mündlichen Prüfung: 15.05.2001

“For the things we have to learn before we can do them, we learn by doing them.”
Aristotle (384 - 322 BC), Nichomachean Ethics

“First you guess. Don’t laugh, this is the most important step. Then you compute the consequences. Compare the consequences with experience. If it disagrees with experience, the guess is wrong. In this simple statement is the key to science.”
Richard P. Feynmann

“If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is.”
John L. von Neumann (1903 - 1957)

Quotations taken from [3]

Contents

Preface *1*

Introduction **5**

1 Computational immunology **13**
1.1 The model of Kaufman, Urbain and Thomas 14
1.2 The model of Weisbuch and Atlan 14
1.3 The model of Pandey and Stauffer 15
1.4 The bit-string model of Farmer, Packard and Perelson 16

2 The Celada-Seiden model **17**
2.1 The computational model 19
2.1.1 Entity/State description 19
2.1.2 The repertoire 21
2.1.3 The affinity potential 23
2.1.4 The interactions 23
2.1.5 The mutation process 26
2.1.6 Simulations 27
2.1.7 The parameters 27
2.1.8 The dynamics 28
2.1.9 Notes 33
2.2 Learning cascade 35
2.2.1 Collective dynamics in the immune system response . . . 35
2.3 The role of the affinity potential and hypermutation 42
2.3.1 Numerical simulations 44

3 Antigen recognition and evolution **51**
3.1 Introduction 52
3.2 The model 53
3.2.1 Discussion 57
3.3 Numerical integration 59
3.4 Summary and conclusions 60

4 Econophysics 69
4.1 Lévy stable distribution and fat tails 71
4.2 Stylized facts of financial time series 73

5 Microsimulation of stock markets 77
5.1 The Cont-Bouchaud herding model 78
5.1.1 Multi-affinity in financial price series 80

6 An agent – based model of stock market 87
6.1 Model description 88
6.1.1 Trading strategy 89
6.1.2 Collective formation and diffusion 90
6.2 Discussion 92
6.2.1 Chartist's influence on price dynamics 98
6.3 Hurst exponent and modified R/S statistics 101
6.4 Conclusions and future developments 103

7 Forecasting 105
7.1 Introduction 105
7.2 Multi-layer Perceptron 106
7.3 Detrending analysis 107
7.4 Determining the net topology 108
7.5 Stopping criteria 110
7.6 Results 111
7.7 Weekly and intra-day data 115
7.8 Artificially generated price series 116
7.9 Discussion 116

Summary 117

A Basic immunology 119
A.1 The humoral response 119
A.2 The cellular mediated response 121

B CImmSim: optimization and parallel programming 123
B.1 Dynamic memory allocation 123
B.1.1 List permutation 124
B.1.2 The single antigen case 125
B.2 State flags 125
B.3 Optimized Hamming distance computation 126
B.4 Compressed output 127
B.4.1 Saving detailed information 129

B.5 The parallel version of the simulator 129
B.6 Computing requirements 131
B.7 Timing 132
  B.7.1 Scaling the lattice size 132
  B.7.2 Scaling the bit-string length 133
  B.7.3 Scaling the initial number of cells 134
B.8 Porting to other platforms 135

C AMSE: parallel programming **137**
C.1 The parallel version of the simulator 137

D Glossary of Notation **141**

Bibliography **145**

Deutsche Zusammenfassung **157**

Acknowledgments **161**

Lebenslauf **167**

List of Figures

1 Computational Science 2
2 FHP lattice 7
3 Finite State Automata 9

2.1 Triangular lattice 19
2.2 Bit-string representation 22
2.3 B cell automaton 26
2.4 Population of virus and interferon-γ 29
2.5 B cell population dynamics 30
2.6 T cells population 31
2.7 APC population 32
2.8 T killer population 32
2.9 PLB population 33
2.10 IC population and D-signal 34
2.11 Total number of Ag and B-cells 38
2.12 B-cells distribution $f(m,t)$ 38
2.13 Time evolution of μ_n, $H(t)$ and $G(t)$ 40
2.14 Total affinity as a function of time 46
2.15 Affinity classes 49
2.16 Same data as in fig 2.15 for runs without mutation. 50

3.1 Fitness function or reproduction efficiency 54
3.2 Real part of the eigenvalues 57
3.3 The range $\mathcal{U}$ of figure 3.2 58
3.4 Asymptotic states of the model of eq(3.2) 61
3.5 (cont.) asymptotic states of the model of eq(3.2) 62
3.6 Error threshold 63
3.7 Error threshold p_h for different affinity increments $l - m_c$ 65

4.1 Comparing a random walk with S&P500 70
4.2 The market crash of 1987 71
4.3 Histogram of price change of the S&P500 index 73
4.4 Autocorrelation function of absolute net return of the DJI index . . 75

5.1 Histogram of price change for the Cont-Bouchaud model 81
5.2 Scaling exponent of DJI and Gaussian random walk 83
5.3 Scaling exponent for the CB-model synthetic returns 84
5.4 Scaling exponent for $L = 1001$ 85

6.1 Fundamentalists and chartists automata 90
6.2 Variation from the initial price 94
6.3 Wealth per agent-class 95
6.4 Profit per capita 96
6.5 Histogram of return 97
6.6 Comparing the log return of f_t and p_t 98
6.7 Autocorrelation function of volatility of return 99
6.8 Moving averages 100
6.9 Volatility histogram. Run with and without chartists 101
6.10 Histogram of volatility 102
6.11 MRS statistics 104

7.1 Learning, validation, check and test data sets 106
7.2 S&P500 detrended time series 108
7.3 A multi-layer perceptron 109
7.4 Forecast with MLP 111
7.5 Normalized $\mathcal{S}$ as function of Δp 112

A.1 Cell specific interactions 120

B.1 **CImmSim** communication pattern 130
B.2 **CImmSim**: CPU time 131
B.3 Scaling the lattice size 132
B.4 Scaling the bit string length 133
B.5 Scaling the initial number of cells 134

C.1 **AMSE**: communication scheme 138
C.2 **AMSE**: speedup 139

List of Tables

2.1 Cellular and molecular entities 20
2.2 Entity/State table 20
2.3 Cell interactions 22
2.4 Entity/Repertoire table 25
2.5 Affinity potential for convex and concave shapes 45

3.1 Error threshold as function of $l - m_c$ 64

7.1 Good/Bad neural net's ratio 113
7.2 Net topology 113
7.3 Success ratio for the prediction of the sign change 114
7.4 Success of a “random” prediction 115

B.1 State-determining flags 126

Preface

Since the advent of digital computers, the way research proceeds has dramatically changed. The study of physical systems for example, is traditionally investigated by means of mathematical models. These models are often very difficult to be solved analytically and approximations are necessary to reach a solution.

The use of computers has brought great advantages in handling complex models in two different although complementary ways: (1) the mathematical formulation of a model can be solved numerically using sophisticated *algorithms* to find a good “numerical” solution; (2) a system can be analysed in term of its constituents, i.e. the overall dynamics can be *simulated* by its very microscopic elements and global quantities can be compared with experimental data.

Among the two, the first field is by no means more mature. In fact, since the fifties, a huge amount of methods have been developed and many more are currently under study. Books and articles describing what is considered “standard literature” on *Numerical Calculus* are largely available nowadays.

Traditionally, also the term “simulation” refers to numerical methods to compute, for example, the solution of a system of partial differential equations. In contrast, the definition that will be used throughout this manuscript will point to a different narrower meaning.

To simulate a system means to reproduce the “functional behaviour” of the constituents under particular laws which rule the global dynamic of the system itself. These laws are not known in general and are exactly the target of the investigation. Even the variables defining a single system-constituents are not known in general.

This approach is much more valuable as the computing power of today’s computer increases. In fact the number of *micro-constituents* of the system should be, by definition, large [8].

Complex behaviour can occur in any system made up of large numbers of interacting constituents with non-linear coupling, be they atoms in a solid, cells in a living organism, or traders in a financial market.

It is the availability of digital computers that makes possible to solve sophisticated models and, in so doing, to reveal the micro-dynamics of some complicated natural phenomena. Thus, Microsimulation (MS) [66] belongs to the Computational

Sciences and mainly refers to methods similar to those used by computational statistical physics that are being applied to other disciplines [36]. Figure 1 sketches a diagram in which the Computational Sciences are identified as the intersection of physics methods and applied disciplines, with the use of computers as combining element.

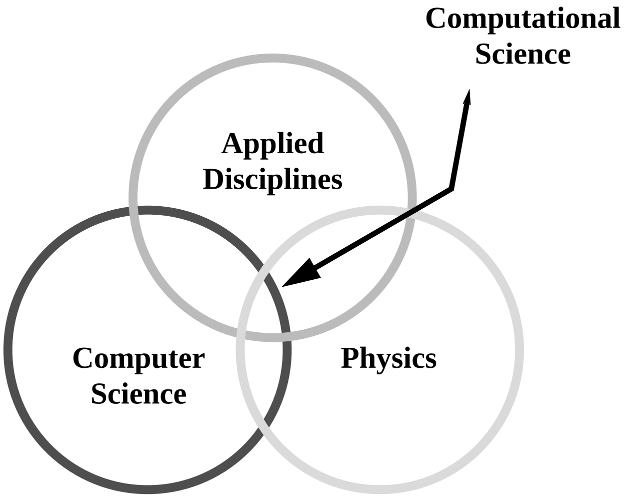


Figure 1 Computational Science

From the mere technical point of view, the Moore's law (computer power doubles every 1.5 years) assures increasing memory and CPU speed to simulate larger and larger systems. Although many problems are now solvable by common personal computers, there are problems for which even the largest parallel machine is not able to find a "real" solution. These problems are called *Grand Challenges*. A Grand Challenge is a large-scale science or engineering computational problem. Examples can be found in Physics, Biology, Chemistry, Materials Sciences, Fluid and Plasma Dynamics, Finance, Environment and Earth Sciences, and so on.

The reason of the intractability of such problems lies in the level of details one wants to take into accounts. For example one may think of a biological cell as a single element ruled by very simple dynamics which brings the cell at most into one excitatory state. In this scenario the amount of memory needed to represent a cell is reduced to the minimum (a bit) and the number of operations to test and possibly switch its state is negligible. Stated like this, even if we wanted to take into account millions of cells at one time we would not run into troubles today if we could access a reasonable workstation. Problems arise when we want to go into the details of the cell. In fact, a cell, is a whole universe for its own, with an unthinkable level of details. Even top supercomputers would not be able to represent all that information. So, what chances do we have? The possibilities stay in between:

we cannot take all the details at once but we can add them one after another as long as the computing power required is available. This philosophy has gone long enough today to allow sufficiently-detailed simulations of complex phenomena.

The present manuscript deals with complex systems composed by many interacting elements. In particular it deals with problems from biology and finance. In both fields the rules governing the micro-behaviour of the constituents (cells, molecules but also traders and brokerage agencies) are mostly unknown. All that is given is the macro-behaviour that can be observed empirically either by experiments (this is the case in biology) or by applying statistical methods to the already given data (this is the case in finance where the use of databases allows to track any transaction worldwide).

This manuscript is divided in two main parts. The first part, composed by chapters 1 – 3, is devoted to what is called "Computational Immunology" and in particular it deals with the microsimulation of the immune system response. A sophisticated MS model called the Celada-Seiden model is discussed in details and then used to investigate the immune response in typical statistical mechanics fashion. The "learning cascade" is proposed to explain the mechanism by which the collective recognition of (reads "the information on") the antigen proceeds as a cascade in a suitable state-space. Chapter 3 shows how it is possible to use a different approach to understand complex phenomena. It deals with a coupled-map system to reproduce the learning cascade first discovered using the microsimulation model.

The second part, composed by chapters 4 – 7, is about the emerging field of "Econophysics". After a description of the stylized facts of financial markets in chapter 4 and short review of the existing models given at the beginning of chapter 5, particular attention is devoted to the Cont-Bouchaud percolation model in section 5.1. In the following chapter 6 a new MS model is presented. This model has many technical points in common with the immunological model presented in the preceding chapters. In particular they both belong to the class of *unbounded lattice gases* that will be defined in the introduction. Finally, chapter 7 deviates a bit from the path of this manuscript (although it shows a model inspired by biology applied to a financial problem) and face up to the problem of forecasting financial time series.

The joining element between the two MS models of chapters 2 and 6 is not only conceptual but also technical in certain respects. In fact they are both being coded following a precise architectural schema. This aspect, together with a short overview of pre-existing models of this kind that are being used since decades in various fields of science, will be presented in the introductory section.

Appendix B and C will report some technical details about the implementation of the two simulation programs mentioned before, to allow them to run on parallel computers. In appendix A it is given a small introduction on the immunology needed in chapter 2. Finally, a glossary of notation with the list of symbols used throughout this manuscript is given. It is safe to say however, that few symbols are being used in more than one contest with different meaning, in places where there is very little danger to generate confusion. Because this manuscript is about computational models, the name of the codes (simulators) presented herein are given in bold.

Part of chapters 2, 3, 5, 6 and 7 have been published on journals on computational physics or complex systems. The corresponding references are given in the section “Erklärung” at the end of this dissertation. I wish to thank from the very beginning all the coauthors.

A final section of this manuscript was mandatory to express my debt of gratitude to all the people who introduced me into this fascinating field.

Cologne, February 2001

Introduction

Nowadays, the scientific study of a phenomena in general consists of three major approaches: theoretical, experimental and computational. The computational aspect becomes more and more important. Computational science has the flavor of both theoretical and experimental science. One must have a very good theoretical background to study a subject by means of computational methods. A good computational method often comes from a thorough theoretical analysis. On the other hand, the analyses of results are not much different from analysing experimental data. Computational methods in science become advantageous when (1) the problem at hand is too difficult to do analytically; (2) an approximate theoretical result may not be reliable, and it is necessary to check with a different method; (3) an experiment is expensive or not feasible at all.

As already mentioned in the preface, computational methods can be roughly divided in two areas, that of numerical analysis and that of computer simulation. Numerical methods include solving linear equations, eigenvalue problems, solving differential equations and partial differential equations, etc. They are very useful and important but are not the core of this dissertation.
In contrast, computer simulations are methods that try to model the physical world *directly*, rather than solving the equations governing the physical processes.

A complex systems (physical, biological, chemical or financial, just to mention a few) can be defined as a system with a large number of degrees of freedom. Thus, microsimulation (MS) is a method to mimic a complex phenomenon through the description of its micro-components. That is, leaving the system free to evolve without too many constraints and simplifying assumptions.

In the following we present techniques belonging to the class of MS methods. These have been applied with success in the field of statistical mechanics [12]. Some of them are being also used in biology and recently in finance [36, 66]. At the end we define a new class of MS models that we use throughout this manuscript to describe two different simulation algorithms dealing respectively with problems from immunology and finance.

The very first definitions are those of “spin” and “lattice”. A ferromagnet can be regarded as a system composed by a large number of elementary magnets placed on the sites of a crystal lattice. To model and to understand the magnet properties of solids, various types of *lattice spin models* have been proposed. Such models are defined by (i) a lattice type (dimension 1,2,3..., and topology, i.e. cubic, triangular and so on); (ii) the possible values of the random variable, called "spin" at each lattice site, that is, the number of possible states a spin can take (these may be either discrete or continuous; also, a spin can be in generalized sense, a single value, a vector or a tensor, although some representations may lack of a physical meaning); (iii) the interactions among spins, in terms of rules determining the way the value of the spins are coupled and how they change with time. Iterating the interaction rules, one gets a discrete dynamical system.

The following overview of spin systems is by no means complete. Moreover, we voluntary avoided to talk about the interaction rules which, together with the topological definition of a spin system, is the most important element to distinguish one application from another.

If the total number of lattice sites is L, we identify a spin with a stochastic variable s_n, for $n = 1, 2, \ldots, L$. According to (ii) one deals with different spin models. The most popular are the enunciated below.

Ising model A spin may take on just two values; "up" or "down", usually $s_n = +1$ or $s_n = -1$.

Potts model of the K^{th} order It is a generalization of the Ising model with the spins taking one of K possible values, i.e. $s_n = 1, 2, 3, \ldots, K$.

XY model Each spin is a complex number of absolute value 1, i.e. $s_n = e^{i\phi_n}$.

Later, other models have been derived from these definitions.

Cellular Automata (CA) Inspired by the early work of J. von Neumann [121] on self-replicating machines, they are discrete dynamical systems where each spin per lattice site is updated according to the state of the spin in its neighborhood [127]. CA are being used to model many physical systems but seem more suited to model biological systems.

Lattice Gas Automata (LGA) Lattice gas automata were introduced by Frisch, Hasslacher and Pomeau as a means to solve the Navier-Stokes equations of fluid dynamics. The two dimensional triangular lattice gas (figure 2) is indeed called FHP lattice [46]. A lattice gas is like the Potts model in which the states of particles represent velocities. The FHP model is associated to particular types of lattices with peculiar interaction rules (collisions conserving mass and momentum). The

FHP model is a two-dimensional triangular lattice, thus the number of velocities K is equal to six.

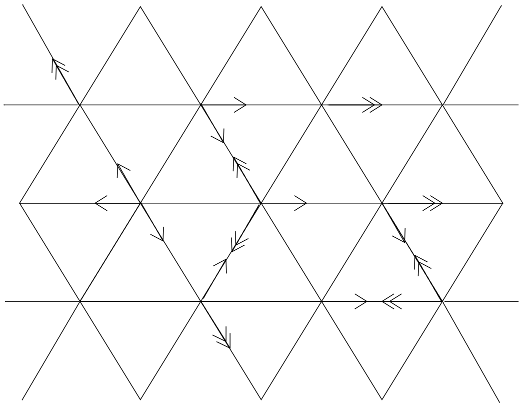


Figure 2 Triangular lattice. Particles at time t and $t+1$ are marked by single and double arrows, respectively.

Integer Lattice Gas Automata (ILGA) A generalization of lattice gas has been proposed with the name of Integer Lattice Gas Automata [17]. The generalization is given allowing more than a single particle per each direction to stay on a lattice site, e.g. $s_n \in \{1, \ldots, K\}^r$ with fixed number of particles per site equal to r.

Through this step-wise definition of spin models we come to the definition of a spin system in which each site of the lattice contains exactly r particles, each in one of K different states-velocities. To reach the goal of defining a model to describe the systems of the following chapters, we still need two further generalizations: (1) we want to represent different particles types each having their own micro-state space; (2) we do not want a fixed number of particles on each lattice site.
The first of these requirements leads to a definition of spin with many components $s_n = \mathbf{s}_n = (s_n^{(1)}, s_n^{(2)}, \ldots, s_n^{(E)})$ with E the number of different types of entities [1]. Each agent belonging to class $e = 1, \cdots, E$, can be found in a micro-state taken from a *discrete* set of states $\{1, \cdots, K(e)\}$ whose number depends on e.
The second point requires to consider $r \equiv r(x, t)$ that is, the number of entities on each lattice site x is a function of time t and x. In general, the number of entities in a given lattice site depends on the diffusion process we choose. In practice it is sufficient to choose r as the maximum number of particles on the lattice during the whole simulation to recover the definition of integer lattice gas.

---

[1] It is time now to use also the term *entity* in place of particle. In fact “entity” fits better the meaning of (e.g.) agents in a stock market or cells in the immune system.

Because we allow $r(x, t)$ to grow without constraints we set apart this particular case of *unbounded capacity* and call it Unbounded Lattice Gas.

Unbounded Lattice Gas (ULG) It is a lattice gas with unlimited number of particles on each lattice site. Particles belong to different classes $e = 1, \cdots, E$. They may take on one micro-state from a set $\{1, \cdots, K(e)\}$ which in turn depends on the class e to which they belong to.

Summarizing, we are able now to define models where the different entities belongs to different classes. They occupy the lattice sites with no constraint on the number. They interact *locally* instead of interacting with the neighborhood as in CA models. Finally, eliminating the constraint on the occupation number we allow the particles to diffuse freely on the lattice grid. At this time we consider the particles to follow the classical *Brownian* motion. General non-uniform diffusion schema are also well defined thanks to the unconstrained capacity $r(x, t)$.

As already anticipated in the preface, the use of ULG as formal definition of the microsimulation systems developed and discussed in the following chapters is justified by the availability of large-memory computers. Instead of storing a single bit like in the Ising model or at most few bytes as in the Potts model to keep the memory consumption at minimum, [2] we can now represent *particles* (cells, atoms, molecules, traders etc) as a collection of information or *attributes*. Thus, the informative structure representing a single particle is heterogeneous as we allow to mix binary information, integer numbers or even arrays of more complicated records. Note that we intentionally restricted ourself to the use of integer numbers to represent the internal states. The reason is to avoid floating points operations to assure unconditional *numerical stability* to the simulation algorithm. Moreover, given the unbounded capacity of the lattice, the choice of a *static* data structure is clearly wrong. Indeed, a *dynamic* memory allocation is in order. In practice, all we need to represent a d-dimensional [3] ULG is a *pointer* to a list of "records" containing the information structure of the entities *for each* lattice site.

In our models the complex behaviour of the entities is subjected to precise *state-changes* upon interaction. Every single entity can be thought as a *Stochastic Finite State Machine* (SFSM) [80] which processes information and changes its state according to the result of the interaction with other entities, or with external fields. Probabilistic or stochastic models should not be confused with *non deterministic*

---

[2] For Ising and Potts models many computing techniques which optimize memory usage have been developed. For example, according to the kind of coding approach one speak about "multi-spin" and "multi-site" coding [61, 19].

[3] We will always use $d = 2$.

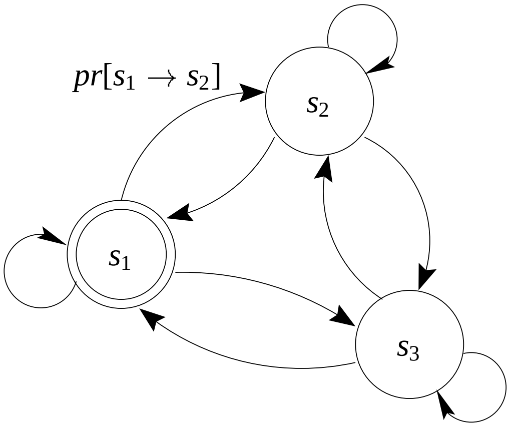


Figure 3 Stochastic Finite State Automata. The probability $pr[s_1 \to s_2]$ to switch from state one to state two, for example, can be given or can be the outcome of a more complicated procedure. In our case these probabilities are computed by complex rules (see chapters 2 and 6).

models in theoretical computer science [80]. A typical example of stochastic system is a Markov chain where each state transition is subject to a given probability.

A typical diagram showing a stochastic finite state (automata) is given in figure 3. The transition between the states s_1, s_2 and s_3 is stochastic. The transition probabilities can be fixed or changing in time. In our case they depend on the outcome of *more or less* complicated interaction rules between entities. They can also depend on some global quantities or external fields. For them, a special syntax has been developed and examples of its use will be given in chapter 2 to describe the model of the immune response.

Finally, because the particles interact locally (i.e. inside each lattice site) and only after they diffuse to adjacent sites, we can “easily” divide the CPU-load distributing the lattice grid to different processors of a parallel machine [100]. Message passing among processors is needed only during the diffusion phase. This allow us to simulate a large number of interacting entities with a high level of details.

Part One

Chapter 1

Computational immunology

Vertebrate animals have an Immune System (IS) protecting them against diseases. The immune system is composed by many cooperating agents (cells and molecules) whose task is to recognize and defeat “foreign agents” as virus, bacteria and dangerous molecules.

The panorama of immune system models is quite large. This short chapter is meant to give a very introductory review of some of the existing automata models chosen according to my personal knowledge. The review of Perelson and Weisbuch [95] reports on the different models developed to study the immune response from the physics point of view. In that work both equation-based and computer-simulation models are discussed. Instead, in this chapter, we concentrate on the models where computer simulation methods similar to statistical physics are employed. For differential equations we refer to the recent review of Lippert and Behn [72], the already mentioned review of Perelson and Weisbuch [95] and also the two volumes “Theoretical Immunology” [93, 94]. A review about the specific use of cellular automata in modeling the immune response is given by R.M. Zorzenon Dos Santos in [130].

In the next chapter we will concentrate on the Celada-Seiden model which is one of the most detailed lattice gas automata for the immune system response. Its complexity derives from the fact that in addition to different cellular populations considered, also a molecular representation of the cell and molecular binding site is given in term of specific recognition between bit strings. Moreover, a kind of *intra-cellular* interaction is modeled for the presence of the *Major Histocompatibility Complex* allowing for a direct self-non self discrimination via lymphocyte selection in the *thymus*.
The basic mechanisms of the immune system, or at least what is sufficient to known here, are given in appendix A.

1.1 The model of Kaufman, Urbain and Thomas

One of the first application of discrete automata to immunology is the one of Kaufman *et al.* [57] in 1985. The original model considers five types of cells and molecules: antibodies (A), helper cells (H), suppressor cells (S), white blood cells (B) and virus (V). Each entity is represented by a variable denoting "spin up" (high concentration) and "spin down" (low concentration). The rules modeling the dynamic evolution of these variables are expressed by logical operations. The application of the rules is iterated over discrete time and the dynamics is observed. The discrete evolution rules are:

$$\begin{aligned}
A(t+1) &= V(t)\ \texttt{AND}\ B(t)\ \texttt{AND}\ H(t)\\
H(t+1) &= H(t)\ \texttt{OR}\ V(t)\ \texttt{AND}\ \texttt{NOT}\ S(t)\\
S(t+1) &= H(t)\ \texttt{OR}\ S(t)\\
B(t+1) &= H(t)\ \texttt{AND}\ (V(t)\ \texttt{OR}\ B(t))\\
V(t+1) &= V(t)\ \texttt{AND}\ \texttt{NOT}\ A(t)
\end{aligned}$$

where `AND`, `OR` and `NOT` are the usual logical operators of the *first order predicate calculus*, respectively *and*, *or* and *not*. There are five fixed points in the state space composed by $2^5 = 32$ points. Fixed points identify the global state of the immune system: naive, vaccinate, immune, paralyzed, paralyzed and sick.

1.2 The model of Weisbuch and Atlan

This primitive model was followed by many other models. For example Weisbuch and Atlan [125] focused on the special case of auto-immune diseases like multiple sclerosis, in which the immune system attacks the cells of the nervous system of our own body. As the model of Kaufman *et al.*, this model uses five binary variables representing: killer cells (S_1), activated killers (S_2), suppressor cells (S_3), helpers (S_4) and suppressor produced by the helpers (S_5). The different types of cells influence each other with a strength which is 1, 0 or -1. At the next time step, the concentration of one cell is unity if the sum of the interactions with the various cell types is positive; for zero or negative sums, the concentration is taken as zero.

In formulas

$$\begin{aligned}
S_1(t+1) &= \text{sgn}\left(\sum S_1(t) + S_4(t) - S_3(t)\right) \\
S_2(t+1) &= \text{sgn}\left(\sum S_1(t) + S_4(t) - S_3(t) - S_5(t)\right) \\
S_3(t+1) &= \text{sgn}\left(\sum S_1(t)\right) \qquad (1.1) \\
S_4(t+1) &= \text{sgn}\left(\sum S_1(t)\right) \\
S_5(t+1) &= \text{sgn}\left(\sum S_4(t)\right)
\end{aligned}$$

where $S_i(t)$ denotes the concentration of the ith component at time t and the function $\text{sgn}(x)$ defined on the natural numbers ($\mathbb{N}$) is 1 if $x > 0$ and 0 otherwise. This model shows the existence of only two basins of attractions over $2^5 = 32$ possible states: the empty state where all the concentrations are zero and a state where only activated killers disappear while the other four concentrations are unity.

Further generalizations of this model consider the same dynamics but putting the cells on a lattice to allow simulations in statistical physics way (Ising-like models). In Dayan *et al.* [35] the authors put five variables on each lattice site corresponding to five boolean concentrations (0 or 1). Recalling the definitions of the precedent chapter we may see the model of Dayan *et al.* as an Integer Lattice Gas with $r = 5$ (five entities) and $K = 2$ (two states per entity).
Each site influences itself and its nearest neighbours in the same way as in the model of Weisbuch *et al.*. For a square lattice of $L \times L$ sites there are $5 \times L^2$ spins. The main difference is that in this model the summation in eq(1.1) runs over the site itself and its nearest neighbours.
This lattice-version of the Weisbuch-Atlan model is found to have a simpler dynamics than the original model as the number of fixed points is found to be smaller than in [125].

1.3 The model of Pandey and Stauffer: the case of AIDS

Pandey and Stauffer further extended the model of Kaufman *et al.* using a probabilistic generalization of deterministic cellular automata. Their model focus on a possible explanation of the time delay between HIV infection and the establishment of AIDS [89, 90]. They represent helper cells (H), cytoxic cells (S), virus (V) and interleukin (I). The interleukin molecules produced by helper cells induce the suppressor cells to kill the virus. The dynamics is given by the following rules:

$$\begin{aligned} V(t+1) &= H(t)\ \texttt{AND NOT}\ S(t) \\ H(t+1) &= I(t)\ \texttt{AND NOT}\ V(t) \\ I(t+1) &= H(t) \\ S(t+1) &= I(t) \end{aligned}$$

The dynamics of this model has been investigated. Oscillatory behaviour followed by a fixed point where the immune system is totally destroyed, similar to the real onset of the AIDS, is found.

1.4 The bit-string model of Farmer, Packard and Perelson

A peculiar class of models in immunology is the so-called *bit-string* models. The first of these models has been introduced by Farmer, Packard and Perelson [42] to study the theory of the *Idiotipic Networks*. All the molecules and cell binding sites (e.g. cell receptors) are modeled as binary strings of length l. Antibody molecules are assumed to recognize the antigen whenever their bit strings can be matched complementarity. The specific rule that was used was to align the bit string and require a complementary match over a stretch of at least r adjacent positions. For string matches over exactly r adjacent positions, a low affinity was assigned, say 0.1. If the match was bigger than r adjacent positions the affinity was increased by means of a certain formula which depends on some probability counts.

The idea of bit strings has been taken up by the Celada-Seiden model. In that, the match between virus and lymphocyte receptors, for example, is given considering the number of complementary bits in the bit-wise comparison. For example, if the lymphocyte B is equipped with the binary string 00010101 ($l = 8$) while the virus is represented by the string 11101010 then the probability to trigger a response is very high (see also figure 2.2). In this model the bit-string match is not required to be perfect, i.e. some *mismatches* are allowed like for example one bit from eight in the case above.

This list of immunological models is by no means complete. We send the reader to the aforementioned bibliography for a better reference (see also the chapter on immunological models in [36]).

Chapter

2

The Celada-Seiden model

Cellular Automata based models have proven capable of providing several new insights into the dynamics of the immune system response. A qualitative picture of the IS behavior can be obtained with small-scale simulations. However for a more detailed analysis and to further validate the models, large scale simulations are required.

One of the most prominent attempts to cope with the quest for biological fidelity is the **IMMSIM** (Immune Simulator) automaton, developed by P.E. Seiden and F. Celada in 1992 [29, 106]. We will refers herein to **IMMSIM** to identify the computational model, i.e. the algorithm or the code, while CS-model refers to the conceptual model in term of logical statements describing the entities and their interaction rules.

IMMSIM belongs to the class of immunological cellular automata, but its degree of sophistication sets it apart from simpler CA in the Ising-like class [57, 90]. In the words of Franco Celada, “in machina” experiments should complement the traditional *in vivo* and *in vitro* experiments of the immunologists.

Immunologists distinguish between humoral and cellular response. They also set apart the clonal selection and *idiotypic networks* theory. Formulated by Niels K. Jerne in 1973, according to the idiotypic network theory [13], the organism forms antibodies combating its own antibodies in such a way that a kind of immunological balance and an exchange of information is established in the immune system in the same way as in the central nervous system. Together with Georges Kohler and César Milstein, Niels K. Jerne was awarded the Nobel Prize for Physiology/Medicine in 1984.

The CS-model explicitly implements both kind of response (cellular and humoral) but rest its foundation on the *clonal selection theory* of the Nobel Price F.M. Burnet (1959) [20] developed following the track first highlighted by P. Ehrlich at the beginning of the twenties century. The theory of the clonal selection states that the immune response is the result of a selection of the “right” antibody by the antigen itself, much like the best adapted individual is selected by the environment in the theory of natural selection of Charles Darwin. It is noteworthy the fact that F.M. Burnet got the Nobel Price in the 1960 together with P.B. Medawar for his works on the *acquired immune tolerance* and *not* because of his discoveries about the clonal selection theory.

The original implementation of the model made use of APL2 that is an interpreted language with no “explicit” dynamic memory allocation capability. This choice along with the intrinsic complexity of the model prevented the authors from running any but “relatively” small scale simulations.

This situation is not uncommon. The study of complex systems behavior by means of simulations of the single entities *micro-dynamics* is extending from the physics and biology to other fields like the financial markets analysis [68]. Unfortunately, many times these complex models are implemented in such a naive way that the code must be considered just a “proof of concept” rather than a real working tool.

When we started to face the problem of extending the **IMMSIM** automaton capabilities, we decided from the very beginning that the new version of the simulator had to be a parallel code written in a highly efficient, compiled (and portable) language. Armed with these considerations, we developed a parallel version of the **IMMSIM** automaton named **CImmSim**, coded by means of the C language [25] and PVM as message passing library.

CImmSim was designed according to criteria of *openness* and *modularity* to allow smooth upgrades and addition of new features (cells, molecules, interactions and so on) for future investigations. As a matter of fact, the aforementioned modularity has been recently exploited when the description of new types of cells, involved in cellular response, has been introduced with a *reasonable* effort. The current version of the code (version 4.3) is able to simulate both the *humoral response* described in [29, 106, 25] and the *cellular response* described in [16]. In this respect it is the most advanced parallel version of the Celada-Seiden automaton. Actually, there is another complete version of the **IMMSIM** automaton which is developed and maintained by P. Seiden himself [105]. However, being still based on APL2, that code imposes hard constraints on the maximum system size. The corresponding limit of **CImmSim** is, up to date, almost two orders of magnitude bigger than that.

In the following paragraph we will present the “computational” version of the CS-model. The underlying formal model is that of an Unbounded Lattice Gas derived in the introductory chapter.

2.1 The computational model

A single lymph node of a vertebrate animal is mapped onto a bidimensional triangular lattice (six neighbour sites) $L \times L$, with periodic boundary conditions in both directions (up-down, left-right).

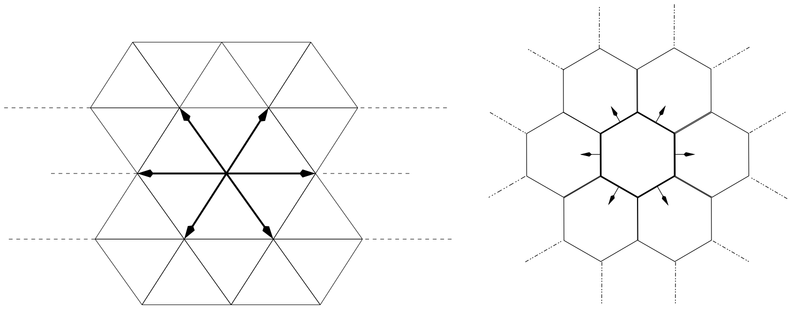


Figure 2.1 The hexagonal lattice (left) is equivalent to the "honeycomb" lattice. The difference is only apparent.

Cells and molecules belonging to the IS cooperate, with different roles, to the defense of the host organism from attacks of potentially offending invaders called Antigens (Ag).

CImmSim belongs to the class of *bit string models* [42]. The bonds among the entities are described in terms of *matching* between binary strings with fixed directional reading frame (or the dual *mismatch*). Bit strings represent the "binding site" of cells and molecules.

The following description of **CImmSim** is given by key points: i) entities representation, ii) repertoire, iii) affinity function, iv) interactions among entities, v) hyper-mutation.

2.1.1 Entity/State description

A simple way to describe the model is to look at the possible states for each biological entity represented. The entities are divided in cells and molecules. The former are much more sophisticated structures compared to the latter because of their inherent higher complexity. Table 2.1 lists all the entities used in the model. For the molecular entities we need to make just a few remarks:

- IFN and D-signal belong to the class of the *lymphokines*, i.e. molecular carriers of physiological signals used by the cells to acknowledge the occurrence of certain events.

Cellular entities	Molecular entities
Lymphocyte B (**B**)	interferon-γ (**IFN**)
Lymphocyte T helper (**Th**)	Danger signal (**D**)
Lymphocyte T Killer (cytotoxic) (**Tk**)	Immune complexes or Ab-Ag binding (**IC**)
Macrophage (generic antigen processing cell) (**APC**)	Antigen or generic Virus (**Ag**)
Epithelial (generic target) cell (**EP**)	Antibody (**Ab**)
Lymphocyte Plasma B (**PLB**)	

Table 2.1 Cellular and molecular entities of the CS-model.

- Major Histocompatibility Complex (MHC); these are not listed in table 2.1 since they are not considered independent entities. This means that they are present inside other entities (like B, APC and EP cells) [95] whereas the other molecules can circulate in the lymphatic system. MHC molecules are divided in class I and class II. Further details will be given in section A.2.
- *Peptide(s)* and *epitope(s)* refer to fragments of a single molecule. For instance, the antigen's epitope is, by definition, the part of the antigen which is bound to cell receptors.

The major difference among cellular and molecular entities is that cells may be classified on the basis of a *state* attribute. The state of a cell is an artificial label introduced by the logical representation of the cells behavior.

Every single cell can be thought as a *Stochastic Finite State Machine* (SFSM) which processes information and changes its state according to the results of the interaction with other cells. The transitions among the various states are determined by stochastic events.

E/S	ACT	INT	INF	EXP	LOA	RES	STI	DEA
B	×	×		×			×	
Th	×						×	
Tk	×						×	
APC	×	×		×	×	×		
PLB	×							
EP	×		×		×			×

Table 2.2 Cellular entity/state.

The set of possible transitions for our SFSM's are defined by the interaction procedures reported in table 2.3. Table 2.2 summarizes the entity-state description. It should be read as follows: the cellular entity E can assume state S if the corresponding entry in the table is crossed. The semantic of the labels is the following:

- ACT (Active) means *normal* state. This is the initial state for each cell;
- INT (Internalized) means that an antigen presenting cell (like B and APC) has phagocitated one antigen. This state follows to an interaction with an antigen;
- INF (Infected) means that one antigen (now called virus) has penetrated the cellular membrane of the cell. After that, the virus duplicates inside the host cell;
- EXP (Exposing) means that the cell has phagocitated one antigen and has already processed it. If the bind with the MHCII molecule is successful then the cell is exposing the MHCII molecule bond with one antigen peptide;
- LOA (Loaded) means that the cell expose the MHCI molecule loaded with one antigen peptide;
- RES (Resting) means that the cell is in resting state, i.e. inactive;
- STI (Stimulated) means that the cell is in duplication phase;
- DEA (Dead) means that the cell has been marked to die by *lysis* by a Tk (Tk "kills" the cell).

All other entities (i.e. the molecules) may be considered always *active*, i.e. are ready to interact. The set of plots in figure 2.4 to 2.10 show the total number of entities for a large simulation.

The state, along with other information, is stored in a *flag byte* for each cell (see paragraph B.2 in the appendix). C language specific macros have been defined to access and modify the flag byte.

2.1.2 The repertoire

In the CS-model a clonotypic set of cells is characterized by the receptor which is represented by a bit-string. The bit-string length l is clearly one of the key parameters in determining both time and space complexity of the algorithm that simulate the behavior of the whole set of entities as the number of potential repertoire of receptors scales as 2^l (see [25, 27]).

Every entity is represented by a certain number of molecules, the receptor being one of these. The repertoire is then defined as the cardinality of the set of possible instances of entities that differ in, at least, one bit of the whole set of binary

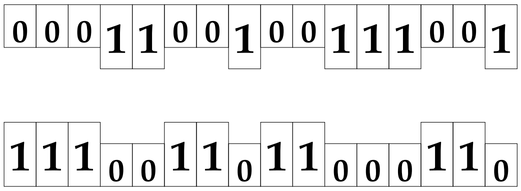


Figure 2.2 Bit-string representation of the binding site of cells and molecules. In this figure two complementary strings of length $l = 16$.

strings used to represent its attributes.
Indeed, the cells equipped with binding sites and the antibodies, have a potential repertoire of $2^{N_e l}$. Where N_e indicates the number of binary strings used to represent receptors, MHC-peptide complexes, epitopes and so on, of the entity e. Other entities do not need to be specified by binary strings so their repertoire is just one (i.e. $N_e = 0$). An example are the interleukin molecules like the Interferon–γ (IFN) and the Danger signal (D). Table 2.4 summarizes the number of strings used to represent each entity. Since the number of different MHC molecules (class I or II) is limited to a few units, they do not contribute to the complexity of the cells. In other words all cell equipped with MHC molecules carry the same number of the same molecules. Actually, we need to represent the MHC-peptide complex produced by the internal processing of antigens (*endocitosys*). This bit string is important because it is used for further recognitions of the T (helper of killer) lymphocytes.

external interactions	cell-internal interactions
B – Ag, B – Th, Ab – Ag, Th – APC, Tk – APC, Tk – EP, APC – IC, APC – Ag, EP – Virus,	B – MHCII, APC – MHCII, APC – MHCI, EP – MHCI

Table 2.3 External and internal interactions. Here antigen and virus are the “same” entity.

Although the Ag are specified by n_p+n_{ep} binary strings (n_p indicates the number of peptides whereas n_{ep} is the number of epitopes), they do not need to be explicitly represented by a repertoire of $2^{n_p+n_{ep}}$ because in the current version of the simulator there is no support for antigen mutation. As a consequence the number of antigen represented at run time is limited by the number of *different* antigens we plan to inject into the host. Thus, during the simulation, we need just to distinguish among n_j number of injections (cfr table 2.4). It follows that also the IC’s may be treated in this simplified way.

2.1.3 The affinity potential

Some entities have, on their surface, molecules, usually called *receptors* or *binding sites*, which are in charge of recognizing the antigen. In this model two entities equipped with receptor interact with a probability which is a function of the *Hamming distance* [51] between the binary strings representing the entities' binding site. This probability is called the *affinity potential*. For two strings s and s' such probability is max (i.e. equal to 1) when all corresponding bits are complementary ($0 \leftrightarrow 1$), that is, when the Hamming distance between s and s' is equal to the bit string length. A good and widely used analogy is the matching between a lock and its key.
If l is the bit string length and m is the Hamming distance between the two strings, the affinity potential is defined in the range $0, \ldots, l$ as follows:

$$\upsilon(m) = \begin{cases} \upsilon_c^{(m-l)/(m_c-l)}, & m \geq m_c, \\ 0, & m < m_c. \end{cases} \tag{2.1}$$

where $\upsilon_c \in (0,1)$ is a free parameter which determines the slope of the function whereas m_c ($l/2 < m_c \leq l$) is a cut-off (or threshold) value below which no binding is allowed (see figure 3.1). With **CImmSim** it is possible to choose among four different affinity potentials. The impact of a different affinity potential shape on the IS dynamics will be discussed in section 2.3.

2.1.4 The interactions

The interactions among entities are described in terms of state transitions (*condition - action* description). They can be divided in two categories: *external interactions*, which happen among cells and molecules having the same position on the lattice and *internal (to the cell) interactions*, that account for MHC Ag-peptide interactions *inside* the phagocitating (B and APC) or infected (EP) cells.

External interactions.

There are 11 different interactions among the whole set of entities. The syntax to describe each interaction is the following:

```
INTERACTION : < involved entities >
SPECIFIC    : < Yes | No >
MATCH       : < involved molecules >
CONDITION   : < allowed state for <involved entities> >
ACTION      : < final state for <involved entities> >
```

where `involved entities` are the two interacting entities;
the field `SPECIFIC` is `Yes` if the interaction probability depends on the matching degree between the `involved molecules` of the two entities and is `NO` otherwise (*aspecific* bind).
The conditions are expressed in terms of *first order predicates* by means of the logical AND, OR, NOT and *boolean unary operators* that check each possible state of the entities looking at their *state-flags* (cfr B.2): `IsACT`(·), `IsINT`(·) and so on; the name of the operator is self-explaining; e.g. `IsACT(cell)` is true if `cell` is found in state ACT.
The state transitions are registered by flipping the corresponding flags in the state-flag-byte as described in the paragraph B.2 of the appendix. To this purpose a set of unary operators like `DoACT`(·), `DoINT`(·) and so on, have been defined. In addition, the operators `Kill`(·) and `Create`(·) respectively delete or create the specified entity. For example:

```
INTERACTION : B, Th
SPECIFIC    : Yes
MATCH       : B-MHCIIpeptide molecule, Th-receptor
CONDITION   : IsEXP(B)  AND  IsACT(Th)
ACTION      : DoSTI(B)  AND  DoSTI(Th)
```

means that the specific interaction between the receptor of Th and the MHCII-peptide molecule exposed on the surface of a B cell may happen only among instances of active Th's and instances of exposing B's. No other state is allowed. If the interaction really happens (as determined by the stochastic event), the action is to update both Th and B to state STI (note that we are not specifying the details of the actions taken to store the information required for successive processing).

Moreover some entities have more than one receptor (or generic binding site) like the antigen that may be represented with two or more epitopes. In this case the interaction is allowed if *at least one* of their binding-site matches. In particular the action are undertaken as soon as one match is successful (greedy paradigm).
For every *aspecific* binding (i.e. `SPECIFIC: No`) the probability to bind is given as parameter and do not depends on any match (the field `MATCH` is empty).

Each of these procedures examines the data structures (see appendix B.1) of the two involved entities looking for a simultaneous true value of the predicates expressed in `CONDITION`. For each *specific* interaction the Hamming distance between the `involved molecules` specified by the tag `MATCH` is computed as described in section B. After that, the probability of a successful interaction is obtained by looking at the affinity function. At this time, a random number between 0 and 1 is generated. If the random number is less than the interaction probability, then the actions specified in the tag `ACTION` are undertaken. These actions are usually composed by very few assignments so they do not constitute a major overhead compared to the scanning of the two lists.

One may argue that it would be helpful to have a single data structure for each possible state. Instead, the advantage of keeping all the cells mixed in a single list is that code upgrades are simpler.

A list of all the interactions (external and internal) is reported in table 2.3. The extensive description of these interactions is not the goal of the present work.

Entity	N_e	Repertoire
B, APC	2	2^{2l}
Th, Tk, EP, PLB, Ab	1	2^{l}
IC, Ag	$n_p + n_{ep}$	$2^{(n_p+n_{ep})l}$
IFN, D	0	2^{0}

Table 2.4 Entity/Repertoire. For example, N_e is 2 for the B cells because each B cell carries a receptor and a MHCII-peptide molecule. For the IC and Ag, we show here the theoretical repertoire. Actually, in the current version of **CImmSim**, the number of possible choices for these entities is reduced to n_j as explained in section 2.1.2.

Cell-internal interactions.

The syntax for the description of the cell-internal events is very similar to the previous case. The only difference is that one of the involved entities is always the infecting/phagocitated antigen. This is, from the computational viewpoint, a major advantage since just one data structure must be scanned to select which cells will be processed. We report here just an example, whose meaning follows the previous one.

```
CELL INTERNAL INTERACT. : B
SPECIFIC                : Yes
MATCH                   : MHCII-molecule(s), Ag peptide(s)
CONDITION               : IsINT(B)
ACTION                  : DoEXP(B)
```

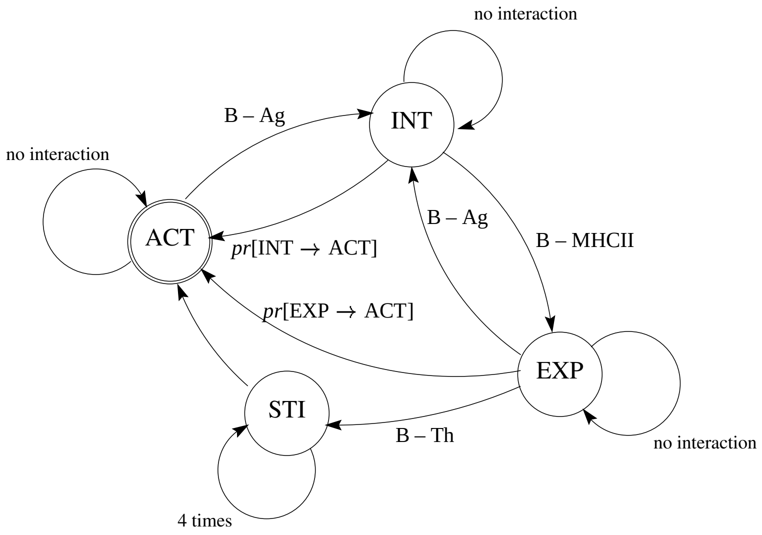


Figure 2.3 Stochastic finite state automaton corresponding to B cell behaviour.

An example of finite state automaton applied to the behaviour of the entities of this model is given in figure 2.3. The B cell state dynamics is considered. A B cell starts in the active state. After having recognized a virus it goes in the internalized state. If the internal recognition of the peptides with the MHC class II molecules is successful, then it goes to the exposing state. If not, there is a chance to get back to the active state otherwise keep trying with the internal B–MHCII interaction. From the exposing state it may goes either to the stimulating state or back to the active state, depending on the interaction with a Th cell. It stays in the stimulating state for a certain number of time steps while it creates clones of itself. After the duplicating period is expired it goes back to the active state, ready to start this cycle again.

2.1.5 The mutation process

"Hyper-mutation" is a term used for indicating a set of complex phenomena whose result is a mutation in the portion of the DNA of the lymphocyte B coding for the *variable region* of the antibody [124].

In **CImmSim**, mutation of each string representing the cell receptor in the duplicating B cells, is implemented in two different ways that the user may choose at compile time.

- *Binomial distribution*: the number of mutations follows a binomial distribution with parameters l and p_b (p_b is the probability to change a single bit).

This schema assumes that the single bit mutation probability is IID (Independent and Identically Distributed) with respect to the other bits of the same string.

- *Poisson distribution*: the mutation process follows a Poisson law. [1]

The effects of the hyper-mutation on affinity maturation can be easily highlighted by **CImmSim** by means of the parameter h_c (“hole in the repertoire”). If the value of h_c is within the range $[m_c, l]$, the bone marrow is not allowed to produce cells with a “natural” affinity to the antigen higher than h_c. As a consequence, hypermutation becomes the only mechanism for the IS to enhance the affinity to the antigen.

2.1.6 Simulations

To show the capabilities of **CImmSim** we present the results of a very large simulation. It reproduces an immunization experiment in which the antigen is injected in the body in two different time steps to stimulate the immune response. The immunization consists in a faster secondary response due to the memory the system has produced during the first one. The run has been performed on a shared memory machine and is very demanding especially for what concerns the memory requirements. Details are given below.

2.1.7 The parameters

The bit string length l has been set equal to 24, corresponding to a potential repertoire of 16777216 distinct receptors and molecules. The initial number of cells has been set equal to 0.13×10^6 per type (i.e. a total of 650000 cells). Two injections of 0.5×10^6 antigens at time step 0 and 120 have been considered. The mutation rate per bit p_b (see section 2.1.5) is equal to 5×10^{-3} which means a total mutation rate per string of $1-(1-p_b)^{24} \simeq 0.11$. The cut-off value of the Hamming distance (see section 2.1.3) is $m_c = 16$ whereas h_c (see section 2.1.5) is equal to 20. We recall that receptors with affinity greater or equal to h_c are created merely by mutation from active (i.e. with affinity $\geq m_c$) clones of B cells during the duplication

---

[1] A simple generator (even if somewhat inefficient) for random numbers taken from a Poisson distribution with parameter λ, is obtained using the *rejection method* [98]: *if* $x_1, x_2, \dots, x_k$ *is a sequence of random numbers via uniform distribution between 0 and 1, then k, taken as the first integer for which the product* $\prod_i^k x_i$ *is less than* $e^{-\lambda}$*, i.e.* $min_{k\in\mathbb{N}} \left\{k : \prod_i^k x_i < e^{-\lambda}\right\}$*, is distributed as a Poisson with parameter* λ. After the number of mutating bits in a string has been determined, the position of such bits is chosen at random among the l possibilities. In such a way a bit can be selected more than once and there is the chance that it is finally left unchanged (if selected an even number of times).

phase.
The virus proliferation rate has been set equal to 0.01 for free-circulating viruses and to 0.2 inside target cells. The size of the bi-dimensional grid is 1024 points ($L = 32$).

The value of the parameters is such that the system is able to eliminate all the antigens during the response. Note that, with an higher virus-growth rate, the infection may be strong enough that the system fails in removing the antigens. This case is well described in [16].

2.1.8 The dynamics

Figure 2.4 shows the number of antigens versus time. At time step 0 and 120 half million antigens are uniformly injected on the lattice. The first response takes some time to mount since the production of antibodies and specialized T killer cells requires many recognition steps. However, the response to the second injection is very quick because the system has *memory* capabilities (see figures 2.5, 2.6 and 2.8 respectively for B, Th and Tk cells).
The second peak of Ag during the primary response (about 17^{th} time step) is due to the antigens proliferation inside the EP cells. When such antigens reach a critical number, the cell explodes spreading them on the lattice site. The second peak is very high because in the time steps following to the first injection the virus infections are almost simultaneous on a large number of EP cells (see the plot at the bottom of figure 2.7). The time required to the antigen to reach the critical number inside the target cell depends on its grow rate and the threshold value which are both settled as input parameters.
The large number of Interferon-γ in bottom figure of panel 2.4 indicates that many APC are in the *exposing* state during the first response.

The plots in figure 2.5 describe the evolution of the B-cell population during the simulation. The top plot shows the B cells affinity classes. Two B-cells belong to the same affinity class if their receptors have the same Hamming distance from the antigen. We define the mismatch of two bit strings as the difference between the bit string length l and their Hamming distance. So there are affinity classes with mismatch m equal to $0, 1, \ldots, l - m_c$. The number of cells belonging to any affinity class is normalized by division with the binomial coefficient $\binom{l}{m}$. Since the value of h_c (see section 2.1.5) is equal to 20, the value of the mismatch for the cells produced by the bone marrow is equal or greater than 4. Nevertheless an affinity class with mismatch 3 is generated by means of the hyper-mutation mechanism. Note that the class with best mismatch (i.e. 4) grows faster than the others during the secondary response. This is exactly the affinity maturation phenomenon expected by the clonal selection theory (as reference see bibliography in [29, 106, 25, 119, 95]). During the primary response (between time step 0 and 30) the class with mismatch

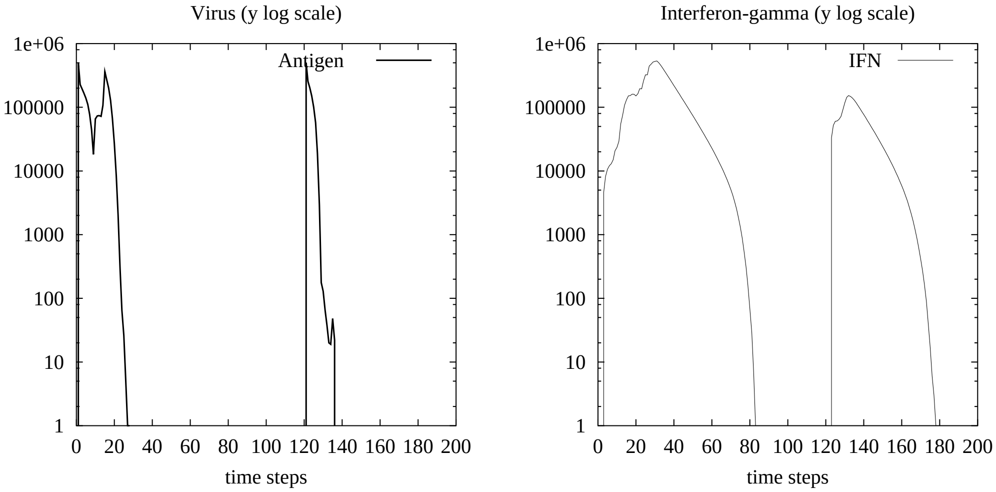


Figure 2.4 Population of antigen-virus on the top; Population of interferon-γ at the bottom.

5 wins (as expected) the race with the others classes.
The bottom plots in panel figure 2.5 shows the number of *memory* and *naive* B cells along with their sum. The number of naive cells remains constant because stimulated B cells produce just memory and plasma cells.
The plot at the bottom of the same panel shows the number of B cells for three (out of four) of the possible "stable" states. The fourth state ("Internalized" in this case) is considered "unstable" because the cell switches to another state within the same time step.

During the first and second response there is a relative large fraction of cells in *exposing* or *stimulated* state. The shift of few time steps between the two corresponding curves is not surprising since the latter state follows the former in the recognition process.

Figure 2.6 shows the population of T helper cells. Their evolution is closely related to the B cells one.

The top plot in figure 2.7 shows the number of APC cells for any possible state. The cells go very quickly from *active* to either *loaded* or *exposed* state. A fraction becomes inactive (state *resting*) but is immediately brought to another state by the antigenic stimulation.
Some of the possible states for the APC (cfr table 2.2) are not shown because they are not *stable* (the definition of "stable" state is the one adopted for the B cells). A counting between time steps would show nothing but zero for them.
Note that the small decrease in the total number of cells (in this plot and in others)

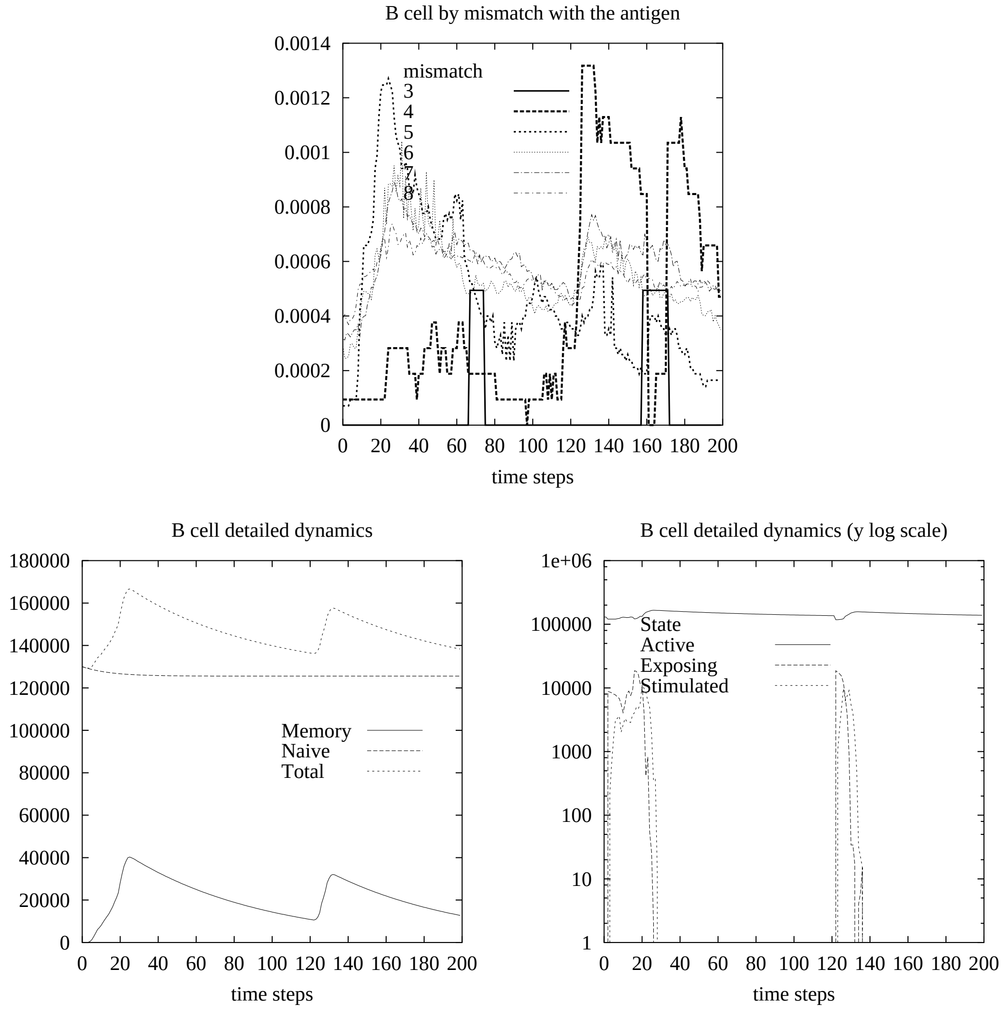


Figure 2.5 The upper plot shows the B cells affinity classes defined by the mismatch with the infecting virus. The number is normalized by the binomial coefficient $\binom{l}{m}$ where m is the mismatch indicated in the key panel of figure. The bottom plots show the population of B cells (memory or naive) in all the possible states.

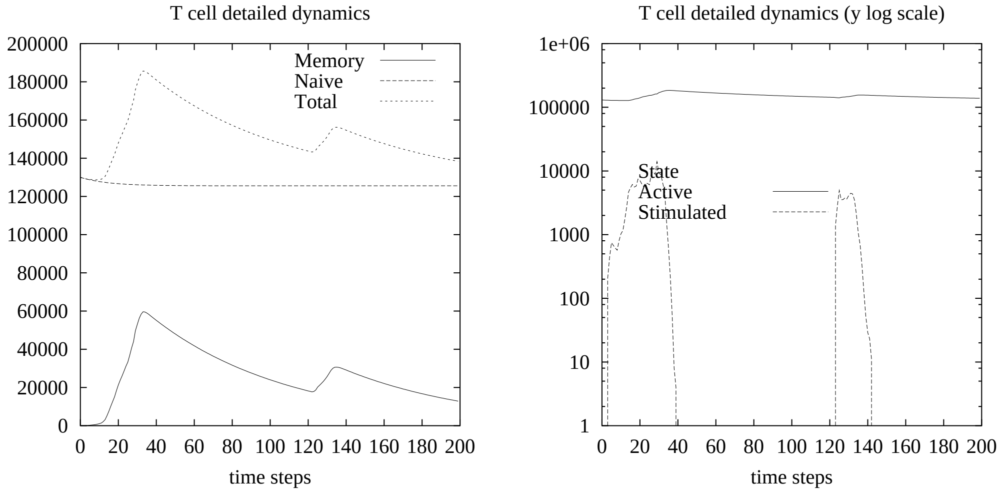


Figure 2.6 Population of T helper cells (memory or naive) in all the possible states. The growth follows the antigenic stimulation. The peak is reached few time steps later compared to the B cells (cfr fig.2.5) because the stimulation depends also on the concentration of interferon-γ (cfr fig.2.4).

is due to the normal birth-death ratio which sets the stable point at a lower value compared with the initial one. This is an effect of numerical approximations.

The plot at the bottom of panel figure 2.7 shows the evolution of the EP (virus-target) cells. The global number decreases because the antigens start to kill them (approximately at the 15^{th} time step). Actually, many EP cells go immediately to the *loaded* state. These are the infected cells. Their number decreases because either the inside proliferating virus or a T killer cell kill them. The total number of cells does not decrease significantly during the second response because the IS is able to recognize the virus in time preventing further infections (*immunization*).

Figure 2.8 shows the evolution of the T killer population. The highest peaks are at time step 20 and 125, this means that there is a delay with respect to the antigenic stimulation. The Immune System needs more time to mount a *first* cellular response (3 time steps to start and about 20 to become strong enough). During the second response the reaction is much faster. The T killer stimulation needs, as in the T helper case, the presence of interferon-γ (cfr fig.2.4).

Figure 2.9 shows the Plasma B cells affinity classes normalized with respect to the binomial coefficient $\binom{l}{m}$. As in the B cells case, the mismatch m is equal to the bit string length l minus the Hamming distance between the Plasma B cell receptor and the antigen. It is clear that the humoral response is completely dominated by

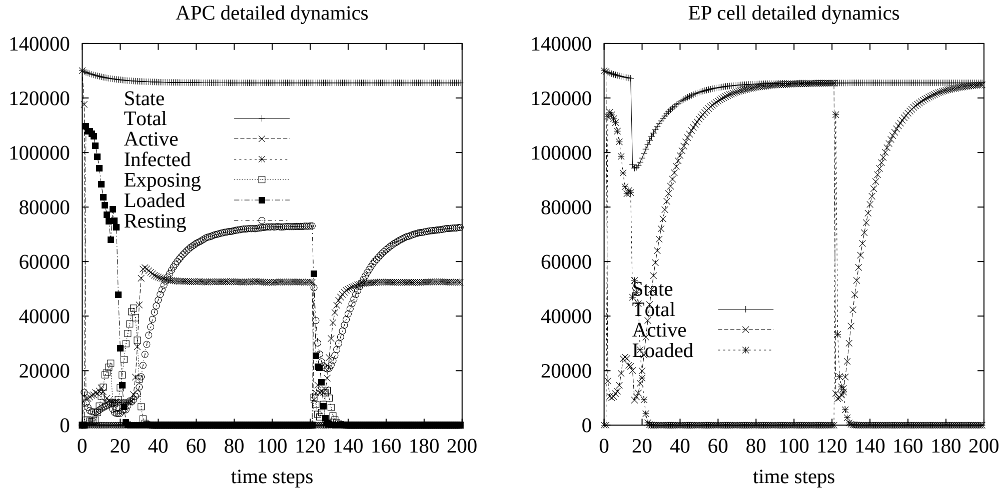


Figure 2.7 On top it is shown the population of APC in the different states. At the bottom the EP population.

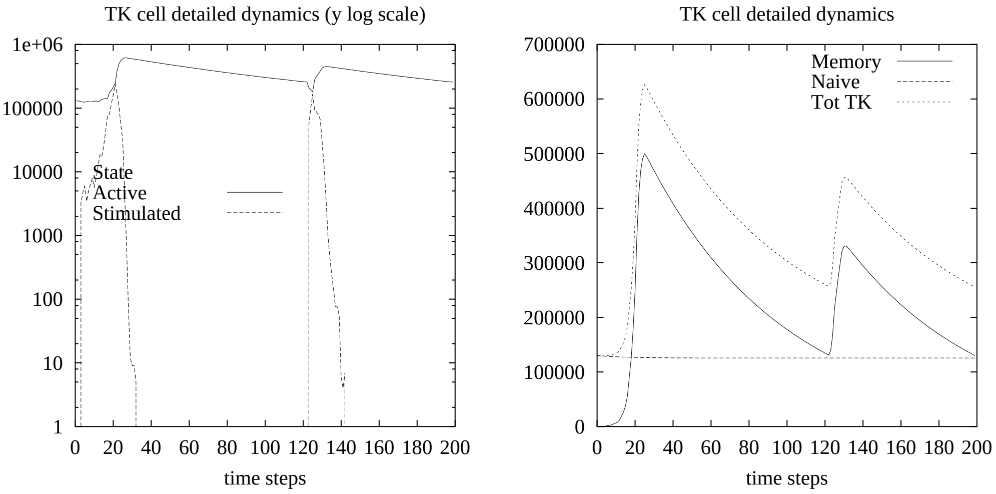


Figure 2.8 T killer cell population detailed dynamics.

high affinity clones ($m = 4$ and 5). The Hamming distance is equal to 19 during the first response. Then it increases to 20 during the second response as a result of the hyper-mutation process. The same remarks apply to antibodies (the bottom plot in figure 2.9) since they are produced directly by the Plasma cells.

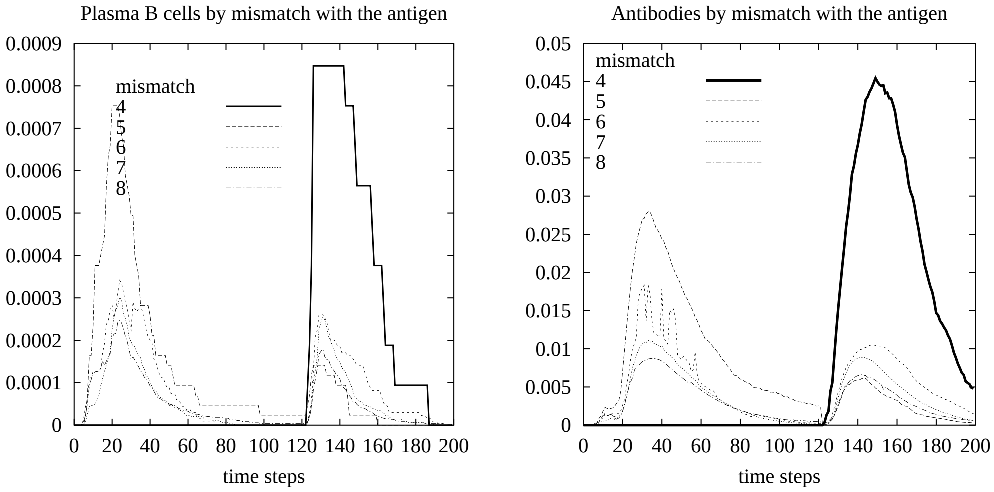


Figure 2.9 Plasma B cells and antibodies by mismatch (normalized to $\binom{l}{i}$).

The figure on the top of panel 2.10 shows the evolution of the immunecomplexes. The huge number of IC in the first response witnesses the massive production of antibodies required to eliminate the viruses in this phase. In the second response the IC population is much smaller because the antigens do not have the time to grow (the antigen is eliminated more quickly).

In the same panel it is shown the Danger signal (bottom plot in figure 2.10). Again, during the first response a large amount of D-signal is released by the EP cells that get killed by the virus.

2.1.9 Notes

The model allows a large repertoire to be represented. This is not a simple “technical” result. For instance, the memory of a past immunization can be thought as a fixed point in the population dynamics of memory lymphocytes [6]. The number of fixed points, which reflects the “general” memory capacity of the system, is more rich and close to the reality if we may represent a large repertoire.

Moreover, another aspect worth to be implemented is the mutation of the antigen (during the replication inside a cell). This, and new interaction procedures

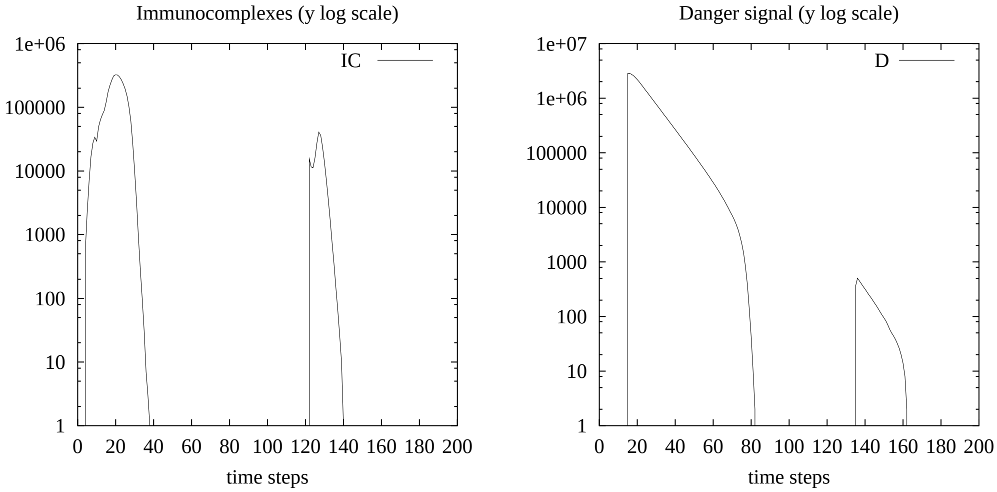


Figure 2.10 Immunocomplexes (top) and D-signal (bottom) both in semi-log scale.

which allow the T helper to be *target* cells of the antigen. The microscopic scenario of mutating viruses which attack the Immune System itself is important because it corresponds to the behavior in case of HIV infection. Using **CImmSim** we could hopefully discover interesting aspects not yet revealed by other models developed for this purpose (see [90]).

2.2 Learning cascade

In the recent past, the simulation of the Immune System (IS) has been drawing significant benefits from the resort to *cellular automata (CA)*, namely fully discrete dynamical systems evolving according to boolean rules [113, 56]. CA appear particularly well suited to the simulation of biological systems mainly on account of their capability to naturally incorporate complex non-linearities. In addition, owing to their space-time locality, they are almost ideal candidates for massively parallel processing.

In the following paragraph we use **CImmSim** to investigate this phenomenon by means of computer simulations. The cellular response triggered by the presence of lymphocytes T killer (Tk) has been turned off, thus only the humoral response is considered.

This work represents an attempt to frame the immune system response to the *Information theory* of C.E. Shannon [107, 60]. It is suggested that the process by which the immune system learns how to recognize foreign invaders proceeds through a cascade of "metastable states" behaving like collective modes in a bit-matching space.

2.2.1 Collective dynamics in the immune system response

For the present study, the affinity potential is chosen in the same form of the truncated exponential of eq(2.1). Here υ_c is equal to 0.05, m is the number of matching bits and m_c is the "cut - off" match below which no recognition takes place. In addition, each cell is endowed with a set of internal degrees of freedom specifying its internal state (e.g. inert, stimulated, Ag-processing, etc, as in table 2.2). Full details on the system specification are given in [64].

Based on a set of computer simulations, we have come up with the following picture of the immune system response. Antigens injected from time t_0 onwards start to interact with a random background of B-cells, distributed along a Maxwellian

$$M_0(m, \mu_0, T_0) = (2\pi T_0)^{-1/2} \exp\left[-(m-\mu_0)^2/2T_0\right]$$

centered about $\mu_0 = l/2$ with variance $\sigma_0 = \sqrt{l}/2$. Here $T_0 = \sigma_0^2$ is the "temperature" measuring the scattering (uncertainty) around the mean value μ_0. Subsequently, after a given induction period, selective Ag interactions with B-cells, lying in the tail of M_0 with matchings above m_c, trigger the growth of a new population of high-match B-cells centered about a higher matching number $\mu_2 = m_c$. This growth proceeds via stimulation of B-cells by Th-cells and subsequent proliferation via clonal multiplication. This is the start-up of the learning process: B-cells peaked about m_c trigger, in turn, the growth of higher-match populations, in a sort of upward cascade in m space ending up with the highest available bit-match number $m = l$.

Such "bump-in-tail" distributions are often encountered in physics where they are usually associated with instabilities ensuing from their high energy/entropy content. In the immunological context, however, there is no thermodynamic principle forcing the release of the entropical excess associated with the "bump-in-tail". On the contrary, the system dynamics is presumably geared towards a negative entropy production feeding the learning process that allows the IS to recognize foreign invaders. Hereafter, we are going to show that a quantitative measure of this learning process is provided by its *relative* entropy.

To be more specific, let us consider a dynamical process turning state 1 into state 2, characterized by distributions f^1 and f^2 respectively. The quantity G_{12} defined as

$$G_{12} = \sum_m f_m^2 \log\left(\frac{f_m^2}{f_m^1}\right) \tag{2.2}$$

is the *relative* entropy or *Kullback information* of the process [38].

Owing to the inequality $\log x \geq (1 - 1/x)$ (equal sign applying if $x = 1$), it is readily shown that G_{12} is positive definite and zero just in case the two distributions are exactly the same. According to [12], G_{12} represents the *information gain* encoding the extra-knowledge associated with the availability of an additional distribution function. We observe that G_{12} may be related to a metric measuring the distance between distributions f^1 and f^2 in a suitable information space [10].

We shall consider the information gain associated with the transition from an initial state f^1 to a final state f^2, identified by their first three moments:

$$n_i = \sum_m f_m^i, \qquad n_i\mu_i = \sum_m m f_m^i \quad \text{and} \quad n_i T_i = \sum_m f_m^i (m - \mu_i)^2$$

(n_i is the number of individuals belonging to f^i, with $0 \leq n_i \leq 1$).
The total entropy of the "bound-state" $(f^1 + f^2)$ is given by

$$H_{12} = H_1 + H_2 + H_X$$

where

$$H_i \equiv \sum_m f_m^i \log f_m^i \quad (i = 1, 2)$$

are the entropies of the "pure" states f^i and

$$H_X = \sum f_m^1 \log(1 + f_m^2/f_m^1) + \sum f_m^2 \log(1 + f_m^1/f_m^2)$$

is the exchange entropy due to superposition of the two states.
Using Maxwellian states as interpolants, simple algebra on eq(2.2) yields

$$G_{12} = n_2 \log \frac{n_2}{n_1} + \frac{n_2}{2}(\log \theta_{12} + \theta_{12}^{-1} - 1 + \delta_{12}^2) \tag{2.3}$$

where $\theta_{12} = T_1/T_2$ and $\delta_{12} = (\mu_2 - \mu_1)/\sqrt{T_1}$.

Such simple formula calls for a number of comments. The term $\log \theta_{12}+\theta_{12}^{-1}-1$ on the rhs of eq(2.3) is the "thermal" component of the information gain, namely the information gained through a differentiation of the two states via a temperature change (scale dilatation/contraction in m space). It is positive definite and vanishes only for $T_2 = T_1$.

The other term on the rhs of eq(2.3), δ_{12}^2, is the information gain associated with differentiation via shifts in m space and consequently it shows *explicitly* the desired dependence on the mean displacements we were looking for. It is also positive definite, regardless of the sign of the displacement δ_{12}, and symmetric under the exchange $1 \leftrightarrow 2$ so that it can serve as a metric distance between f^1 and f^2.

How do these notions map out onto the immune system response? The information gain G_{12} alone can not tell the whole story because its translational invariance does not allow to distinguish between "smart" (high μ) and "dumb" states (low μ). Thus, this indicator has to be complemented with the sign of the mean matching separation δ_{12}. In other words, the sign of δ_{12} indicates whether the system has moved uphill (learning) or downhill (unlearning) along the learning landscape, the module of information gained/lost in such process being given by G_{12}. Within this picture, G_{12} is naturally interpreted as the information cost associated with the process yielding a learning amount $|\delta_{12}|$.

These considerations allow us to gain a better insight into the actual results of the numerical simulations. The runs have been performed with the following parameters: $l = 12, m_c = 10$, grid size = 16 × 15, $\upsilon_c = 0.05$, initial number of *B*, *Th* and *APC* cells equal to 2000. The Antigens are continuously injected at a rate of 300 units/step. All the input values are drawn from [29, 106] which represents the basic reference with respect to the biological parameters of the model. The total number of Antigens and B-cells, as a function of time, is represented in figure 2.11. The B-cells succeed to level off the *Ag* content after about 50 time steps. Since a single time step covers $1/10$ of a typical B-cell lifetime, this corresponds to about 2 weeks in physical units. The dramatic drop of Antigens after $t = 50$ is a clear clue that the IS has been capable to mount a very effective response to the invading agents.

More insight on the specific carriers of the IS response can be gained by inspecting the time evolution of the B-cell distribution function $f_m(t)$ reported in figure 2.12. Here f_m is the total number of B-cells with matching number m versus the total number of B-cells in the system.

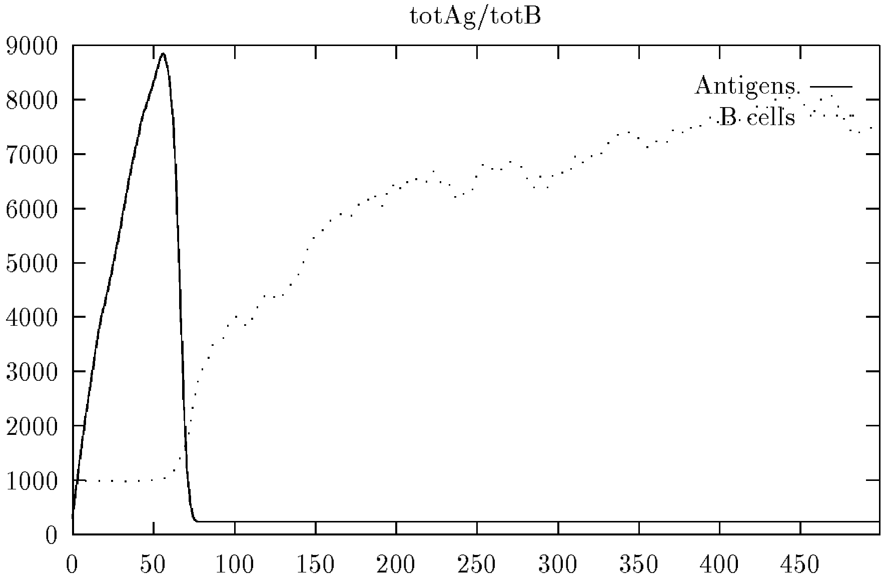


Figure 2.11 The total number of Ag and B-cells as a function of time

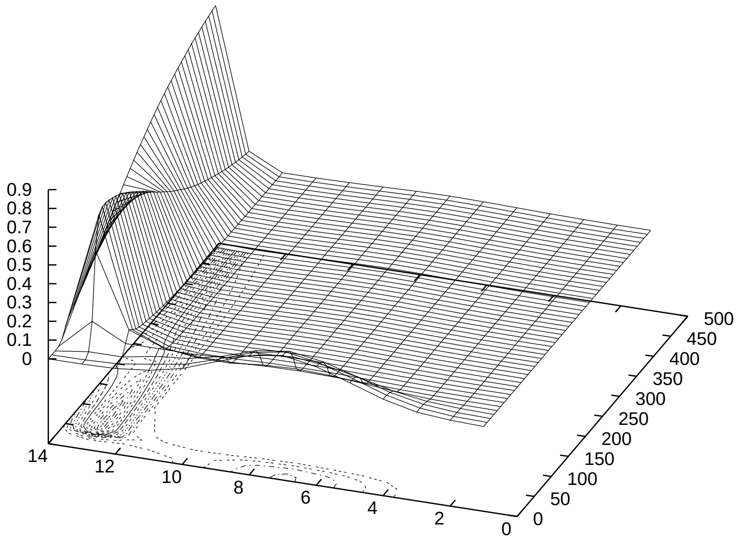


Figure 2.12 The B-cells distribution $f(m,t)$ as a function of time

At the beginning, only the initial Maxwellian corresponding to the mean bit matching number ($\mu = l/2 = 6$) develops. Subsequently, this Maxwellian sends individuals to a second mode, say M_{10}, corresponding to the lowest matching number ($\mu = m_c = 10$) recognizable by the system. The M_{10} mode behaves like an ordered (low temperature) metastable state, being fed by the tail of the M_6 and sending itself individuals to the upper-lying states. In a sense, it serves as an intermediate bridge to accomplish a *learning cascade process* taking $\mu = 6$ states into the final $\mu = 12$ (perfect learning) state.

Several aspects of such process deserve particular attention, like the dependency (if any) on the bit-string length and on the specific form of the affinity potential. For longer strings ($l > 12$), we expect the number of learning stages to increase as $(l - m_c)$. Unfortunately, this hypothesis is hard to test due to the exponential complexity ($O(2^{2l})$) of the model. Any additional bit in the string requires a four-fold increase in computing time.

An interesting feature of the learning cascade is that it is realized via shifted "bump-in-tail" states, which stand out as "collective modes" of the immune system dynamics. This is to be contrasted to an alternative scenario whereby the initial Maxwellian would develop long (exponential or algebraic) tails rather than narrow bumps at high-m.

It is worth to point out an amazing analogy with the mechanism of "current drive" in fusion plasma, namely the dramatic rise of electric current triggered by injection of even minute amounts of radio-frequency power in a range of frequencies much higher than the mean electron speed [118].

Formally the analogy proceeds by identifying B-cells with electrons interacting with Antigens (photons) via the affinity potential $v(m)$ (electron-wave potential). Within this analogy, the mean bit-matching number μ can be seen as the electric current driven by the waves, perfectly in line with the interpretation of m as a fictitious "particle speed". In this context, it would be interesting to define a sort of learning efficiency as the analogue of electric conductivity, namely the ratio between the mean bit-match μ and the strength of the affinity potential v_c.

In figure 2.13 we show the normalized mean match $\mu_n = \frac{\mu(t)-\mu(0)}{\mu(0)}$, ($\mu(0) = l/2$), the entropy $H(t) = \sum_m f_m \log f_m$ and the information gain $G(t)$.

The mean bit match number is the most immediate indicator of whether or not the system is learning to withstand the Antigens attack.

As expected, after an induction time of about 50 time units, the mean bit match exhibits a sharp rise associated with the onset of the $\mu = 10$ mode. This is the time it takes the system to develop the catalytic growth of B-cells lying in the tail ($m \geq m_c$) of the initial Maxwellian. It is also the stage of substantial learning. The subsequent evolution shows a progressive improvement due to the disappearance of M_{10} in favor of the "smarter" M_{12} mode. Since the M_6 is continuously sustained

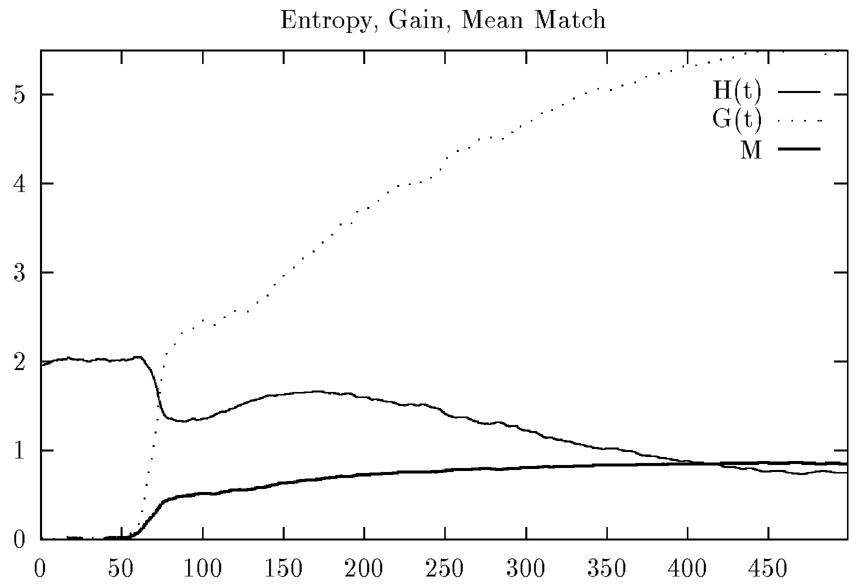


Figure 2.13 Time evolution of the normalized mean match μ_n, standard entropy $H(t)$ and information gain $G(t)$

from the exterior, it never dies out completely thereby preventing μ from achieving the top value $\mu = 12$ (perfect learning state).

After a stagnation period (up to $t \simeq 50$), the entropy undergoes a sudden drop as a result of the IS primary response. Note, in fact, that the statistical dispersion of M_{10}, T_{10}, is significantly smaller than T_6. Subsequently, the simultaneous presence of competing modes causes a slight entropy increase, mainly contributed by the exchange entropy component. Finally, as the mode centered in $m = 12$ prevails, the entropy starts again to decrease monotonically. As a general remark, we observe that the entropy $H(t)$ does *not* behave like a proper H-function, i.e. a monotonically decreasing/increasing function of time. Since our f_m is a standard probability density function (i.e. positive definite and normalized to one), our interpretation is that no standard H function can be associated to the C-S automaton dynamics.

Finally, we inspect the evolution of the information gain $G(t)$. As expected from previous analytical considerations, $G(t)$ proceeds much in sympathy with the mean match number $\mu(t)$. However, the sharp rise around $t = 50$ is significantly steeper, taking almost the connotations of a first order phase transition.

A semiquantitative assessment of the time evolution of $G(t)$ may be attempted on the assumption that the bump-in-tail modes behave like Maxwellian distributions. We want to stress that, due to the limited string length ($l = 12$) this is no more than a reasonable hypothesis. In other words, we have been able to test it just for the initial Maxwellian M_6. For higher-match modes longer strings are required. Nevertheless it is reasonable to state that the specific shape of these modes should not affect the qualitative features of the learning cascade.

By identifying state 1 with M_6 and state 2 with M_{10}, from figure 2.12 we infer $n_1 = 1$, $n_2 = 0.63$, $T_1 = 3$, $T_2 \simeq 0.4$, $\delta_{12} = (10-6)/\sqrt{3}$. According to eq(2.3), this yields $G_{12} \simeq 1.75$ in a satisfactory agreement with the data of figure 2.13. By modeling the subsequent evolution as a transition from M_{10} to M_{12}, we obtain $n_1 \simeq 0.63$, $n_2 \simeq 0.85/2$, $\theta_{12} \simeq 1$, $\delta_{12}^2 \simeq (12-10)^2/0.4 = 10$. Since M_{12} lies on the rightmost boundary of bit-matching space, we must account for finite-size effects. Some algebra yields

$$G_{fs} = G/2 + \frac{n_2}{2}\frac{T_2}{T_1}\delta_{12}/\sqrt{2\pi T_2}$$

where the subscript *fs* stands for 'finite size'. The final result for the transition M_{10} to M_{12} is therefore $G \simeq 2.8$, again in a reasonable agreement with the results of the numerical simulation.

As a further observation, we note that, at variance with standard entropy, the information gain *does* behave like a proper H -function, namely it monotonically increases with time. This is conducive to the idea of a "maximum information-gain principle" [99] of the form $\frac{dG}{dt} \geq 0$ with the equality sign holding when the learning process is basically over. This is a direct consequence of G being a monotonic function of δ, which is quite reasonable in light of the interpretation of G as the information cost associated with the learning amount δ.

In turn, δ is a monotonic function of time because high-*m* bumps develop as a result of the depletion of lower-*m* "parents", hence after them. This is why the system dynamics exhibits a "built-in" time-arrow.

The present study sets a pointer in the direction of kinetic-theory (Boltzmann) as a valuable approach to the IS dynamics, possibly achieving an optimal compromise between CA microdynamics and macroscopic population dynamics.

2.3 The role of affinity potential and hypermutation in the cascade

In the previous section the sequential nature of the process allowing the Immune System to learn how to withstand pathogen agents has been explored by means of large-scale computer simulation of the Celada-Seiden immunological automaton. The question was: how does the IS learn how to mount a specific response against invading entities? This capability of the IS is rooted in the specific functions of each of its micro constituents, but it is also true that the way the IS as a whole learns to withstand antigen attacks depends even more on the mutual interactions between these micro constituents. Once the importance of collective behaviour is acknowledged, a relevant question becomes whether non-equilibrium statistical mechanics and the theory of (non-linear) dynamical systems, provide a convenient mathematical framework to characterize, at least semiquantitatively, the generic features of the immune system response [95, 65].

Within the CS-model, each cell is characterized by a *bit matching number* m denoting the number of matching bits with the bit string representing the Antigen. Bit match when they are complementary ($0 \leftrightarrow 1$). High / Low affinity is therefore to be understood as high / low values of the matching number m. With a string length l, the *repertoire* of the model is best organized into a hierarchical set of $l+1$ classes of cells characterized by the *matching number* $m = 0, 1, 2 \dots l$. The generic m^{th} class contains

$$\binom{l}{m} = \frac{l!}{m!(l-m)!}$$

elements; the sum over all possible classes involving a total repertoire

$$\sum_{m=0}^{l} \binom{l}{m} = 2^l$$

possible specificities. This defines the internal shape space of the automaton. The *active region* (defined later) has a structure markedly pyramidal: only one state with perfect match $m = l$, l states with $m = l - 1$ and so on down the line.

The population density in this phase space is given by the *actual* occupation number $N_m(x, t)$ of cells in class m at site x at time t. Knowledge of the occupation numbers N_m at each space-time location yields a complete characterization of the dynamical system. The B and T-cells binding affinity is expressed via an *"affinity potential"* $\upsilon(m)$. The specific form of the affinity potential is not known in detail from biological data, but it is plausible to express it in the form of a sharply increasing function of m above a critical cut-off $m_c < l$, and zero below it (see eq(2.1)). As a result, only a subset of the phase space, characterized by the condition $m_c \leq m \leq l$, is immunologically active. We shall call this the *active region* of immunological phase-space, whose size $l{-}m_c{+}1$ counts the populations competing for the Antigen.

Numerical evidence was reported in [119] that the IS learning process proceeds through a cascade of higher and higher affinity populations (B-cell) in which the low affinity modes indirectly feed the higher affinity ones in a kind of sequential process dubbed *learning cascade*. A quantitative indicator of the aforementioned learning cascade was identified with the Kullback relative entropy, or information gain [38], defined as in eq(2.2). where we refer to a transformation taking the system from initial state "1" at time $t = t_1$ to final state "2" at time $t_2 = t_1 + \tau$, and

$$f_m = \frac{N_m}{\sum_{m'} N_{m'}}$$

is the probability density of class m. Ordinary Boltzmann entropy did not show any sign of monotonic behaviour, which is not surprising since there is no reason to believe the underlying micro dynamics of the CS automaton should obey a Boltzmann H-principle.

In accord with the theory of clonal selection, the learning process is basically a shift of the occupation numbers towards the high-affinity region of the spectrum, a bias towards "smart" individuals. A prime indicator of this shift is an increase in time of the average matching

$$\mu(t) = \sum_{m,x} m f_m(x, t)$$

where x runs over the spatial extension of the system. The gain can be interpreted as the information-cost needed to bring the IS from low to high-affinity states, and under certain assumptions on the shape of f_m it can be expressed as an analytic function of μ [119].

Our previous results pertained to relatively small repertoire, with $l = 12$, consisting of $2^{12} = 4096$ specificities. This is about four orders of magnitude lower than the *expressed* repertoire of the human Immune System. The question, which makes the hard-core of this paper, is whether the generic features observed in our previous work do survive once larger repertoire are considered. To this purpose, we have upgraded our computational tool to take full advantage of *parallel computing* capabilities [26]. Specifically, we have extended the size of the repertoire from 2^{12} up to 2^{20}, namely more than *two* orders of magnitude above our previous work, more than an order of magnitude beyond any previous study with the CS-model we are aware of, and, more importantly, *only an order of magnitude* below the expressed repertoire of the real IS. Being aware that a mere rise in the size of the repertoire does not necessarily imply a corresponding gain of immunological fidelity, we have also included hypermutation, namely the mechanism by which clones that differentiate from their mother cells, may show point mutations in their receptors [123, 30].

In the model, hypermutation is represented by a given string s turning into a different string s' as a result of one (or more) bits changing state (zero to one, one to zero). The qualitative effect of hypermutation is to generate cells which would not appear otherwise in the system, thus giving the IS more freedom to explore its phase space. The chief question is whether such freedom is used to help affinity maturation, and, if so, to what extent. This question is genuinely dynamical in nature. On the one side, affinity-degrading (high-to-low) mutations are more likely than affinity-enhancing (low-to-high) ones simply on account of the pyramidal structure of the active region of the phase space. On the other hand, since high-match cells are more effective in capturing the Antigens, once generated, they get a chance to reproduce faster than all other competing cells and possibly promote the affinity maturation. Whether such a chance does indeed materialize in actual practice is a non-trivial question which involves a genuinely dynamic non-equilibrium process. Computer simulation is well placed to provide a semi - quantitative guidance in this complex territory.

2.3.1 Numerical simulations

We have performed a series of numerical simulations by varying the string length l, with and without mutation, and the shape of the affinity potential. The simulations are performed on a 16×16 grid, with the following parameters: average lifetime of B cells, $\tau_B = 10$, initial population $B(0) = 2184$, birth-rate $\dot{B} \simeq 0.07B(0)$. The Ag are injected at a rate of 1000 individuals per time step. Each time step corresponds to about 8 hours in real time. Finally, we assume a time independent single-bit mutation rate equal to $p_b = 0.02$. We do not address the issue of *optimal* mutation schedule as in [58]. Our observations are based on a series of simulations with $l = 20$, $m_c = 15$ with and without mutation, and two typical shapes of the affinity: A) Convex, B) Concave.

The values of the affinity functions are reported in Table 2.5:

Each simulation has been performed 40 times with different random seeds to double-check possible dramatic differences in the outcomes. Although no quantitative conclusions can be drawn, we can state that there are no indications of a strong sensitivity to the choice of the random numbers.

Our data show that affinity maturation does take place and in all cases *it proceeds through a sequential cascade from low to high affinity populations* (see fig. 2.15).

Since no virgin B-cells above m_c are allowed, the appearance of active cells in the course of time is necessarily due to hypermutation. The conclusion is that, albeit penalized in the average, hypermutation does have a dramatic effect in pro-

m	VA	VB	VB/VA
15	0.05000	0.05000	1.000
16	0.09103	0.50072	5.501
17	0.16572	0.74829	4.515
18	0.30171	0.88428	2.931
19	0.54928	0.95897	1.746
20	1.00000	1.00000	1.000

Table 2.5 Affinity potential for convex and concave shapes

moting affinity maturation. The intuitive picture is that, once a favourable mutation occurs by fluctuation, the higher reactivity of the high-affinity cells allows them to reproduce and survive for a long time.

In all cases, affinity maturation ramps up between $t = 50$ and $t = 100$ time units, that is between two and four weeks respectively in real time. During this burst, the total affinity, defined as

$$\hat{v}(t) = \sum_m N_m(t) v(m)$$

grows by almost two orders of magnitude, or more, depending on the shape of the affinity potential, as shown in fig. 2.14. This burst is followed by a slower but steady growth associated with an increasing fraction of high-affinity cells.

Runs without hypermutation (see fig. 2.16) also show a learning cascade, actually faster than with hypermutation. This means that the chance of generating active cells ($m \geq m_c$) by a mutation event is smaller than the corresponding chance of generating it out of the binomial distribution of virgin cells in the hypermutation-free scenario.

Given this intrinsically transient scenario, it is useful to develop a semi - quantitative rationale for the role of the shape of the affinity potential.

To this purpose, let us consider all matching bits of a bit string s (whose length is l) as part of a substring g (“good” bits) and all other bits of s as part of a substring b (“bad” bits). String g has length m ($m \geq m_c > l/2$). Obviously, b’s length $\overline{m}$ is equal to $l - m$.

By definition, mutations on b enhance the affinity whereas mutations on g decrease it.

To compute the total probability of increasing the Hamming distance (i.e. the number of $0 \leftrightarrow 1$ matchings) of n units, we must take into account all possible combinations of mutations in the b and g strings such that $k - j$ is equal to n. Here k and j are, respectively, the number of mutations in the b and g string and $j < k$ so that $n > 0$. The expression of such total probability is:

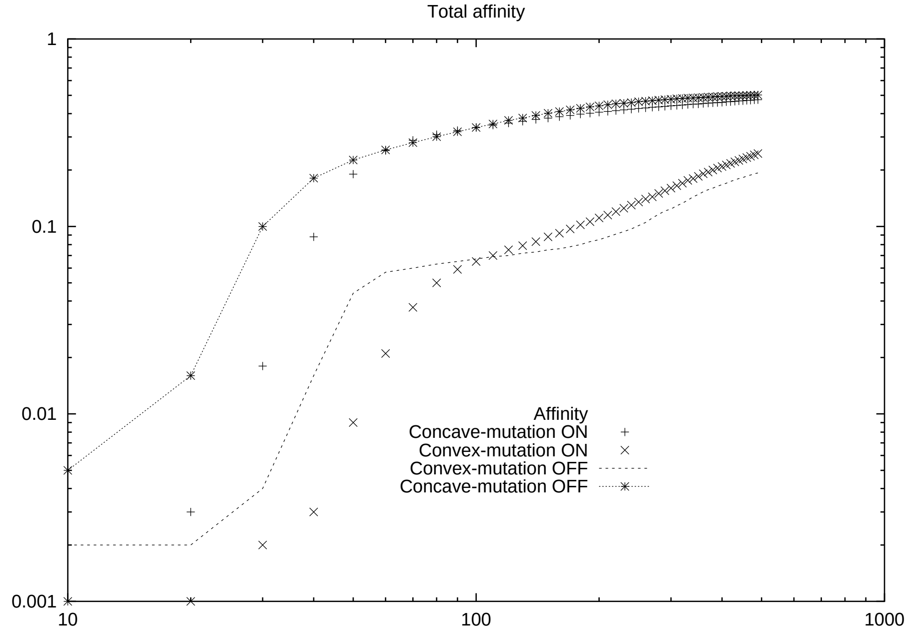


Figure 2.14 Total affinity as a function of time

$$P_n^+ = \sum_{j,k:k=n+j} \binom{m}{j} p^j q^{m-j} \binom{\bar{m}}{k} p^k q^{\bar{m}-k} \tag{2.4}$$

The block $\binom{m}{j}p^j q^{m-j}$ gives the probability of j mutations in the g string, whereas the block $\binom{\bar{m}}{k}p^k q^{\bar{m}-k}$ gives the probability of k mutations in b. Upon using the relation $k = n + j$, we may recast eq(2.4) in terms of the index j, which runs between 0 and $\bar{m}-n$ (otherwise we would have a degrading mutation). Consequently, we can write:

$$P_n^+ = \sum_{j=0}^{\bar{m}-n} \binom{m}{j} \binom{\bar{m}}{n+j} p^{n+2j} q^{l-(n+2j)} \tag{2.5}$$

The probability of affinity degrading mutations, P_n^-, is obtained by considering $j > k$, swapping m with $\bar{m}$ in the above expression and letting the index to run from 0 to $\bar{m}$.

One minute's thought reveals that, since $m_c > l/2$, under the standard condition $m > \bar{m}$, $P = P_n^+ - P_n^-$ is negative, reflecting the intuitive idea that in the average, mutation works against high-affinity cells.

It is now instructive to observe that for low mutation rates $p_b << 1$, the summation in expression 2.5 can be replaced by its first order term. This leads to a very handy expression for the total one-bit affinity improving mutation rate

$$P_1^+ \simeq \bar{m}p(1-p)^{l-1} = (l-m)p(1-p)^{l-1} \tag{2.6}$$

showing that, within this approximation, the one-bit improving mutation probability decays linearly with m.

Coming back to the interpretation of our results, the picture is as follows. Once an active cell materializes, the chance to capture an Antigen and rapidly duplicate can be estimated as $A_m = 1 \times \upsilon(m)$, that is one cell, times its microscopic affinity $\upsilon(m)$. How many cells should materialize before the duplication process is actually triggered? The condition is of course $N_m\upsilon(m) \geq 1$, which sets a natural threshold $1/\upsilon(m)$ for the affinity maturation process to take off. This threshold would of course favour high m's, were it not for the strong penalty set by the mutation probability: a direct jump of n matching numbers ahead, scales roughly like p_b^n, which means that the next matching number above m_c is picked up by the condition $N_m\upsilon(m)p^{m-m_c} > 1$, associated with a critical threshold $N_m^c \simeq p^{m_c-m}/\upsilon(m)$. It is easily seen that N_m^c is a sharply decreasing function of m, unless $\upsilon(m)$ would grow faster than the decrease of $p_b^{m-m_c}$, not a plausible assumption given the small value of p_b. This explains the sequential nature of the learning cascade.

The next question relates to the mid-long term dynamics of the response. Here two competing effects must be balanced.

The probability of cells' clonation is proportional to $\upsilon(m)$. A stimulated cell duplicates every step during four steps after stimulation, yielding 16 clones in four steps. This exponential growth is contrasted by a mean hypermutation loss proportional to P. No way for mutation to compete with such exponential growth in the short term.

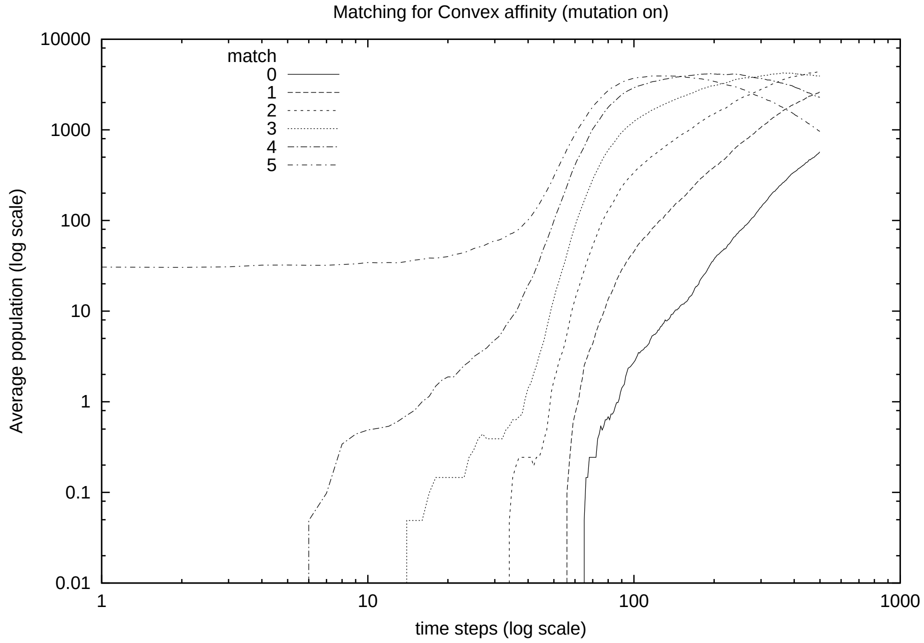
It is plausible to assume that the long-term winner is selected by the condition of maximizing $N_m\upsilon(m)/P$, which leads to a second threshold, $N_m^{c,2} \simeq P/\upsilon(m)$. Now, using the simplified expression P_1^+ given by 2.6, it is readily seen that highest affinity modes are favoured unless $\upsilon(m)$ grows slower than P_1^+, i.e. linearly, with m. This supports the intuitive idea that convex potentials favour the development of high-affinity populations as the asymptotic carriers of the IS response.

Of course, the notion of asymptotic, long-term carriers, although interesting from the point of view of statistical mechanics (final attractor of the system) is not necessarily the most relevant one to immunological purposes. To this end, one is probably more interested in the short and mid-term (days-to-weeks) dynamics of the total affinity $\hat{\upsilon}(t) = \sum_m N_m(t)\upsilon(m)$.

This is shown in fig. 2.14 for the two different choices of the affinity potential. From these curves we see that, notwithstanding the significant statistical fluctuations in the initial phase, indeed the concave potential yields the quickest and most intense response, especially within the ramp-up period, up to $t \simeq 100$. This is obviously due to the much higher affinity of the B-cells (see fourth column in Table 1). Asymptotically, however, our data suggest that the convex shape might be able to recover due to the emergence of perfect match cells with $m = l$ suffering less competition with other cells as compared to the case of a concave potential.

The above considerations, albeit still semi - qualitative, provide a sound background for the interpretation of affinity maturation as a cascade process in affinity space. They also show that the learning cascade is quite robust vis-a-vis the shape of the affinity potential. This latter, however, plays a central role in the short and mid-term dynamics of the IS response.

In closing, a few considerations on computational performance are in order. The parallel simulator (see appendix B) takes about 10 ms / step per grid-point, corresponding to about $10,000$ seconds elapsed time for a 500 step long (about 160 days of real life) simulation on four processors of an UltraSparc Enterprise 4500. Memory requirements peak at about 2 GBytes during the burst of affinity maturation. These figures prove that the numerical investigation of the Immune System response via the Celada-Seiden automaton requires substantial amounts of computational resources.



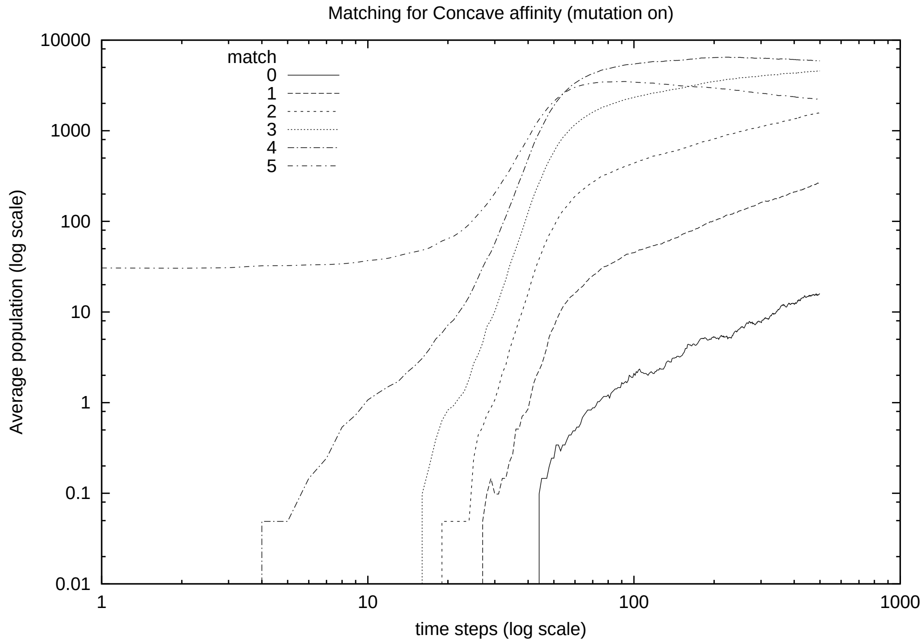
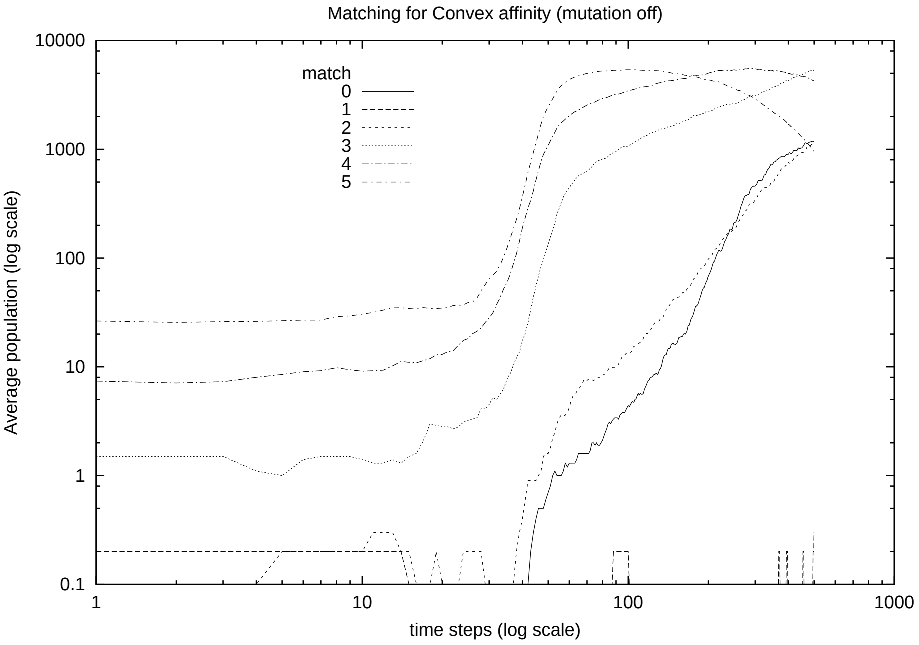


Figure 2.15 Population growth for affinity classes of the active region for string length of 20 bits (details in the plots' titles).



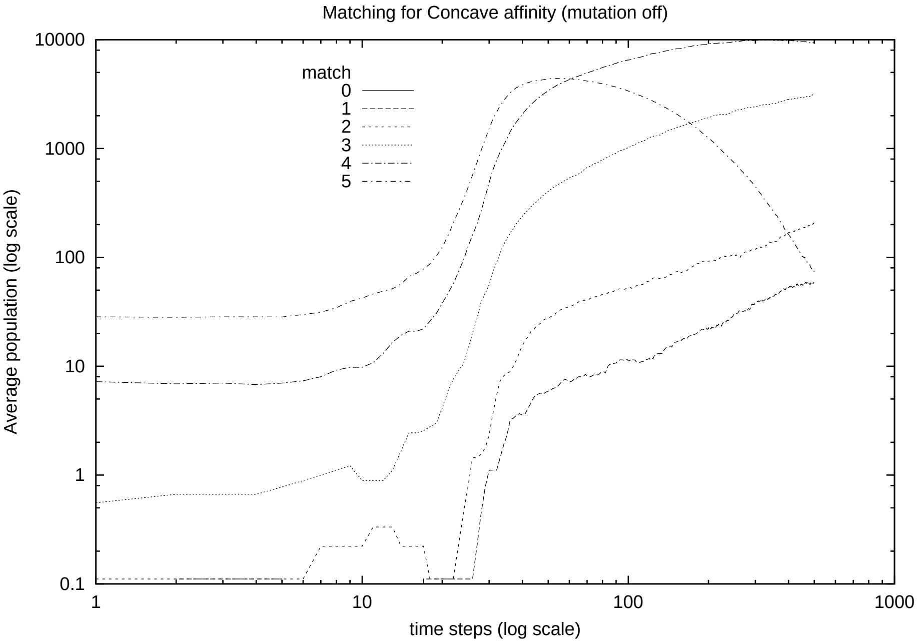


Figure 2.16 Same data as in fig 2.15 for runs without mutation.

Chapter 3

Antigen recognition and evolution

The study of the learning cascade in section 2.2 and 2.3 has given a qualitative assessment about the role played by the affinity potential $\upsilon(m)$ and the hypermutation in the learning cascade which drives the IS toward the recognition of a foreign antigen.

We present here a simple model of selection and mutation of cell populations which is based on generic assumptions and in this respect it resembles a model for evolution in simple ecosystems. In this case the selection acts by means of the same affinity function of eq(2.1) representing the ability of a certain class of lymphocytes to recognize a foreign antigen. Carrying out some analytical and numerical studies we determine the critical values for the parameters ruling the selection and mutation.

Starting from the simple definition of bit string representation in complex biological system originally given in [42] for the immune system modeling, we developed a simple spatial *mean field* approximation of the clonal expansion and affinity maturation in the IS. We carried out some analytical studies on its dynamics and also developed a numerical resolution of the corresponding discrete model.

The presented model works on general assumptions of mutation and selection which are the basis of any model for Darwinian evolution in simple ecosystems. In this respect it resembles the Eigen model [37] for species formation, further studied in [9]. They consider the evolution of an infinite set of self-reproducing (i.e. asexual reproduction) molecules that undergo Darwinian evolution. In the presence of a very selective environment the model shows the formation and survival of a cluster of genetically affine well-adapted molecules (called *quasi-species*).

In the following some basic concepts are borrowed from this framework; then, we will make use of terms as *fitness* and *affinity*, or *matching class* and *population*, interchangeably, to keep the metaphor of the evolution always in mind.

3.1 Introduction

The immune system is an ecosystem for its own and the population dynamic is ruled by selection and mutation. In [42] the original idea of representing a molecule by means of a binary bit string was used to investigate the dynamics of the immune system using differential equations. The idea was further extended in a discrete fashion in the Celada-Seiden automaton [106]. The CS-model concentrates a *lot* of biological complexity in a single spatially extended automata system. In that model the *genotype* coincides with the *phenotype*; there is no distinction between the *DNA portion* coding for the receptor (called the *complementarity-determining-region*, CDR) and the receptor itself that will bind the antigen. Both are bit strings of a certain length l.

In our model we focus on the ability of the lymphocytes to recognize the antigen. Then we have that the *genotype-phenotype map* is accomplished indirectly by means of the affinity function υ, which in turns selects highly adapted elements on the basis of their recognition ability for the antigen. This leads to a one - dimensional phenotypic space (integers between 0 and l).

In a previous work [119] we showed how such models could be used to understand the mechanism by which the IS *learns* to recognize the shape of the invading antigen. In that case we identified with the word "learning cascade" the dynamics of the cell population distributed according to the match with the antigen; in other words with a string length l, the *repertoire* of the model is organized into a hierarchical set of $l+1$ classes of cells characterized by a given *matching number*, $m = 0, 1, \ldots, l$, denoting the number of bits of the lymphocyte receptors which match with the string representing the antigen. The generic m-th class contains $\binom{l}{m}$ elements; the sum over all possible classes involving a total repertoire of $\sum_{m=0}^{l} \binom{l}{m} = 2^l$ possible specificities.

Apart from the details of the other cellular entities involved in the process (B and T lymphocytes, Plasma B lymphocytes, Macrophages, Antibodies and so on) which we leave to the bibliography, one may concentrate on the affinity function υ of the match with the antigen to discriminate the good (i.e. recognizing) lymphocytes from the bad ones. This function is the key to push the system toward the proliferation of the recognizing clones of lymphocytes and, thus, can be thought as a generic *fitness* function in a competition landscape, where the goal is the recognition of a certain pattern or set of binary attributes (i.e. the bit string representing the antigen).

Questions on how the function υ influences the dynamics of the response in terms of cell population predominance eventually leading to a perfect match response (i.e. clone with match $m = l$) are addressed here. Moreover, because the mutation phenomena have been demonstrated to be crucial for the IS to generate diversity and cover the unavoidable holes in the native repertoire [30], we consider *point mutations* in the genotype, that is, the possibility that elements with match m produce, during the cloning expansion, some with match m'.

The *error threshold* [9] is determined as the value of the mutation rate above which the fittest population is no longer the dominating one. We make estimation of such value for different bit-string-length systems.

In contrast to [37, 9] where the authors consider unlimited bit string length, we are interested in the real *potential* specificity of the immune system of a mouse [95] and thus up to $\log_2 10^{11} \simeq 36$ bits per string [1].

3.2 The model

The experimental setup reference is that of lymphocyte antigen-specific populations that proliferate under constant supply of the antigenic stimulus in the presence of the helping environments like the signal from the T helper (Th) lymphocytes needed to trigger the response. Also suppose a complete initial repertoire and assume continuous regeneration of dying cells so that we may fix the birth-death rate equal to zero and observe just the growth due to cloning expansion.

The affinity potential and the grow rate. Given $l \in \mathbb{N}$ and $l/2 < m_c < l$, the affinity potential or fitness function is defined over the integer $0, \ldots, l$ in the same way it has already been discussed for eq(2.1) reported here for convenience:

$$\upsilon(m) = \begin{cases} \upsilon_c^{(m-l)/(m_c-l)}, & m \geq m_c; \\ 0, & m < m_c. \end{cases}$$

where $\upsilon_c \in (0, 1)$ is the free parameter (see figure 3.1). Here υ_c determines the sharpness of the fitness (this is the same as in [9], but they do not use any threshold). The fitness summarizes the effect of the recognition ability as well as the reproduction efficiency. This means that high affinity cells proliferate at higher rate than the lower affinities. Note that class $m < m_c$ does not grow, for $\upsilon(m < m_c) = 0$. Also note that for $\upsilon_c \to 1$ the affinity transforms into a flat landscape and there is no preferred genotype for the interacting individuals ($m \geq m_c$); in this case, a vanishing mutation rate leads the evolutive path into the subregion of the state space corresponding to constant population of not-recognizing clones ($m < m_c$) and *equally* distributed, though dominating, populations of recognizing clones ($m \geq m_c$).

---

[1] We should point out the difference between the *expressed* repertoire and the *potential* repertoire; this last is the number of possible receptors that can be constructed given the genetic mechanisms involved

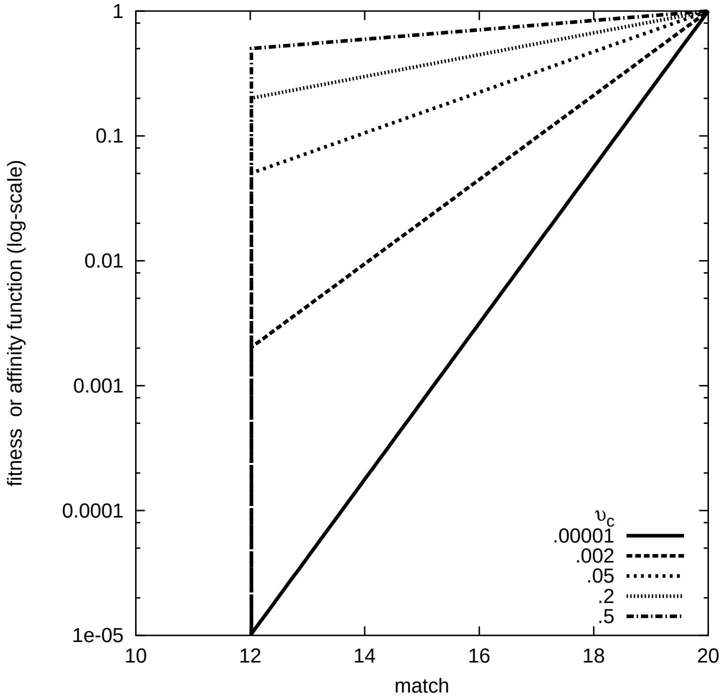


Figure 3.1 The affinity function (y axis in log-scale) or reproduction efficiency (fitness) is a monotonic function of the match with the antigen-target. Matches below the threshold m_c have null fitness score. In figure $l = 20$ and $m_c = 12$ for different υ_c. Small υ_c give sharp peak affinity. For υ_c approaching unity it transforms into a linear fitness and for $\upsilon_c = 1$ the landscape is flat corresponding to an environment which makes only difference between recognizing and not recognizing class.

The mutation process is performed at the *gene* level but is expressed at the *phenotype* level in the form of increased/decreased match with the antigen. Mutations occur during the cloning expansion meaning that only class $m \geq m_c$ are affected. The number of mutated bits is binomially distributed with parameters p_b and l.

To calculate the probability that a string with match m is mutated into a string with match i we subdivide the original string in two substrings G, B composed by matching bits (G good) and non matching bits (B bad); let be $|G| = i$ the actual match and $|B| = l - i$ ($| \cdot |$ denotes the number of bits in the string). Be γ the number of flipped bits in substring G, $0 \leq \gamma \leq i$, and analogously β in B, $0 \leq \beta \leq l - i$. The matrix $\{m_{ij}\}$ gives the probability of a jump from class j to class i by mutation

$(q = 1 - p)$;

$$m_{i \leftarrow j} = m_{ij} = \sum_{(\gamma,\beta):\gamma-\beta=j-i} \binom{j}{\gamma}\binom{l-j}{\beta} p^{\gamma+\beta} q^{l-(\gamma+\beta)} \tag{3.1}$$

Observe that since we take $m_c > l/2$, the mutation is most likely to lower the affinity than to increase it, i.e.

$$\forall k \geq m_c > l/2, \quad \sum_{i=k+1}^{l} m_{ik} \leq \sum_{i=0}^{k} m_{ik}.$$

This means that mutations have no other positive effect other than generating diversity, thus spreading a proliferating class into adjacents, with preference to the lower ones.

A very approximate equivalence [30] between the *per bit* mutation rate p_b and *per base-pair (bp)* mutation rate *in vivo* r, is the relation $r = 1 - (1 - p_b)^{l/180}$ bp. [2]

The model. Given l and m_c fixed, and υ_c and p_b as free parameters, we indicate with $x_j(t) = x_j$ the population of class j at time t. The initial time $t = 0$ corresponds to the *infection* time, and the initial unbiased set of population match-classes is binomially distributed with parameters l and $\frac{1}{2}$. [3] The population dynamics can then be expressed by a set of $l + 1$ linear differential equations (balance equation for population x_j);

$$\frac{\partial x_j}{\partial t} = \sum_i \upsilon(i) m_{ji} x_i - \upsilon(j) x_j \sum_{i \neq j} m_{ij} \qquad \forall j = 0, \ldots, l$$

that is, after some easy manipulations

$$\frac{\partial x_j}{\partial t} = \sum_{i \neq j} \upsilon(i) m_{ji} x_i + \upsilon(j)(2m_{jj} - 1) x_j \tag{3.2}$$

Mutations realize a kind of *cooperation or symbiosis* among populations. There is no *competition* in our model because we assume to have sufficient antigens to feed the proliferation of the clones during the immune response. This is surely the case if we consider the effect [95] of the *Dentridic Cells* that retain the antigen for long periods (weeks).

---

[2] Assuming that $\simeq$ 60 out of 210 amino acids contribute to the V_H and V_L CDR, the number of base pairs is $\simeq$ 180.

[3] This is equivalent to have the initial population of cells with the receptor-strings drawn at random. The probability to choose a single bit as '1' or '0' is $\frac{1}{2}$.

It is important to note that we have excluded any limiting factor in our system; this corresponds to have unlimited *carrying capacity* K, in the Verhulst factor $(1-\sum x/K)$ [85] which sounds quite unrealistic when speaking about the IS. This simplification will be taken out in the following section when we compute numerically the solution. For what concerns the analytical study, we may avoid such complication as we are mainly interested in observing the cloning expansion during the infection and the production of the immune response which clearly last for a finite period (days or weeks), otherwise leading to the death of the host. This translates in observing the dynamics far from the saturation given to the carrying capacity K.

In system 3.2, if we use the properties $\sum_{i\neq j} m_{ij} = 1 - m_{jj}$ and we made the following substitution

$$\pi_{ij} = \begin{cases} \upsilon(j)(2m_{jj}-1), & i=j, \\ \upsilon(j)m_{ij}, & i\neq j. \end{cases}$$

we may write the associated matrix equation as $\partial\mathbf{x}/\partial t = \Pi\mathbf{x}$, where $\mathbf{x}$ denotes the l+1 row vector $(x_0, x_1, \ldots, x_l)$ at time t and $\Pi = \{\pi_{ij}\}$. Note that the first m_c *columns* of Π are null as $\upsilon(j<m_c)=0$.

For analytical manipulation it is convenient to "condense" all the population $x_{j<m_c}$ into a single cumulative population. We will then make the following substitution of variables and consider a number of populations equal to the number of interacting plus one; i.e. class m_c, class m_c+1 and so on, up to class l. Then we indicate with new indices -1 the *absorbing* population, with 0 the *minimum interacting* population m_c, with 1 the *intermediate* or m_c+1 population and so on, until the *perfect* match population l, that will take index $l-m_c$.
In the following we fix the number of interacting population $l-m_c+1$ *equal to 3* so that populations will be indicated by $\mathbf{x}' = (x_{-1}, x_0, x_1, x_2)$. The corresponding values for the affinity potential are $(\upsilon_{-1}, \upsilon_0, \upsilon_1, \upsilon_2) = (0, \upsilon_c, \sqrt{\upsilon_c}, 1)$;
The evolution equation expressed in matrix form is

$$\frac{\partial}{\partial t}\mathbf{x}(t) = \Pi'\mathbf{x}(t) \tag{3.3}$$

where the matrix Π' governing the dynamics is:

$$\Pi' = \begin{pmatrix} 0 & \upsilon_c\mu_{-1,0} & \sqrt{\upsilon_c}\mu_{-1,1} & \mu_{-1,2} \\ 0 & \upsilon_c(2\mu_{0,0}-1) & \sqrt{\upsilon_c}\mu_{0,1} & \mu_{0,1} \\ 0 & \upsilon_c\mu_{1,0} & \sqrt{\upsilon_c}(2\mu_{1,1}-1) & \mu_{1,2} \\ 0 & \upsilon_c\mu_{2,0} & \sqrt{\upsilon_c}\mu_{2,1} & (2\mu_{2,2}-1) \end{pmatrix}$$

The elements $\mu_{i,j}, i,j \in \{-1,0,1,2\}$ can be calculated from the mutation matrix $\{m_{ij}\}$ as it follows

$$\begin{array}{ccc}
\mu_{-1,0} = \sum_{j<m_c} m_{j,m_c}, & \mu_{-1,1} = \sum_{j<m_c} m_{j,m_c+1}, & \mu_{-1,2} = \sum_{j<m_c} m_{j,m_c+2}; \\
\mu_{0,0} = m_{m_c,m_c}, & \mu_{0,1} = m_{m_c,m_c+1}, & \mu_{0,2} = m_{m_c,m_c+2}; \\
\mu_{1,0} = m_{m_c+1,m_c}, & \mu_{1,1} = m_{m_c+1,m_c+1}, & \mu_{1,2} = m_{m_c+1,m_c+2}; \\
\mu_{2,0} = m_{m_c+2,m_c}, & \mu_{2,1} = m_{m_c+2,m_c+1}, & \mu_{2,2} = m_{m_c+2,m_c+2}.
\end{array}$$

(note that subscript $m_c + 1 = l - 1$ and $m_c + 2 = l$ as we have fixed $l - m_c = 2$).

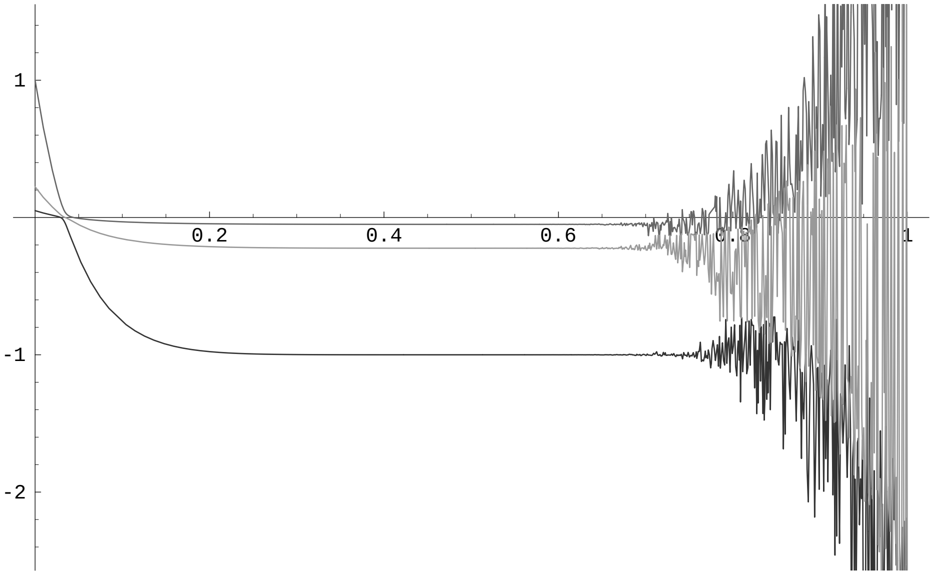


Figure 3.2 $\Re(\lambda_i)$ for $\lambda_i \neq 0$, computed for $l = 20, m_c = 18, v_c = 0.05$. In the range $(.02, .05)$ the real part of all eigenvalues pass from positive to negative values (see figure 3.3). It is also possible to identify a critical value $p_c \simeq 0.65$ correspondent to bifurcations.

3.2.1 Discussion

We have used *Mathematica*© [4] to compute the eigenvalues λ_i of the matrix Π' as functions of the parameters p_b and v_c. We may determine the qualitative behavior of the solution by looking at the sign of the real part of the eigenvalues indicated by $\Re(\lambda_i)$. Its study must not be intended in the sense to give analytical considerations about the behavior for $t \to \infty$. In fact the validity of the model is limited to the time the immune system takes to set up the immune response which, fortunately, requires a limited amount of time.

---

[4] Symbolic calculation package [128], ©Copyright Wolfram Research `www.wolfram.com`

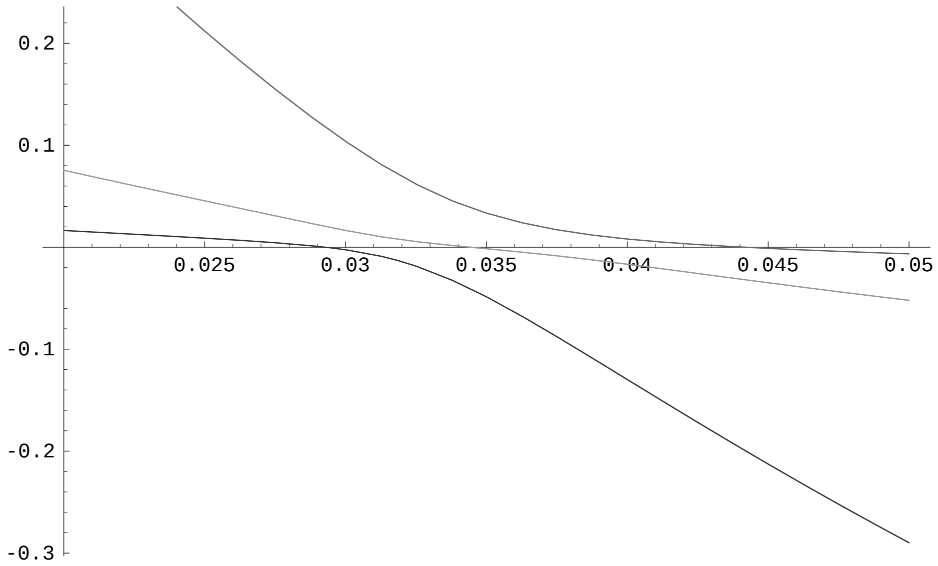


Figure 3.3 The range $\mathcal{U} \equiv (0.02, 0.05)$ of figure 3.2.

One eigenvalue is zero because of the first column of Π'. From the fact that we allow unlimited grows of the populations follows the only trivial fixed point $\mathbf{x}(0) = \mathbf{0}$, which is unstable. In any case this initial condition has no biological meaning. From figure 3.2 and 3.3, showing $\Re(\lambda_i)$ for $\lambda_i \neq 0$, it is possible to identify the interval $\mathcal{U} \equiv (u^-, u^+)$ as the range in which all the eigenvalues pass through zero and become negative (see particular in figure 3.3).

For values of $p_b < u^-$ the eigenvalues are positive and the solution will diverge in the limit $t \to \infty$. When $p_b \in \mathcal{U}$ the real part of some eigenvalues are positive and some are negative. The asymptotic behavior corresponds to an indefinite growth of *some* interacting population. Note that this argument cannot tell which population will eventually dominate. We will answer this question in the following section, showing the results obtained numerically integrating the system.

When $p_b > u^+$ all λ_i have negative real value and then the interacting populations, $x_j \geq m_c$, will feed the non-interacting, $x_j < m_c$, because of the highly disruptive mutation. This corresponds to have no immune response at all.

The range $\mathcal{U}$ is where we find the most variable solutions, ranging from the best response x_2 to the lack of response x_{-1}.

The best fitted population x_2 happens to dominate only for very small values of the mutation rate p_b. We then indicate with p_h the highest value for which the system still exhibits the domination of the fittest population x_2. This value is called the *error threshold* [9]. In the next section we will give some estimate of such value.

One may argue that $\mathcal{U} \equiv \mathcal{U}(\upsilon_c)$ indicating a dependence on the shape of the affinity. This is certainly true as we found that large values of υ_c slightly shrink the range $\mathcal{U}$ and vice versa smaller values enlarges it but in any case this variation can be reasonably ignored (on the order $< 10^{-2}$).

For large values of l ($l = 20$, approaching the real IS *expressed* specificity $10^7 \to \log_2 10^7 \simeq 24$ in the mouse), we found a critical value $p_c \simeq 0.65$ such that for $p_b > p_c$ all $\Re(\lambda_i) \neq 0$ are subjected to strong oscillations around zero (see figure 3.2). We know that, in general, parameter values for which the real part of the eigenvalues passes through zero are associated with qualitative changes in dynamics (bifurcations). The same does not apply for smaller values of the bit string length l. In fact carrying out the same calculations for $l = 3, 8$ and 12 we did not find any critical values p_c (to tell the whole story, for $l = 12$ and $p_b > 0.8$ we observed small oscillation of $\Re(\lambda_i)$ but not large enough to change the sign). Probably this unexpected behavior comes from the lack of the limiting factor K that we are going to add in the next section.

What happens if we consider $l - m_c > 2$, i.e. more interacting populations? For example if we have $l = 12$, $m_c = 9$, so to consider four interacting populations, the analogue consideration leads to $\mathcal{U} \equiv (0.045, 0.100)$ for $\upsilon_c = 0.002$ (which is the value used in [30] and will be mentioned later). This means that decreasing m_c, thus having more interacting populations $l - m_c + 1$, shows *not* to affect the value u^-, but moving $u^+ \to 1$. Instead, what is interesting is the relation between the error threshold p_h and m_c that will be investigated below.

3.3 Numerical integration

Iterating the correspondent discrete of the *complete* system 3.3 we were able to further classify the asymptotic dynamics.

We assumed non-overlapping generations in the discrete population dynamics so that the time unit is equal to the generation time, or cell duplication time, for the best fitting population (i.e. $\upsilon(l) = 1$ means that perfect match cells double every time step, match $l - 1$ double every $\upsilon^{-1}(l - 1)$ and so on).

Here we want to add the limiting factor, so instead of x_j^t we consider $x_j^t(1 - N^t/K)$, with $N^t = \sum_{j=0}^{l} x_j^t$, as the equivalent discrete and limited population variables where K is the carrying capacity.

In the following we will classify the asymptotic behavior in which only one of the four populations x_{-1}, x_0, x_1 and x_2 (reduced populations) will dominate over the others.

Asymptotic dynamics We call “attractors” the sets

$$S^j = \{\mathbf{x} : x_j > x_k, \forall k = \{-1, 0, 1, 2\}, k \neq j\}.$$

For example S^1 is the set of possible populations $\mathbf{x}$ such that $x_1 > x_k, \forall k = -1, 0, 2$. Figures 3.4 and 3.5 depict four representative trajectories leading to S^{-1}, S^0, S^1 and S^2.

The asymptotic dynamics fall in one of these, according to the mutation rate p_b and, *at a first approximation*, independently by υ_c (supposing $\upsilon_c \ll 1$ so not to be in a flat fitness landscape) and independently also from $\mathbf{x}(0)$, even though we are interested to the case in which the initial population of cells is binomially distributed as already mentioned.

We may summarize the results obtained in the following schema:

- for $p_b \in (0, p_h)$, $\mathbf{x}(t) \to S^2$, meaning the perfect maturation of the response, corresponding to the domination of the best fitting population;
- if $p_b \in \mathcal{U}$, $\mathbf{x}(t) \to S^j$, for some $j \in \{-1, 0, 1, 2\}$. This case includes all the possible responses. For $p_b \to u^+$ the match of the dominating population will decrease, until there is no response at all (i.e. x_{-1} dominates).
- For $p_b > u^+$, $\mathbf{x}(t) \to S^{-1}$, which means lack of response.

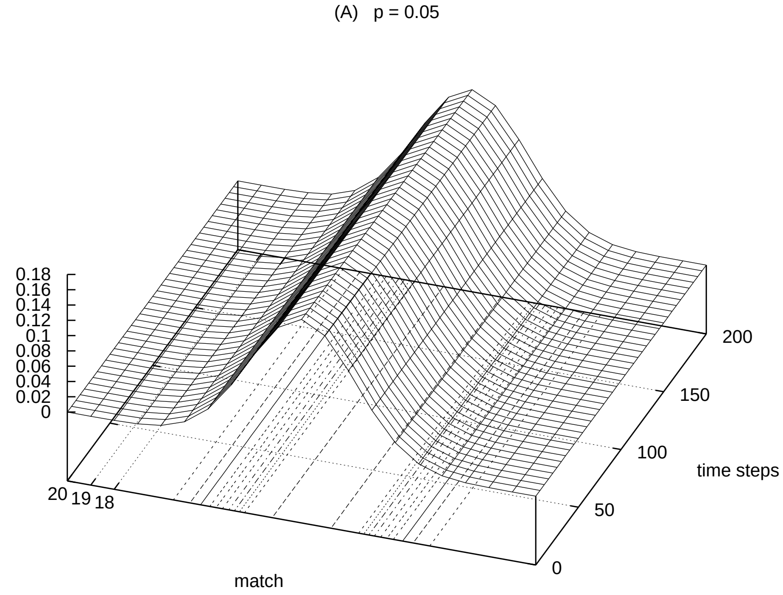
From the last point, in agreement with the analytical studies, we find u^+ to be the threshold for which the mutation is lethal, thus leading to immunodeficiency. Moreover, from figure 3.3 we find $p_h \propto \upsilon_c^{-1}$ that is a demonstration of the intuitive idea that sharp fitness favors the best adapted population.

Error threshold. In figure 3.4 we show the error threshold for different values of the interacting populations (i.e. different $l - m_c + 1$). The same values are reported in table 3.1. We see that p_h decreases exponentially with $l - m_c + 1$.

It is worth to compare these results with the ones obtained using the Celada-Seiden automaton [30]. In contrast to our schema, they used a mutation rate p_s corresponding to the probability to change *only one bit* in a string. They found the most efficient maturation occurs for $p_s = 0.2$, using $l = 12$, four interacting populations (i.e. $m_c = 9$) and $\upsilon_c = 0.002$. This means $p_s = \binom{12}{1} p_b (1 - p_b)^{11}$ giving the *per-bit* optimal mutation rate $p_b \simeq 0.021$, which is below our estimated error threshold $p_h = 0.054$ for the same parameter values, thus in line with our previsions.

3.4 Summary and conclusions

We developed a simple linear system of differential equations to model the cloning expansion and mutation in the immune response. The model mimics some of the features of the automata model of discussed in sections 2.



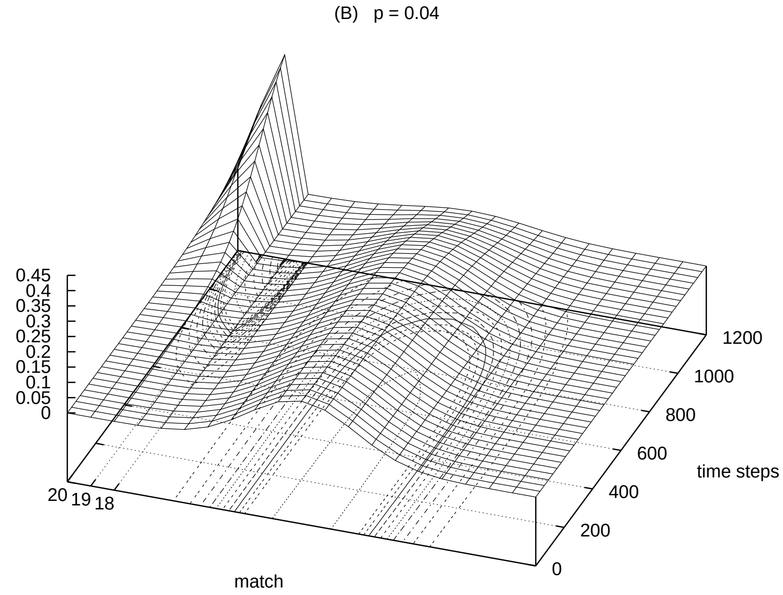


Figure 3.4 The possible asymptotic states (normalized populations for parameters $l = 20, m_c = 18, v_c = 0.05$). Typical value for $\mathbf{x}(0)$ is the binomial distribution centered around $l/2 = 10$. Picture (a) corresponds to absence of response S^{-1} for $p_b = 0.05$; (b) is the m_c-response S^0 for $p_b = 0.04$ but shown with a longer transient time (1200 iterations);

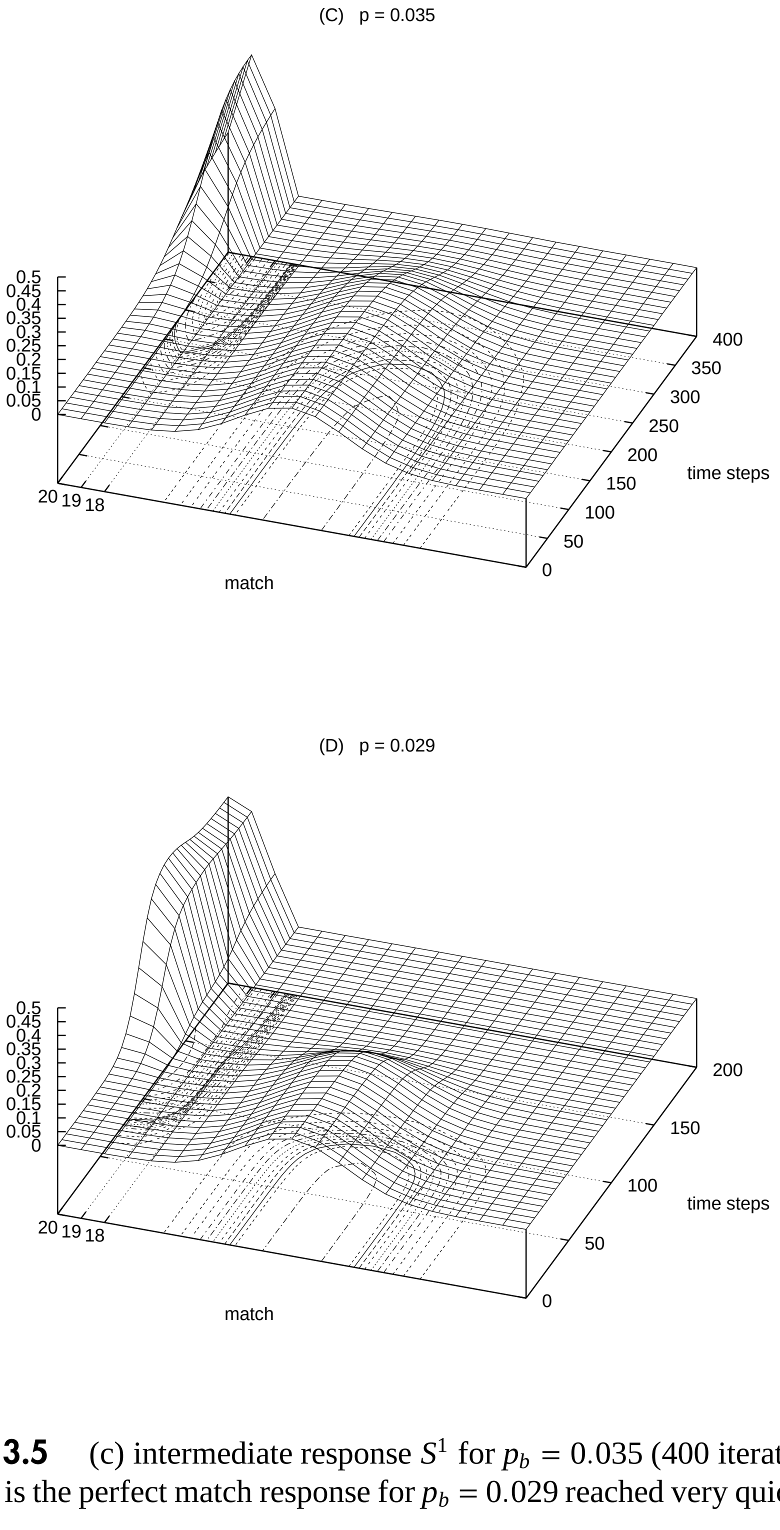


Figure 3.5 (c) intermediate response S^1 for $p_b = 0.035$ (400 iterations) and (d) is the perfect match response for $p_b = 0.029$ reached very quickly.

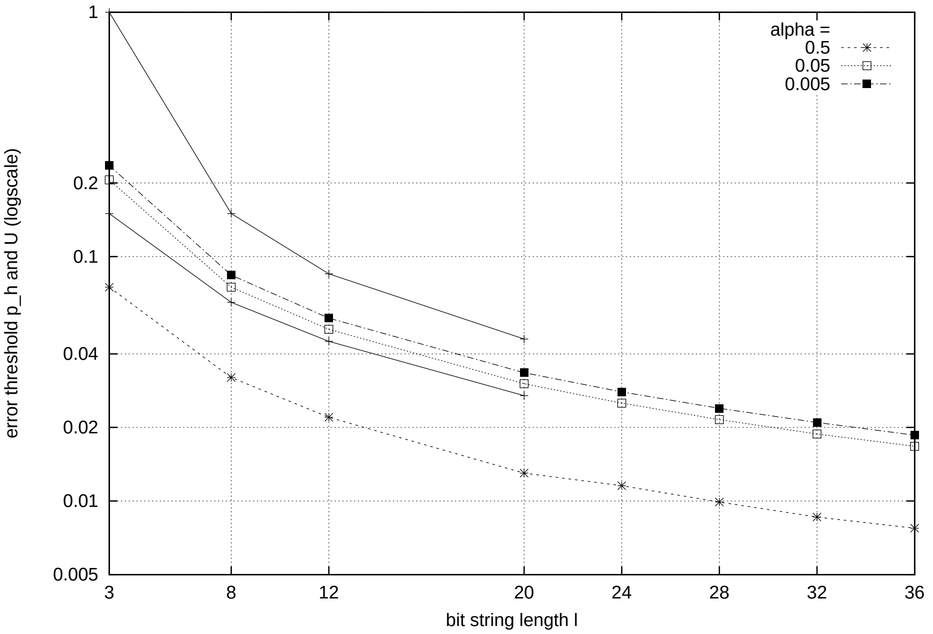


Figure 3.6 Different values for p_h computed numerically iterating over 1000 time steps the equivalent discrete of system 3.2, for $v_c = 0.005$, 0.05 and 0.5, with l = 3,8,12,20,24,28,32 and 36. The continuous lines depict the range $\mathcal{U} = (u^-, u^+)$ computed with $v_c = 0.05$. We may observe that $p_h \propto v_c^{-1}$. The carrying capacity is fixed to $K = 10^3$; its variation doesn't affect the results.

We have shown some similarity with the Eigen's model of evolution. This parallel seems particularly fruitful while studying the cloning expansion of recognizing cells apart from the complexity of the IS. In particular we were interested in questions as to what extend the fitness function of the class-specific cell populations as well as the mutation rate influence the dynamics of the response.

We have found error thresholds for the mutation rate (in agreement with [9]), both with analytical arguments and with numerical integration of the equivalent discrete system, for different values of the bit string length. They show that the error threshold scales as a decaying exponential function of the number of interacting cells populations.

It is interesting to compare these results with the well established idea that mutation is not a time-independent process, instead alternating its action between periods of free growing with periods of "hyper mutation" (the mutation rate during the hyper mutation period is most likely to be very high, i.e. $p_b > u^+$). It seems that this scheme is the optimal to achieve the maturation of the fittest population

$l - m_c$	p_h	$r_h \times 10^3$ bp mutations
1	0.019112	3.85196
2	0.018605	3.74900
3	0.017438	3.51218
4	0.016165	3.25411
6	0.013835	2.78244
8	0.011905	2.39242
10	0.010451	2.09899
12	0.009225	1.85185
16	0.007459	1.49627
18	0.006811	1.36593

Table 3.1 Error threshold for $l = 36$ e $v_c = 0.005$, $K = 10^3$ as function of the number of interacting populations $l - m_c$. In the third column the estimate of the in vivo *per base pair* error threshold $r_h = 1 - (1 - p_h)^{l/180}$.

[58]. In the light of our considerations this time-dependence could be a natural way to achieve *high mutation rates* and, at the same time, escape from an intrinsic immunodeficiency that would have been with *constant* high mutation rates.

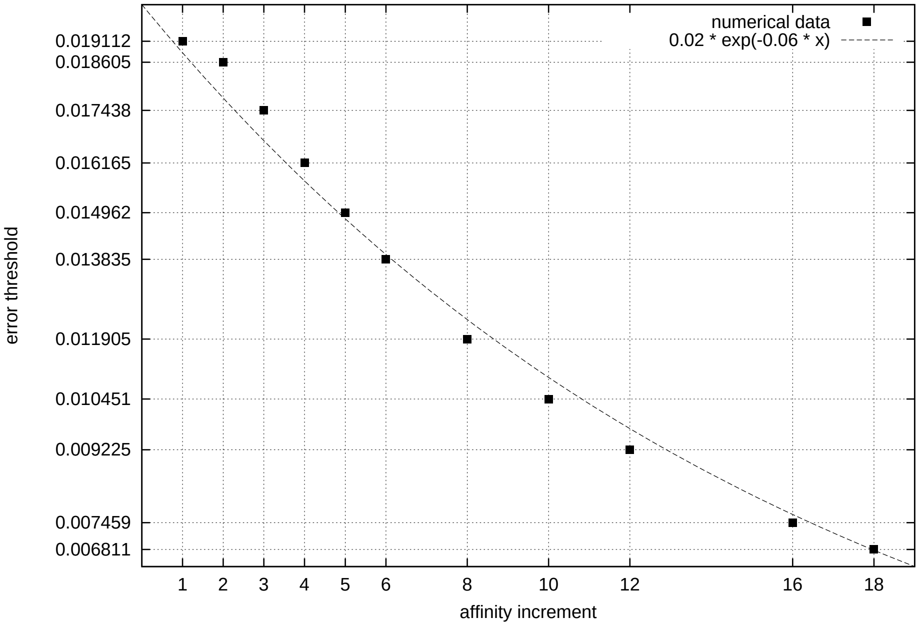


Figure 3.7 Error threshold p_h for different affinity increments $l - m_c$; $l = 36$, $\upsilon_c = 0.005$, $K = 10^3$. These values are reported in table 3.1. Exponential fit Ae^{Bx} with $A = 2 \times 10^{-2}$ and $B = -6 \times 10^{-2}$.

Part Two

Chapter 4

Econophysics

A new field of research called “Econophysics” has been populated, during the last decade, by physicists who wish to apply techniques from theoretical and statistical physics to complex dynamics such as stock prices, currency exchange rates and other more complex financial products. Apart from the interest in studying the stock markets as a genuine complex system, another reason for the increasing popularity of this field among scientists is the joint availability of powerful computers and large historical databases which store nearly every financial transaction worldwide.

Before we proceed further, it is better to define the most used and studied quantity in financial time series analysis: the price fluctuations. Some authors consider the price change $\Delta p_t = p_t - p_{t-1}$ as the most straightforward definition of fluctuation of the variable p_t, which in turn indicates the price of a certain asset at time t (e.g. seconds, minutes, days, years). Instead of considering the price change it is common practice to use one of the following definitions of *relative change* or *return* [21]:

– *simple net return* R_t, on the asset between dates $t-1$ and t, $R_t = p_t/p_{t-1} - 1$;
– *simple gross return* $1 + R_t$;
– the *continuously compounded return* or *log-return* is the logarithm of the gross return, $r_t = \log(1 + R_t) = \log p_t/p_{t-1} = \log p_t - \log p_{t-1}$.

Since Louis Bachelier’s random walk hypothesis for price change back in 1900, one of the main goals in this area was to state a correct form for the distribution of price change. In his original work, Bachelier postulated an uncorrelated random walk with independent identically Gaussian distributed increments to model the stochastic process underlying the asset price fluctuations on varying time scale.

This hypothesis has survived for a long time but it has been finally criticized and, at a certain level, contradicted. The problem with the random walk hypothesis leading to a Gaussian distribution is that it underestimates the possibility of *extreme events* such as crashes and/or bubbles. In a Gaussian, this probability is just too small. In reality this events are not so rare and actually dominate the markets. In figure 4.1, for example, it is shown both the time series of S&P500 index on a daily time scale (from February 1992 to January 2000) and a random walk with Gaussian independent increments (generated by $x_{t+1} = x_t + \varepsilon_t$ with $\varepsilon_t \propto \mathcal{N}(\mu, \sigma^2)$, Gaussian with mean $\mu = 0$ and standard deviation $\sigma = 2$). The arrow in figure points to a "rare event", something much unlikely to be produced by a random walk. A well known "rare event" is the market crash reported in 1987. Figure 4.2

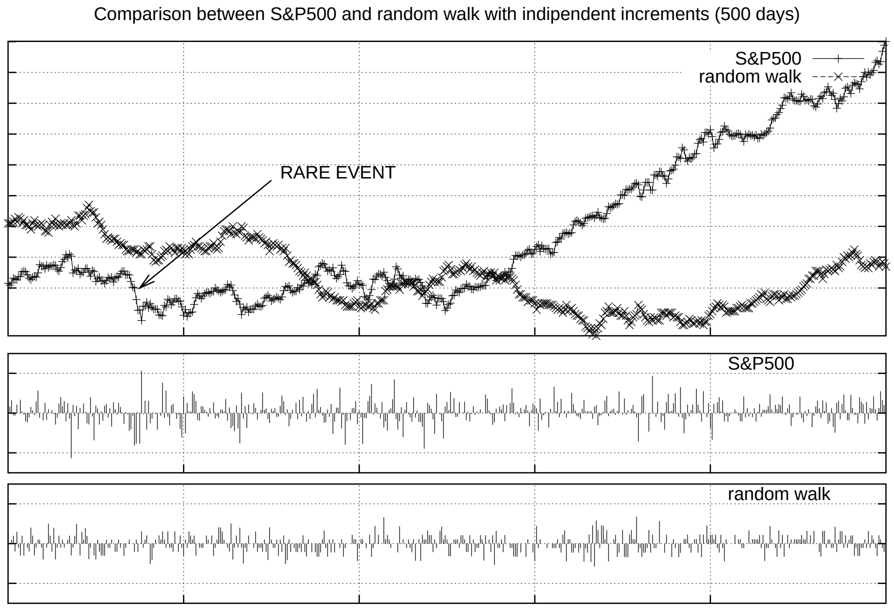


Figure 4.1 Comparison between a random walk with *i.i.d.* increments iterated for 500 time steps and 500 daily close values of the index S&P500 during the periods February 1992 - January 2000. In the small plots down in the figure it is shown the correspondent net return R_t.

shows two major indexes of the New York Stock Exchange (NYSE), the Standard and Poor 500 (S&P500) and the Dow Jones Industrial Average (DJI) taken over a period of fifty years on a daily time scale. The market crash of 1987 is visible. Strong fluctuations are visible in the lower plot which reports the log-return computed for the DJI.

The random walk hypothesis with *independent identically distributed* (*i.i.d.*) increments is the basis of the *Efficient Market Hypothesis* (EMH). It states, in few and simple words, that price variation is random as a results of the activity of traders who attempt to make profit (arbitrage opportunities); the application of

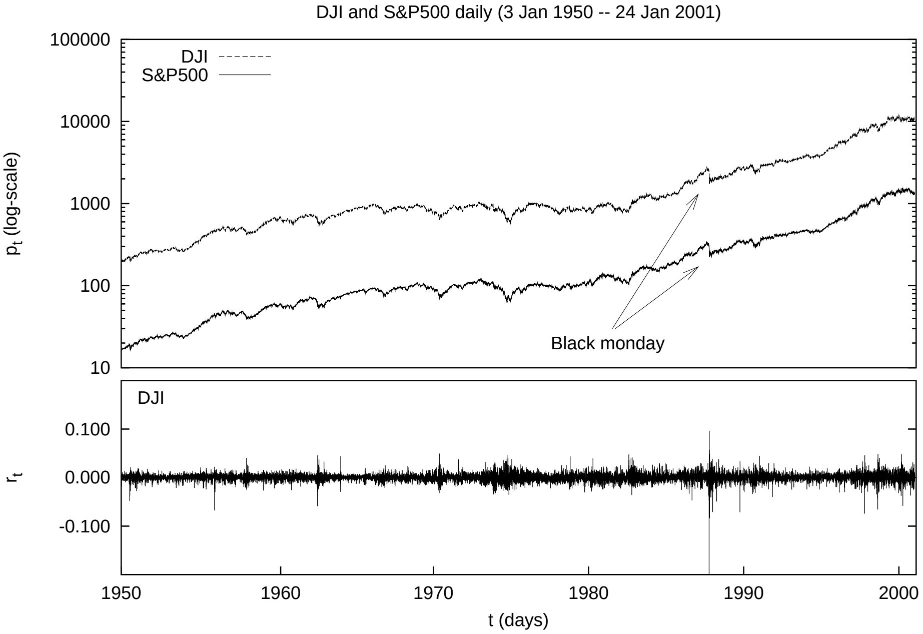


Figure 4.2 DJI and S&P500 indexes are shown together with the log-return computed for the DJI series. The market crash (black Monday) of October 19, 1987 is shown.

their strategies induce a dynamical feedback on the market influencing the stock price that become random as a consequence. Thus, under the strict EMH, methodic strategies producing wealth are impossible. On the other hand, we know that some people are actually able to take advantage of deviations from this law. To overcome this contradiction various variants of the EMH have been suggested (strong, semi-strong and weak) [21, 129]. In particular it seems that in a weak or semi-strong efficient market hypothesis there is place for smart speculators; a probabilistic edge to be exploited in real markets [129, 83].

4.1 Lévy stable distribution and fat tails

The only partial agreement of the EMH (i.e. the random walk with independent increments) with the recently discovered empirical evidences, suggested the needs for more accurate, though more complex, probability distribution functions (PDF) than the Gaussian as models for price fluctuations or return.

In 1963, Benoit Mandelbrot proposed the Lévy stable distribution as model for the distribution of returns [78]. He proposed a power law form $P(x) = |x|^{-(1+\alpha)}$ with

$0 < \alpha \leq 2$ for intermediate and large $|x|$, but rounded peak at $|x| \to 0$. In statistics, the Lévy distribution arises from the generalization of the central limit theorem to a wider class of distributions which do not have a finite second moment [70, 49, 18]. In particular it is the asymptotic distribution of a random variable $X_n = \sum_{i=1}^{n} x_i$ for $n \to \infty$ where the variables x_i are distributed as $P(x) \propto |x|^{-(1+\alpha)}$.

In general, the *central limit theorem* of statistics [43] assures that under certain conditions of the distribution of the variables x_i, the sum X_n converges to a Gaussian or to a Lévy distribution. The existence of the variance of the variables x_i plays a role. In particular, if the variance is infinite then only if x_i is distributed as a Lévy one can have convergence. In fact, for generic distributions of x_i, if the variance is finite then the sum converges to a Gaussian (at least in the central part) otherwise it does not converge. Instead, if x_i are distributed as a Lévy (or power) law then ii) for infinite variance ($\alpha < 2$) X_n converges to a Lévy stable distribution; iii) $\alpha = 2$ converges to a Gaussian; iv) for finite variance ($\alpha > 2$) X_n converges to a Gaussian in the central part but with non-Gaussian tails.

The Lévy stable probability density does not have an analytic closed form but can be expressed by its characteristic function [76] or in term of its Fourier transforms [50]. This distribution function is *leptokurtic* (positive excess kurtosis [1]) that is, it has more probability mass in its tails and center than a Gaussian. The advantage over the Gaussian is that the Lévy PDF with its fat tails accounts for *a higher probability of extreme events.*

Recent empirical studies conducted on asset prices of different markets have shown *partial* agreement with the Lévy distribution [81, 50]. In fact the excess kurtosis of daily returns ranges between 2 and 50, and is even higher for intra-day data.

Although Lévy distribution are stable under convolution, that is, the sum of two independent random variables characterized by a Lévy distribution is itself characterized by the same Lévy distribution, the problem is that the resulting limit distribution have infinite second and higher moments. This is in contrast with empirical data which show finite second moment in the distribution of return. Moreover, the central part appear to be well fit by a Lévy distribution but the asymptotic behaviour shows faster decay then predicted by a Lévy distribution. For this purpose a *truncated Lévy* distribution (TL) has been proposed [77, 82, 81]. Recently the *scale-invariant truncated Lévy* distribution (STL) has been proposed to model the *scale invariance* observed in empirical data [78, 81]. The STL distribution is defined as

$$P(z) = Ae^{-\lambda|z|^{\beta}}|z|^{-(1+\alpha)} \qquad 0 < \alpha < 2 \quad \beta > 0.$$

---

[1] The kurtosis κ of a distribution is defined as $\kappa = \mu_4/\sigma^4$ where μ_4 is the fourth central moment and σ is the standard deviation. One often refers to *excess kurtosis* as $\kappa - 3$, relative to the kurtosis of the Gaussian $\mathcal{N}(0, 1)$.

λ^{-1} is related to the size of the truncation of the Lévy distribution and the exponential pre-factor $e^{-\lambda|z|^{\beta}}$ ensures a smooth truncation. The parameter A determines the "spread" in the central region [82, 97].

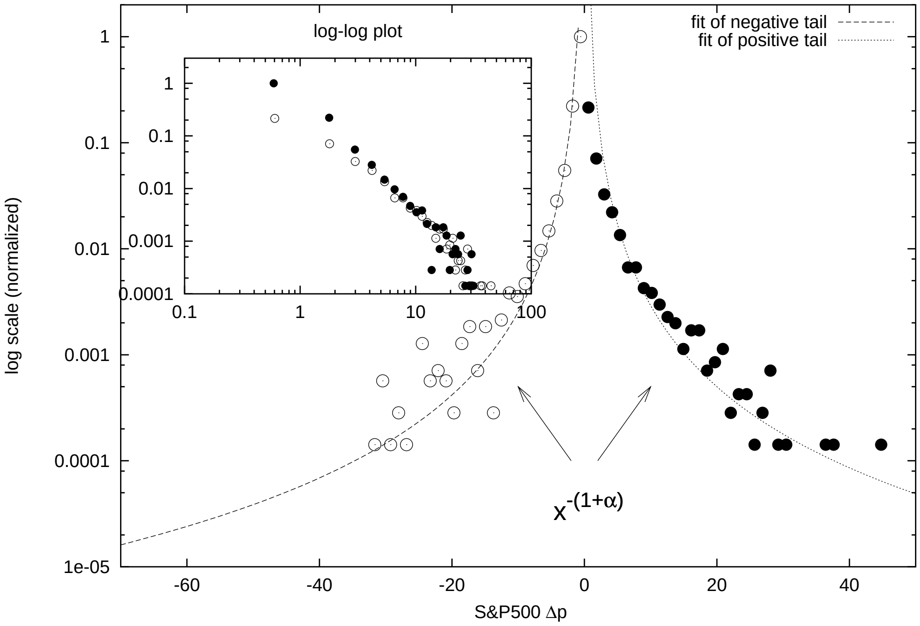


Figure 4.3 Histogram of Δp for the S&P500 index computed over 12446 daily close values and fit with a power law $x^{-(1+\alpha)}$ with $\alpha = 1.598$ for the negative tail and $\alpha = 1.538$ for the positive tail. In the inset panel it is shown the tails of the distribution in log-log scale. The excess kurtosis of the equivalent unnormalized histogram is $\simeq 56$.

4.2 Stylized facts of financial time series

Empirical studies on volatile markets have discovered some universal characteristics of price fluctuation. Some of them are well accepted while others are still under debate.

Fat tails The already discussed leptokurtosis of returns on small time scale (see figure 4.3). In empirical studies [50] a universal power law with exponents between 3 and 5 has been found.

Multi-scaling The concept of scaling in physics has been applied to the distribution of return (see section 5.1.1). It is still quite controversial whether the distribution of return is multi-scaling or not. Some studies reported that the distribution of return possesses the property of *scale invariance* with respect to the time scale (i.e. mono-scaling). In other words, rescaling the histograms of returns computed on different time scales (1 min, 1 day, etc) this will overlap [78, 81]. This item will be addressed in the following chapter.

No correlation of price return The absence of correlation of price returns. This means roughly that the price time series looks superficially as a random walk with independent increments. To measure correlations in a numerical series x_t one often uses the *autocorrelation function* defined as

$$c(\tau) = \frac{\sum_{t=1}^{N-\tau}(x_{t+\tau} - < x >)(x_t - < x >)}{\sum_{t=1}^{N-\tau}(x_t - < x >)^2} \tag{4.1}$$

For time scales beyond 20 minutes the autocorrelation function of r_t is found at level of noise [50] in agreement with the EMH. Thus, for time scales beyond hours or days no correlation is observed (see inset panel of plot in figure 4.4).

Clustering of volatility The amplitude of increments usually called *volatility* can be defined in different ways. We sometimes adopt the following definition of volatility, $v_t = \sqrt{r_t^2} = |r_t|$. In real data the volatility clusters in time, meaning that its autocorrelation function is positive and decays slowly to zero. Empirical studies have reported autocorrelation function with universal power law decay with exponent between 0.1 and 0.4. In figure 4.4 we plotted the autocorrelation function $c(\tau)$ for the net return R_t of the Dow Jones Industrials index from more than 12398 daily close values. We also plotted the autocorrelation function of the absolute net return $|R_t|$, i.e. the volatility. The volatility possesses strong correlation slowly decaying to zero. The solid line in plot 4.4 corresponds to a fit with $0.03 \cdot \log(100/\tau) + 0.07$ resembling the curve for volatility defined in [108] and related papers.

Log-normal distribution of volatility This empirical fact has been recently observed in a large database of financial index of the NYSE [73]. Later studies have reported similar results on various currencies exchange rates. The form of the right tail is controversial instead. Some studies [73] reported a power law behaviour with exponent $\simeq 3$ for the S&P500 index while other a common decay in the right tail as in a log-normal distribution for the NYSE index [92].

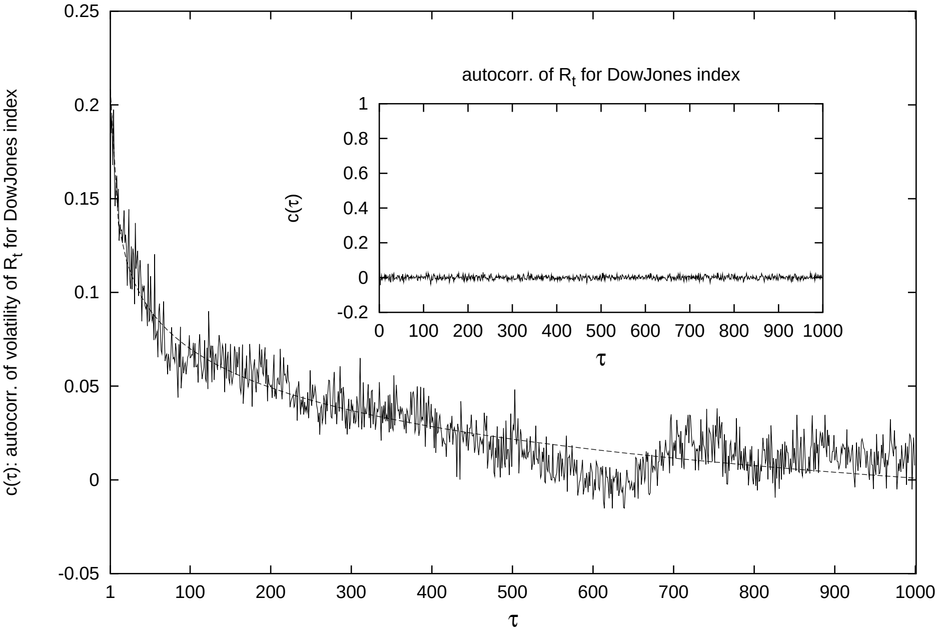


Figure 4.4 In the inset the autocorrelation of net return R_t computed over 12398 daily close for DJI. It shows absence of two point correlations. Instead, the autocorrelation of absolute net return $|R_t|$ shown in the large plot indicates strong correlation slowly declining to zero.

Cross-Correlation between absolute return and trading volume There seems to be a positive correlation between the absolute value of the return and the trading volume (IBM return-volume cross-correlation [63]).

In the following chapter we will discuss some MS models of stock markets that try to reproduce and explain the origin of these stylized facts. In particular the Cont-Bouchaud model seems to be one of the first who proposed the *herding behaviour* as the decisive factor of the fat tailed distribution of return. We will show that this model is able to produce price time series that are not scale invariant, i.e. they are multi-affine. In the successive chapter we will then introduce another microsimulation model able to account for some other stylized facts of financial markets. It is much more sophisticated than the Count-Bouchaud and resemble in many aspects features of other MS models reviewed at the beginning of chapter 5.

Chapter

5

Microsimulation of stock markets

The basic idea is to develop simple microscopic models dealing with the interaction between traders. These models aim to explain the underlined mechanisms that determine the complexity of price fluctuations. All above is summarized in the words of the economist Stigler in 1964 who was the first to perform a stock market simulation [117]:

"The goal is to have, on the one hand, the simplest and most parsimonious description of the market and, on the other hand, the most faithful representation of the observed market characteristics."

Many microsimulation systems have been described in literature (see [66], [76] or [62] for a review). In these models, the traders (single or group of investor) acts buying or selling commodities, e.g. stocks, gold, foreign currencies. The price is then determined, as usual, proportionally to the difference or in more complicated ways to balance supply and demand. The traders assume different strategies according to their stylized behaviour: *chartists* (who follows trends and are subject to some sort of herd behaviour) versus *fundamentalists* (who buy/sell when the price is believed to be below/above the fundamental value) or *noisy* (small investors) versus *capitalists* (people or group of investors who move large quantity of money).

One example is given by Lux and Marchesi which distinguish between *optimistic* and *pessimistic* according to the attitude to follow the fundamental price [77]. In their model traders are not bound to remain within the same group, that is, they may move to a different group if the corresponding strategy gives an advantage.

LeBaron *et al.* [63] or Chen *et al.* [31] try to reproduce the information processing (i.e. the learning process) of individual traders.

Cont *et al.* [33] represent agents as vertices of a random graph to model the *herd behavior* of traders.

Another well known model is the one of Levy, Levy and Solomon (LLS-model) [67, 68, 69]. In that model agents are sophisticated entities. They have the choice

between a risk-less asset (like a *bond*) and a risky asset with stochastic returns. Moreover agents may reinvest the capital or accumulate it. The authors start with a set of rational, informed and identical traders and then, one by one, they add elements of heterogeneity and deviations from rationality to study their effects on the market dynamics. Moreover, traders base their decision on past returns over a certain time horizon.
Kim and Markowitz distinguish between liquidity and stocks at the current nominal price [59].
As a further step towards more realistic simulations, Farmer [39] considers different strategies for *value traders* instead of the simple random buy/sell/inactive choice of [33].

Each of these works allows to achieve a better understanding of the components that may be included in a model of financial markets and how their interaction influences the overall dynamics. Following this line one may think to model not only different characters but also the way they interact (see also the evolutionary agent based models of [41, 40]) to understand the possible relations between traders behaviour and price fluctuations and in general, the statistical characteristics of the market.

5.1 The Cont-Bouchaud herding model

One of the simplest models able to show fat tails in the histogram of returns is the Cont-Bouchaud herding model (CB-model) [33]. In their work the authors showed the relation between the excess kurtosis observed in the distribution of price change and the tendency of market participants to imitate each other (what is called *herd behaviour*). The model consists of a random graph in which the nodes are occupied by agents. The connection between agents have the meaning of forming a coalition of traders influencing each other so to choose the same strategy to buy, sell or not to trade. At a given time step a certain number of coalitions, formed by the random matching, decide what to do. In particular each *cluster* buys with probability a, sells with probability a or stays inactive with probability $1-2a$. The demand of a certain group of traders is proportional to its size. The parameter a determines the “frequency” by which the traders buy or sell: small $a \ll \frac{1}{2}$ means short time intervals with few trades; large $a \simeq \frac{1}{2}$ means long time intervals where a large fraction of all investors participate. Iterating this process the price is determined for each time step and the histogram of returns is computed. The result (also shown analytically in [33]) is the relation between a parameter controlling the connectivity of the graph (i.e. the tendency of the agents to group together) and the kurtosis of the out coming histogram. As expected, high connectivity induces fat tails (or excess kurtosis) in the histogram of returns.

The same model can be seen considering *percolating* clusters instead of random graph [109, 115]. In this view the Cont-Bouchaud random graph corresponds to an infinite-range bond percolation lattice. Same results are found if we limit ourselves to lattices with nearest neighbors and to site percolation instead of bond percolation [110].

In *Percolation Theory* every site of a d-dimensional lattice is occupied with probability p and left empty with probability $1-p$ (the same as the links between two agents in Cont-Bouchaud's random graph). A cluster is then defined as a group of neighboring occupied sites. According to the exact definition of neighborhood we have different cluster formations. In the following we will refer to square lattices ($d = 2$) so that the 4-neighborhood corresponds to up-down-left-right [1]. Each occupied site of the lattice identifies one investor. Clusters are group of investors acting together as in the CB-model.

For p above some critical value p_c an infinite cluster appears spanning the lattice from one side to the opposite side. At the percolation threshold p_c the average number $n_s(p)$ of cluster containing s sites each, varies as a power law $n_s \propto s^{-\tau}$ with a certain exponent τ.

For market applications, each site of the lattice represents one single investor while the percolation clusters are group of investors acting together. As in the CB-model, at every time step of the simulation, each cluster buys with probability a, sells with probability a or stays inactive with probability $1-2a$. The demand of a certain group of traders is proportional to its size. Denoting n_s^+ the number of buying clusters and n_s^- that of selling clusters, the relative price change is proportional to the difference between supply and demand

$$\Delta p \propto \sum_s s \times n_s^+ - \sum_s s \times n_s^- . \tag{5.1}$$

Non-trading clusters have no influence. In this model all investors have infinite supply of credits to buy stocks and infinite number of stock to sell. Moreover, the availability of stocks to buy is always assured.

The power law of the distribution of returns comes from the cluster size distribution of percolation theory: i) $p < p_c$ gives less volatile behaviour, i.e. unrealistic Gaussian also for small a; (ii) $p > p_c$ gives unrealistic oscillations (crashes and bubbles) because the market is dominated by the "infinite" cluster, whatever a; (iii) at $p = p_c$ the fraction p of lattice sites occupied by the investor in a d-dimensional lattice of linear extent L barely suffices to form an "infinite" cluster stretching from top to bottom. In this case we observe power laws: $n_s \propto s^{-\tau}$, $P(r) \propto r^{-\tau}$ for $1 \ll s \ll L^D$ where n_s is the number of clusters of size s, r is the return and $D = d/(\tau - 1)$ is the fractal dimension of the percolating cluster [115]. This last case

---

[1] Generalization to higher dimension can be found in [114, 111, 115, 112].

is much more realistic as we observe power law consistent with the Lévy regime for small activity a and crossover to Gaussian for increasing $a \to \frac{1}{2}$.

This model reproduces some stylized facts of real markets: i) the average return $r_t = \log p_{t+1} - \log p_t$ is zero, ii) there is no correlation between two successive returns simply because there is no memory in the decision of the investors, iii) a power law for the tails of the histogram of return holds for small values of the activity parameter a while for larger $a \to \frac{1}{2}$ (longer time) a progressive crossover to a Gaussian is observed.

No other stylized facts are fulfilled by this simple model if no complications are added. For example in [111] traders are allowed to diffuse on the lattice to induce realistic correlations of volatility.

We have implemented the CB-model in C language with a naive parallelization using the PVM message passing library. The core of the algorithm is a *clustering labeling algorithm* of percolation theory. For this purpose we tried two different algorithms:

1. a recursive algorithm very intuitive and easy to implement;
2. a performant algorithm of *Hoshen and Kopelman* (1976) described and reported in Fortran language in [110].

After having produced a percolation cluster the rest of the Cont-Bouchaud algorithm consists of iterating the traders activity for all the clusters and determine the price. Figure 5.1 shows the histogram of price changes $\Delta p_t = p_t - p_{t-1}$ obtained for different activity a of traders in a two dimensional lattice ($L = 100$) at the critical value $p_c = 0.592746$ over a time period of time steps. The interesting behaviour of the histogram of Δp depending on the parameter is that for large value of activity a there is a crossover to a Gaussian exactly as observed empirically in real markets price ([81] and further papers). In real prices the crossover to a Gaussian is found on time scales from four days up to one month.

5.1.1 Multi-affinity in financial price series

Empirical evidence of non-unique scaling exponent in time series of stock prices, currency exchange rates or market indexes can be found in many recent papers. For instance, Baviera *et al.* analyse the German DM/US$ dollar exchange rates [11], Ghashghaie *et al.* the high frequency bid-ask quotes for the US$/DM exchange rates [48], Rotyis *et al.* analysed a stock index of the Budapest Stock Exchange [102], Lux worked on the German DAX, the NYSE Composite Index, the DM/US$ exchange rate as well as the gold price from the London Precious Metal Exchange [75], Vandewalle *et al.* analyse the US$/DM and JPY/US$ exchange rates [120], Ivanova *et al.* look at Gold price, Dow Jones Industrial Average and

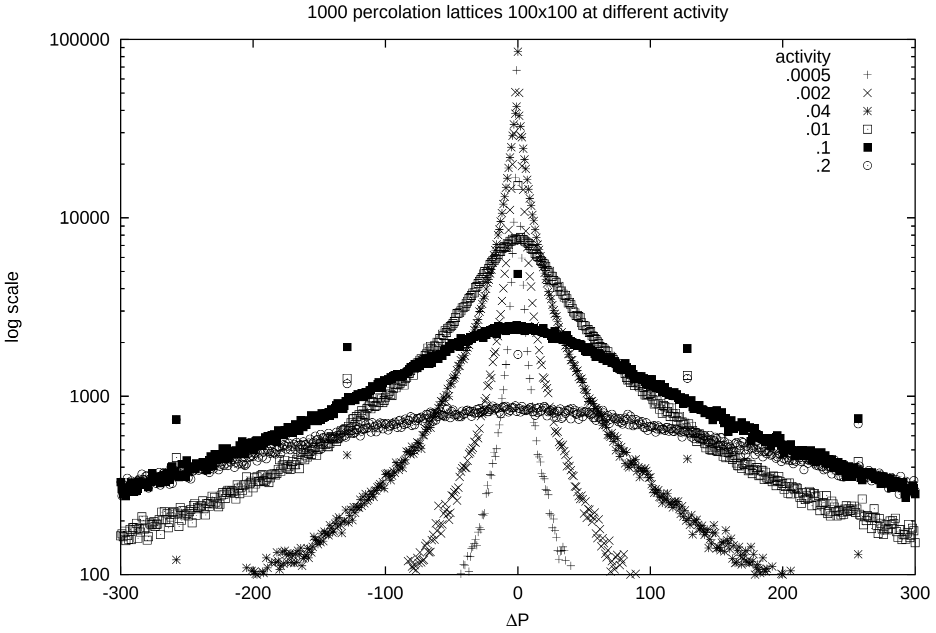


Figure 5.1 Histogram of Δp_t determined with the Cont-Bouchaud herding model. Thousand critical square lattices 100×100 iterated for 1000 time steps. For $a \to \frac{1}{2}$ a progressive convergence to a Gaussian process is observed as in real data.

BGL/US$ exchange rate [55]. They all showed that scaling of return exists but *not with a unique* exponent.

To detect multi-scaling we employ a method that is considered standard in literature of turbulence theory [45] and used in some of the already mentioned works. Define the *structure function*

$$F_q(\tau) \equiv < |r_t^{(\tau)}|^q > \tag{5.2}$$

where $< \cdot >$ is the time average and $r_t^{(\tau)}$ is the time-lagged return defined as

$$r_t^{(\tau)} \equiv \sum_{i=t+1}^{t+\tau} r_i = \log \frac{p_{t+\tau}}{p_t}. \tag{5.3}$$

In the following analysis we will use the log-return $r_t^{(1)} = \log p_{t+1} - \log p_t$ as definition of return while Rotyis *et al.* in [102] and Iori in [54] take the simple price difference $r_t^{(1)} = p_{t+1} - p_t$ thus $r_t^{(\tau)} = p_{t+\tau} - p_t$ instead.

The structure function $F_q(\tau)$ is assumed to scale as a power law i.e. $F_q(\tau) \propto \tau^{\zeta_q}$, where ζ_q is the *scaling exponent*. If $\zeta_q = hq$ for some h, then the process is self-affine (sometimes called uni-fractal). $h < \frac{1}{2}$ is the Lévy-stable case while the Gaussian behaviour corresponds to $h = \frac{1}{2}$. On the contrary, if the scaling exponent is not linear (ζ_q is a convex function of q [43]), the process is called *multi-affine* or multi-fractal. The larger is the difference of ζ_q from the linear behaviour in q, the wilder are the fluctuations and the correlation of the absolute return [53].

Given $F_q(\tau)$, computed as in eq(5.2) for each q, the value of ζ_q has been estimated by standard linear regression over the set of points $\{(\tau, F_q(\tau))\}$ for $\tau = 2^0, \ldots, 2^{16}$. Moreover $F_q(\tau)$ is computed for different values of the exponent $q = \frac{k}{2}$ with $k = 1, 2, \ldots, 20$.

It is well known that, in contrast to the early prediction of Brownian motion given by Louis Bachelier one century ago, real price series have different behaviour [81]. Brownian motions, besides a Gaussian distribution of return, lead to a uniform exponent $\zeta_q = q/2$. In contrast, as already mentioned, financial time series show a scaling exponent very different from $q/2$. To show this we made a quick check on the Dow Jones Industrial Average Index (DJI daily closing) over seven decades, from 1-Oct-1928 to 22-Aug-2000, for a total of 19103 values [2]. At the same time we calculated the scaling exponent of a time series generated by a multiplicative Gaussian random walk $\log p_{t+1} - \log p_t = \varepsilon_t$ with ε_t drawn from a Gaussian $\mathcal{N}(0, \sigma^2)$ with $\sigma = 10^{-2}$, over five million time steps.

Contrary to the scaling exponent of the random walk which we found to have a slope very close to $\frac{1}{2}$, the scaling exponent of the DJI shows in agreement with the literature cited above a clear deviation from $q/2$ (see fig. 5.2). In particular two regions are visible; roughly $q < 3$ with a slope $\simeq \frac{1}{2}$ and $q > 3$ with slope $\simeq 0.12$.

We then generated time series with the CB model and computed the scaling exponent for different values of the activity a and system size L at the critical percolation threshold $p_c = 0.592746$ in two dimensions. The same departure from a Gaussian behaviour, but also from a simple scaling behaviour (straight line), is found for some value of the activity a. Figure 5.3 shows the scaling exponent ζ_q computed as the average of fifty independent runs (different random seeds, i.e. different lattice cluster formations) for each of four values of the activity ($a = 10^{-4}, 10^{-3}, 10^{-2}$ and 10^{-1}) and four values of lattice dimension ($L = 1001, 701, 401$ and 101). A total of 800 simulations composed by one million time steps each took a total running time of about a thousand hours on three fast workstations.

Plot (a) of figure 5.3 refers to $a = 10^{-4}$. Two different regimes are present, one for small q ($= \frac{1}{2}$ and 1) and another for $q \geq \frac{3}{2}$. No significant dependence from the

---

[2] Note that this analysis is partial when compared to those in the referenced bibliography because of the limited amount of points at our disposal. Indeed, using only 19103 values, the linear fit of the structure function was performed over 10 values of τ only instead of 16, i.e. τ varied from 2^1 to 2^{10} instead of 2^{16}.

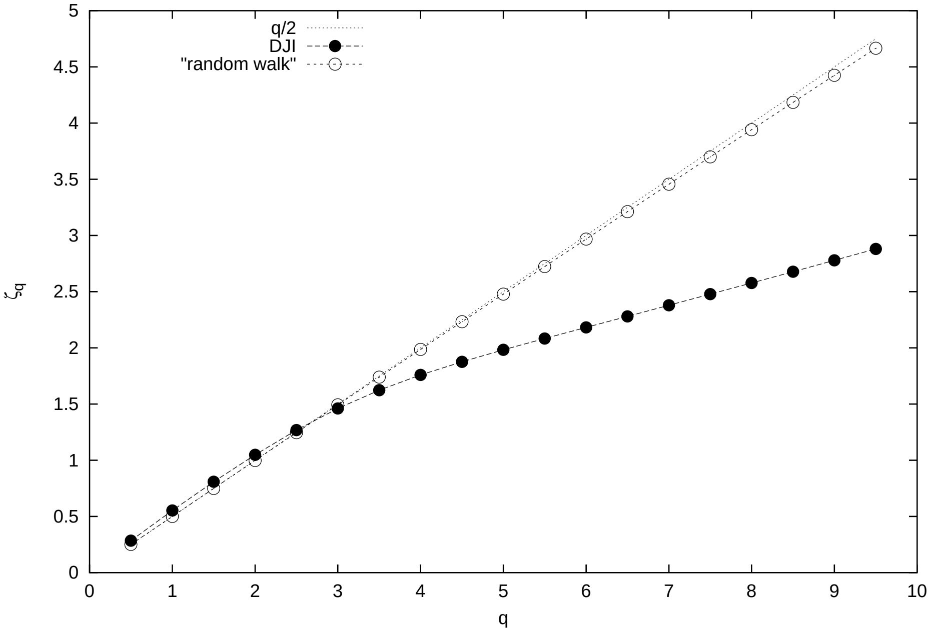


Figure 5.2 Compare the scaling exponent of a multiplicative Gaussian random walk with that of the DJI daily closing values.

lattice dimension L is found (see also figure 5.4). Similar multi-scaling behaviour is found for $a = 10^{-3}$ (panel (b)).

Increasing the activity the model goes to self-affine behaviour as seen in panels (c) and (d) in figure 5.3 corresponding respectively to $a = 10^{-2}$ and $a = 10^{-1}$. The scaling exponent becomes a straight line with slope always below $\frac{1}{2}$ indicating a Lévy regime. The slope of the scaling exponent depends strongly on the lattice dimension L. Indeed figure 5.3 panel (d) which corresponds to $a = 10^{-1}$ shows the slope of the scaling exponent to be inversely proportional with the lattice size L. This finite size effect is better shown in figure 5.4 where we show the scaling exponent relative to $L = 1001$ which is the most representative of the asymptotic behaviour of the percolation of critical lattices. In this plot a crossover to a self-affine behaviour is visible for a going from 10^{-4} to 10^{-1}.

From percolation theory we know that the cluster size distribution is a power law with a certain exponent. Moreover, if the activity a is very small, at any time step only one cluster is active, thus according to eq(5.1), the price change or return is distributed as a power law (this is trivially true in particular when $a < 1/L^2$). Such dynamics seems equivalent to a Lévy walk which we know to be a self-affine process. Thus we need not to look for multi-scaling when $a < 1/L^2$.

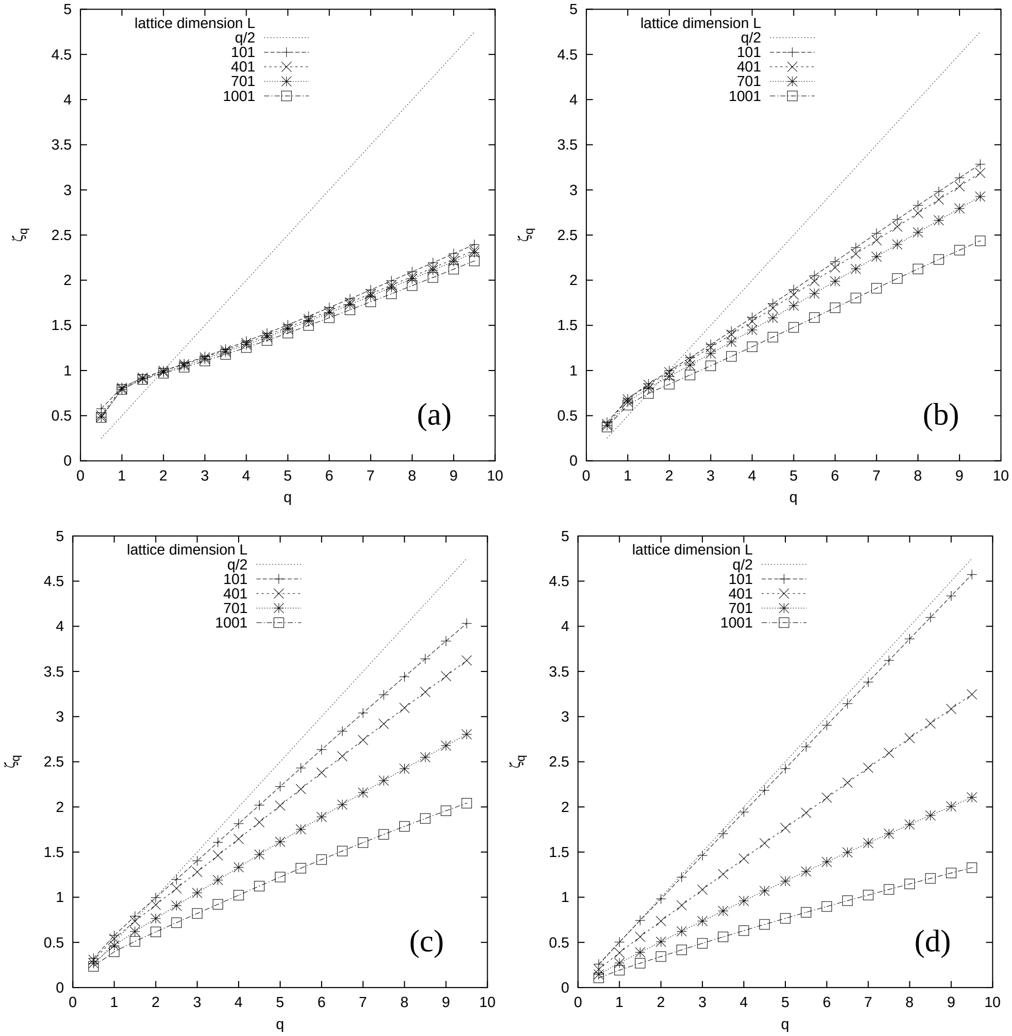


Figure 5.3 Panel (a) corresponds to $a = 10^{-4}$, panel (b) to $a = 10^{-3}$, (c) to $a = 10^{-2}$ and (d) to $a = 10^{-1}$. q on the x-axis, the scaling exponent ζ_q on the y-axis. For each a and L, the scaling exponent ζ_q has been computed. This has been repeated for fifty independent runs (different random seeds, thus different cluster configurations). The scaling exponent here shown is the result of averaging for the single ζ_q.

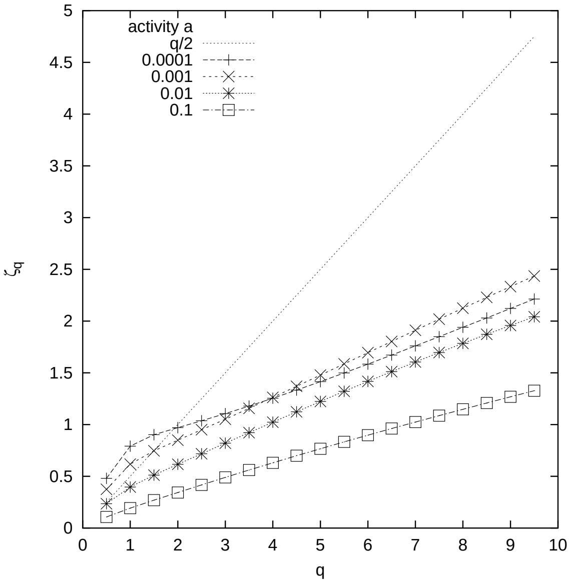


Figure 5.4 Scaling exponent for $L = 1001$, different activity.

Instead, when $1/L^2 \ll a \ll \frac{1}{2}$ the process seems to be multi-affine. This is non-trivial and, more important, is well in line with the empirical findings relative to the real financial series. However, the value of q for which the scaling exponent change its slope ($q \simeq 3$ in figure 5.2 for the DJI) is quite different for the synthetic series generated with the CB-model ($q \simeq 1$ in figure 5.2) and independent from the two parameters investigated in this work, namely the activity a and the system dimension L.

In summary it seems the CB-model shows interesting behaviour over a certain window of the parameter a giving power-law histogram and multi-scaling as real financial time series. Simulations suggest this window to be independent of the system size L.

Chapter 6

An agent – based model of stock market fluctuations

In this chapter we describe a new model to simulate the dynamic interactions between market price and the decisions of different kind of traders. They possess spatial mobility and group together to form coalitions. Each coalition follows a strategy chosen from a proportional voting "dominated" by a leader's decision. The interplay of the different kind of agents gives rise to complex price dynamics that is consistent with the main stylized facts of financial time series.

The present model incorporates many features of other known models and is meant to be the first step toward the construction of an *agent based* model that uses more realistic markets rules, strategies, and information structures. The main goal is to give an easy way to implement different key issues in modeling the stock market, to understand the relevance and the mutual influence of certain factors that other models have treated separately, and to investigate the necessary and sufficient conditions determining the factors which actually drive the empirical observed facts in real markets.

The modeling and simulation of real systems consisting of agents that cooperate with each other has emerged as an important field of research. They are regarded as a consistent paradigm enabling an important step forward in empirical sciences, technology and theory [5]. To model the dynamics of a complex system composed of interacting entities with their internal complex structure and dynamics has many appealing points: self organization strategies and decentralized control, emergent behaviour, autonomous behaviour, cooperative capacity and aggregation, and spatial mobility. The purpose of the model hereafter described is to provide a relatively simple description of the price formation in a stock market. The agents paradigm is most suited to accomplish this task.

Recalling the introduction, the computational model described herein is formally equivalent to an *unbounded lattice gas*. The technical implementation of this computational model called **AMSE** (A Model of Stock Exchange) is very similar to the coding of **CImmSim** already described in chapter 2 and also in appendix B. In appendix C some details about the parallel implementation of **AMSE** are given.

This chapter is organized as follows: in section 6.1 we present the model, in section 6.2 we discuss its dynamics presenting the results of some simulations. In section 6.3 we use a well known statistics to determine long-range correlations in the time series of the absolute return generated by **AMSE**. Finally, in section 6.4 we point out some further developments.

6.1 Model description

In the first version of the model [24] there were two kinds of agents trading for a single asset (stock): *fundamentalists* and *noisy* [77, 116]. The former consider a reference (or "fundamental") value to determine the "right" price of an asset. The latter represent most of the "small" traders which do not follow any reference value and do not look at charts. Their behaviour is mostly random.

Following [77] the fundamental value f_t is modeled as a simple multiplicative random process $\log(f_{t+1}) - \log(f_t) = \varepsilon_t$ where relative changes are drawn from a Gaussian $\varepsilon_t \propto \mathcal{N}(0, \sigma^2)$ with zero mean and standard deviation σ. In this way the fundamental value is an exogenous stochastic process (a critique to this approach is found in [44]).

We later introduced traders which take into account information about the evolution of the price, namely the *moving average* over certain horizons of time [15]. These are the traders that we name *chartists*.

To model how the decisions of agents are influenced by their mutual interaction, we assume that the decision process undergoes a proportional voting where agents occupying the same lattice site express their preference. The single agent's decision is weighted by its *influence strength* to form the collective decision.

Each agent i is defined by a set of attributes. The buy/sell order at time t is given by $x_t^{(i)} \in \{-1, 0, 1\}$ which indicates respectively the decision to sell, to be inactive or to buy for the current step. The capital $c_t^{(i)}$ of agent i at time t is the current amount of money or credits. Each agent starts with an equal capital of money $c_0^{(i)}$ with which (s)he can buy stocks. Agents reinvest their profit and accumulate capital. The number of stocks owned by agent i is indicated by $n_t^{(i)}$. The initial number of owned stocks $n_0^{(i)}$ is randomly chosen for each agent. At each time step, the stocks account for the current *wealth* of the trader i, indicated by $w_t^{(i)}$, as the current nominal price of the asset, indicated by p_t. Thus the current wealth of trader i at time t is $w_t^{(i)} = n_t^{(i)} p_t + c_t^{(i)}$.

The current price p_t is determined by a single market maker. While the traders submit orders $x_t^{(i)}$, the market maker fixes the new price p_{t+1} according to a certain function of the excess demand $D_t = \sum_i x_t^{(i)}$. For the sake of simplicity, we set the price change proportional to the excess demand

$$p_{t+1} - p_t \propto D_t \quad \text{with} \quad D_t \equiv \sum_i x_t^{(i)}. \tag{6.1}$$

The initial setup of the model is given by N_N noisy traders, N_F fundamentalist traders and N_C chartist traders displaced on a $L \times L$ lattice. Each trader starts with a capital $c_0^{(i)}$ and a number of stocks $n_0^{(i)}$ both drawn from uniform distributions.

At each time step the traders decide to trade or stay inactive respectively with probability a and $1-a$. The parameter a corresponds to the *activity* as in [33]. Once each trader determines if (s)he stays inactive or not, the decision to buy or to sell is reached in two phases: (1) first, each agent makes his choice according to his actual situation and potential benefit from the activity as specified by the trading strategy in section 6.1.1 (she/he looks at her/his micro-state, defined by the availability of capital $c_t^{(i)}$ and/or stocks of the asset $n_t^{(i)}$, the price history (if chartist) and the fundamental value (if fundamentalist)); (2) then in the second phase, the individual preferences are summed up in a kind of proportional voting and the resulting "collective" decision determines the probability to follow the majority agreement. If a trader can not fulfill the constraints of money/stocks availability (s)he ignores the collective agreement and stays inactive.
The trading phases are described in the following two sections.

6.1.1 Trading strategy

This section describes the logic which rules the trading activity. At each time step each agent decides with probability a and independently from the others, if to be active or not to trade. For active traders, a different decision path is followed depending on the class.

Fundamentalist strategy Fundamentalists take into account the reference value f_t. They buy if $p_t \le f_t$ or sell if $p_t > f_t$. Their activity can be seen as a global elastic force attracting the price to the fundamental value [28]. The fundamentalist behaviour represented as a SFSM is sketched in figure 6.1.1.

Noisy strategy Noisy traders buy randomly (probability $\frac{1}{2}$) but sell only if convenient. The average price they paid for all the stocks they own must be lower than p_t, i.e.

$$\frac{1}{n_t^{(i)}} \sum_{j=1}^{n_t^{(i)}} p_{t_j} < p_t \text{ with } t > 1 \text{ and } t_j < t \ \forall j.$$

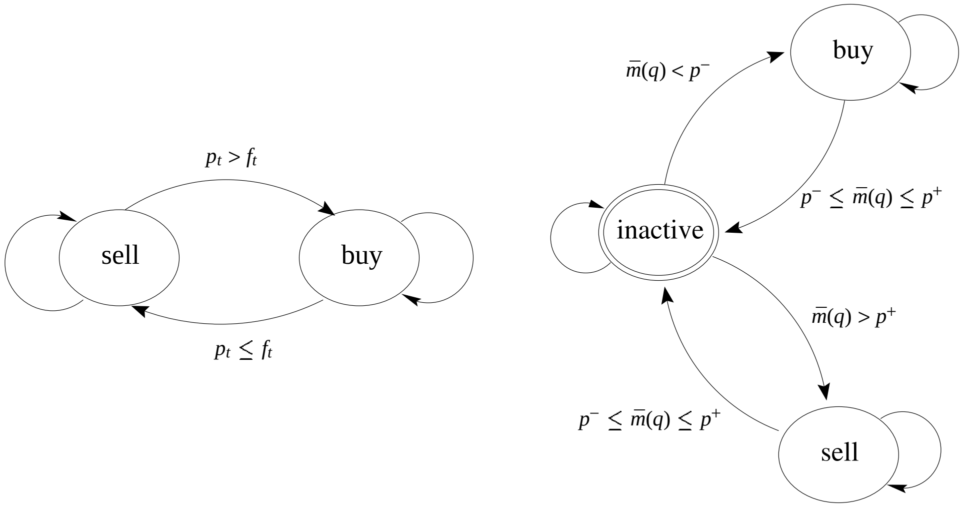


Figure 6.1 Sketch of the fundamentalist (left) and chartist (right) trading rule as finite state automaton.

Chartist strategy Chartists consider the trend with respect to a given time horizon $h \in \mathbb{N}$. At each time step they sell if and only if the moving average value

$$\bar{m}_t(h) = \frac{1}{h} \sum_{t'=t-h}^{t-1} p_{t'} \tag{6.2}$$

computed over the time horizon h is above the “filtered” price $p_t^+(\delta) = p_t + \delta p_t$ ($0 < \delta < 1$ is an input parameter). If $\bar{m}_t(h)$ is below the filtered price $p_t^-(\delta) = p_t - \delta p_t$, they buy. Finally they stay inactive when $p_t^-(\delta) < \bar{m}_t(h) < p_t^+(\delta)$. The chartist behaviour represented as a SFSM is sketched in figure 6.1.1.
We allow three values for h, 10, 60 and 360, so there are three groups of chartists who look at moving averages on different time horizons (respectively short, middle and long term).

6.1.2 Collective formation and diffusion

At this stage of formulation the agents follow independent strategies. Thus, the excess demand D_t of eq(6.1) is the sum of *i.i.d.* random variables and the central limit theorem determines a Gaussian distribution of the histogram of log-return which is different from what is observed in the real markets [81] (see also chapter 4).

To obtain a deviation from Gaussian we need to take into account the *herding behaviour* that has already been demonstrated to determine (at least in part) the fat tail property of the distribution of returns [33, 115, 32, 112].

Instead of determining a priori the clusters distribution as in percolation models we just allow the agents to diffuse on the grid and to aggregate inside each single lattice site .

We model the market as an unbounded lattice gas. Agents are placed randomly on a regular triangular two-dimensional lattice (six neighbour sites) $L \times L$ (six links at 60°) with toroidal boundary conditions (see figure 2.1 in chapter 2). At each time step, agents diffuse uniformly to a neighboring site. The diffusion determines a re-shuffling of agents inside each single lattice site and changes the impact on the price fluctuations.

In the spirit of Nowak *et al.* [87] (see also [32]) the decision of each agent affects and gets affected by other agents on the basis of its influence strength. The influence strength of agent i is represented as a real number $s^{(i)}$. If we assign a much larger influence to some traders among the totality (call them *leaders*), the dynamics of the model will be dependent on how many leaders are present. Leaders are chosen uniformly among all the classes of traders. We represent a *group* by those agents contained in the same lattice site. So, each group of traders forms a collective system. A possible example is a group of people following the advises of a single financial analyst. We also want a group to be dominated by a *leader* so that we can set the number of leaders equal to L^2 given that $N \gg L^2$ where N indicates the total number of agents.

Each agent belonging to a group imposes his/her strategy according to the influence strength $s^{(i)}$. For each $x_t^{(i)} \in \{-1, 0, +1\}$ determined by the trading strategy discussed in section 6.1.1, we sum the influence strength of each agent that is following strategy $x = -1, 0, 1$ and we normalize it to the total so to get values between zero and one to be interpreted as probabilities. The final decision of selling, staying inactive or buying, represented respectively by $-1, 0, 1$ is determined by the following distribution ($\sum_x pr[x] = 1$):

$$pr[x] = P[x_t^{(i)} = x] = \frac{\sum_{j:x_t^{(j)}=x} s^{(j)}}{\sum_j s^{(j)}} \tag{6.3}$$

where the index i runs over all agents of the group. The influence strength of the leader is larger than those of a “regular” trader. How much larger is specified as an input parameter. Given the distribution $pr[-1]$, $pr[0]$ and $pr[+1]$ of eq(6.3), the agents’ strategy $x^{(i)}$ is updated using a *random wheel selection*.

A collective system arises from the coherent behaviour of a group of strongly interacting constituents. It also has a weak external coupling. In our case, the collective can be treated as an individual whose strategy’s distribution is determined by the single agent’s actual strategy $x^{(i)}$ through eq(6.3). This means that if exactly one agent is a leader and he has chosen strategy $\hat{x}$ then, for the other choices x, $pr[\hat{x}] \gg pr[x]$. Thus, the fraction of agents in that collective that will follow strategy $\hat{x}$ is large. If two leaders are present in the same site and their decision is different, then they will compete to determine the majority strategy. On average half of the traders will follow one leader and the other half will follow the other. And so on for the other possibilities. In general, if we set more than one single leader in each site (on average), their competition will destroy the effects of the herding behaviour. This determines, as confirmed by numerical simulations, a Gaussian distribution of returns. In conclusion we decided to set on average a single leader (or less) in each lattice site.

It is worthwhile to note that even in this case the dynamics of each collective group is equivalent to that of a single agent (the leader) with combined capital and the collectives take decisions independently one from the others. Following this reasoning, for the central limit theorem, one would expect again a Gaussian distribution of return. In fact the effect of the collective strategies of the leaders with combined capital would end up to zero when summed over the whole grid just because the uniformity of the collectives' dimension which is about $(N_C+N_F+N_N)/L^2$ on average. The results show a clear deviation from a Gaussian instead. Why this happens?
The answer is found considering the synergy between the trading rule of section 6.1.1 and the collective formation mechanism described in 6.1.2; the collective groups influenced by a fundamentalist leader are coupled (weakly, but they do) by means of the fundamental price that is perceived by all of them equally. So, while the *noisy' collectives*, being totally uncoordinated, only produce noise, it happens that fundamentalists' leaders occasionally end up with the same decision to buy or sell, driving a large fraction of all the traders to follow the same decision. Same rationale applies to the chartist' collectives. Chartists look at the price history (the "charts") and eventually end up to follow the same "trend". This behaviour of fundamentalists and chartists drives a large fraction of agents to take the same decision causing large fluctuations of price, that is, fat tails in the distribution of return.

The re-shuffling given by the random diffusion of agents on the lattice can also be interpreted as a change of preference, in the same way people decide to trust to another brokerage agency or bank. In fact, one agent that leaves a group whose leadership is for example fundamentalist for a group whose leadership is chartist will end up in a *behaviour's change* much like the fundamentalist-noisy switch in [77]. Moreover, not all lattice sites will contain a leader. In these "leader free" sites no collective if formed and the behaviour of the agents is almost independent.

6.2 Discussion

When the number of fundamentalists is higher than that of the noisy traders, the price p_t tracks closely the perceived value f_t. This is both trivial and unrealistic. In

fact, evidence from real market data suggests that, while prices track values over the very long term, large deviations are the rule rather than the exception [22]. The opposite situation results in too random fluctuations given the random behaviour of the noisy.
A non trivial dynamics is obtained for $N_F \simeq N_N$. In this case we observe periods in which the price follows the fundamental price followed by periods of apparently independent fluctuations. The influence of the chartists on the price dynamics is quite different and will be discussed in section 6.2.1.
We set $N_F = N_C = N_N = \hat{N}L^2$ for a certain $\hat{N} \in \mathbb{N}$. The value of this parameter is found observing that the collective dynamics depends on its dimension. If the collective systems are too small we do not get any herding behaviour.

We can also get rid of another parameter and set the activity $a = a_0/(N_F + N_C + N_N)$ for a certain fixed a_0 so to scale with the system size.

Figure 6.2 refers to a simulation with $L^2 = 400$ and $\hat{N} = 25$ for a total of 30000 agents. The influence strength of the leader is set to a hundred times bigger than that of normal ones. The diffusion speed is set to 10^{-1} meaning that each trader changes group every ten time steps on average. As in [23] half of the agents start with one stock while the remaining with no stocks. In this way we obtain a balance between an initial number of people willing to sell and people willing to buy.

It is worthwhile to note that the initial amount of capital each agent is equipped with, induces the ability of the market price p_t to follow the fundamental value f_t when this strongly deviates from the initial value f_0. To see this fact just consider the case in which the agents own little initial capital, then p_t is limited by the global capacity of the agents to buy. On the other hand, the price p_t is limited from below by the amount of initial stocks we equip the agents with; the greater it is, the larger will be a potential fall of the price p_t to follow the fundamental value f_t. These and other related questions will be investigated elsewhere.
Another consequence coming from the constraint given by the availability of funds and stocks is that, given the trading rule described above, the activity of the agents is not uniform. In fact, agents may end up with the decision to stay inactive either if they want to sell but they do not own stocks or they want to buy but they do not have money.

Figure 6.2 shows the price p_t to follow the fundamental price f_t apart of some large occasional deviations. The standard deviation of the price is $\sigma_p = 46$ while that of the fundamental value is $\sigma_f = 44.94$, giving an *excess volatility* of 2.3%. In the small chart at the bottom of the same figure we show the traded volume V_t computed each time step (in contrast to reality where it is computed every certain period of time, e.g. day or week but not instantaneously). Because in this model

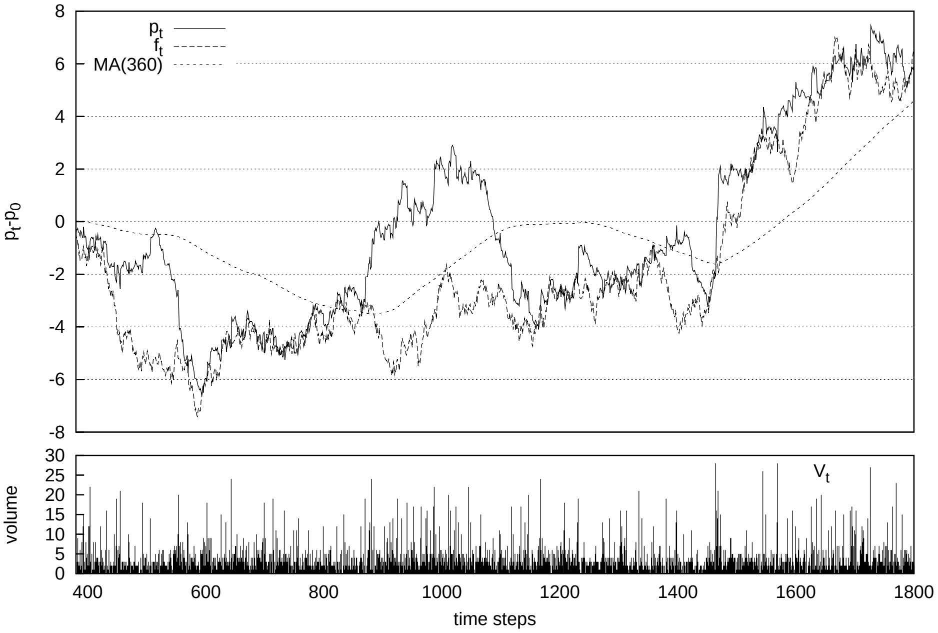


Figure 6.2 Variation from the initial price in percent. Price (up), volume (down). The volume V_t is defined as in eq(6.4). MA indicates the moving average on long term.

we do not require to match the sell/buy orders, we compute the volume as

$$V_t = \sum_i |x_t^{(i)}|. \tag{6.4}$$

The model does not require that each buy order should match a sell order because the balance is assured to be made involving a market maker outside the model itself.

The figure shows an increase in volume when $p_t \simeq f_t$ because, according to the trading rule, fundamentalists trade much more in proximity of the fundamental value. This is opposed to periods in which they either (a) sell all stocks they own because the price is higher than the perceived value and wait to buy a new stock when the price is lower than its expected value or (b) they have already invested all the capital and cannot buy other stocks even if appropriate (see also plots of wealth 6.3).

The dynamics of wealth per agent's type is depicted in figure 6.3. This collective analysis does not show realistic behaviour as the fundamentalists tend to trade to compensate the effect of the noisy traders on price dynamics (see fig. 6.3 plots (a) and (b)). On the other hand, the dynamics of the chartists seems uncorrelated

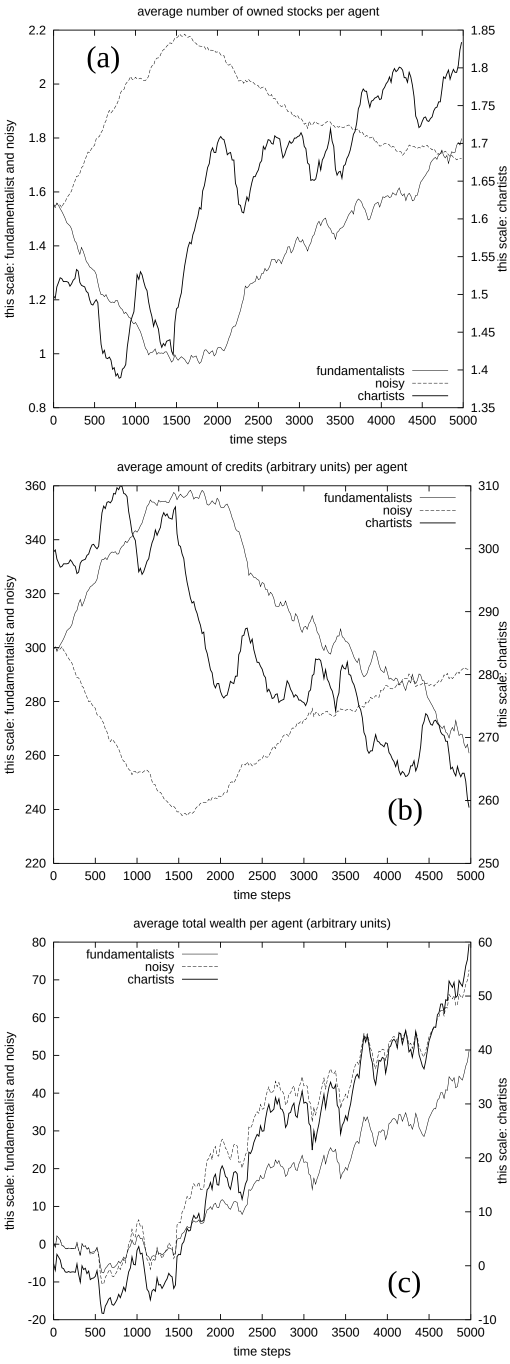


Figure 6.3 Average number of stocks (a) $1/N_e \sum_{i=1}^{N_e} n_t^{(i)}$ and average amount of cash (b) $1/N_e \sum_{i=1}^{N_e} c_t^{(i)}$ for each trader type during the simulation, where N_e is the number of agents per type e. Plot (c) shows the difference with the initial wealth given by both liquidity and nominal price of the owned stocks $1/N_e \sum_{i=1}^{N_e} w_t^{(i)}$

with the rest of the agents. Different simulations have not shown altogether a clear advantage of one strategy with respect to the others.

To better investigate the final distribution of capital we run a large simulation involving two million agents and running for 15000 time steps. In figure 6.4 is shown the histogram of agent's wealth-change distribution. It shows a power law in the central part with slope -1.3 and wide tails. The central part of the distribution is consistent with a Pareto-like distribution [91]. Similar questions have been investigated with the Cont-Bouchaud model [71]. It is noticeable the fact that the initial uniform distribution (all agents start with same capital) is strongly reshaped over a sufficiently long run. Indeed over shorter run the final distribution of wealth is Gaussian-like (not shown).

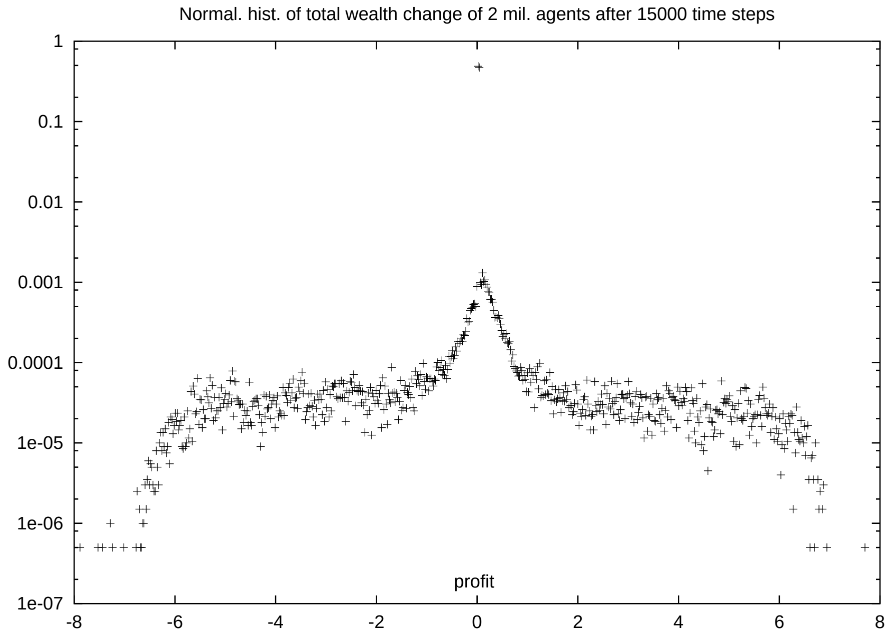


Figure 6.4 On the X-axis the difference between the final accumulated capital per agent and the initial capital they are equipped with. A re-shaping from a uniform distribution (at the beginning they have the same capital) to a power law in the center of the histogram is observed (Y-scale is logarithmic). The unit for the X-axis is given in *Euro* or in an arbitrary unit of money.

Plot 6.5 shows the histogram of log-return r_t. The excess kurtosis κ of the distribution is 4.58. Moreover, the histogram has fat tails and power law decay $\propto |x|^{-\alpha}$ in the central part leading to exponent $\alpha \simeq 2.8$ (see inset plot) roughly consistent with empirical studies [77, 50].

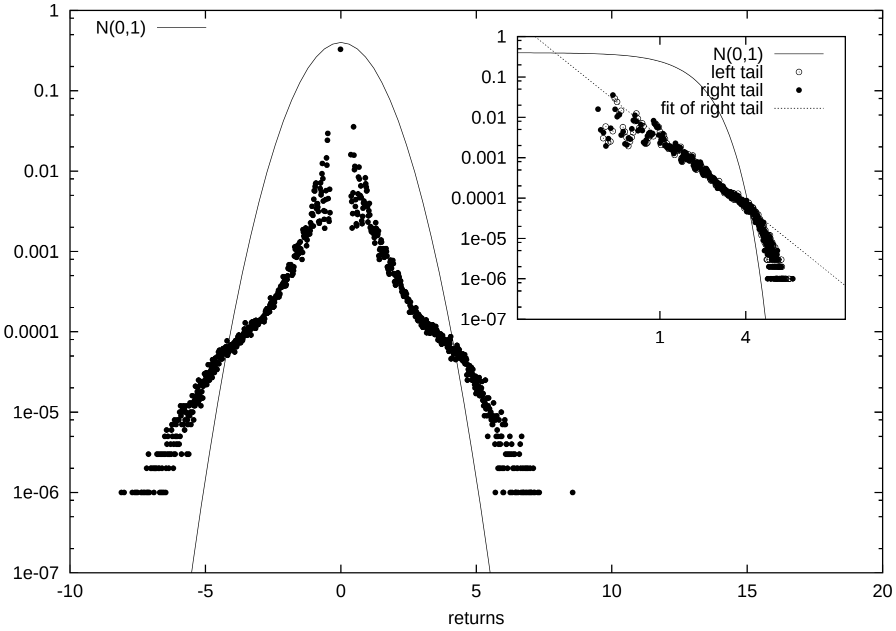


Figure 6.5 Histogram of *standardized* price returns $(r_t - < r_t >)/\sigma_{r_t}$. The excess kurtosis is 4.58. In the log-log inset plot the fit of the central part of the histogram have slope $\simeq 2.8$.

Figure 6.6 show the comparison of the return of the market price p_t and those of the fundamental price f_t. The deviation is clear; the exogenous source of information is transformed into something else by the endogenous dynamics of the market participant as already demonstrated in [77].

Another relevant property of market price dynamics is the absence of correlation of return and the persistence of long range correlation of volatility [77, 50]. Volatility of stock price changes is a measure of how much the market is liable to fluctuate and can be defined in different ways. In the following we define the volatility as the square of return $v_t = r_t^2$.

Figure 6.7 shows the autocorrelation function $c(\tau)$ of volatility v_t defined as in eq(4.1). Empirical studies report a power law decay with exponent between 0.1 and 0.3 for the autocorrelation of volatility in real data [18, 82, 48]. Instead we found a slope $\simeq .013$ that is one order of magnitude less.

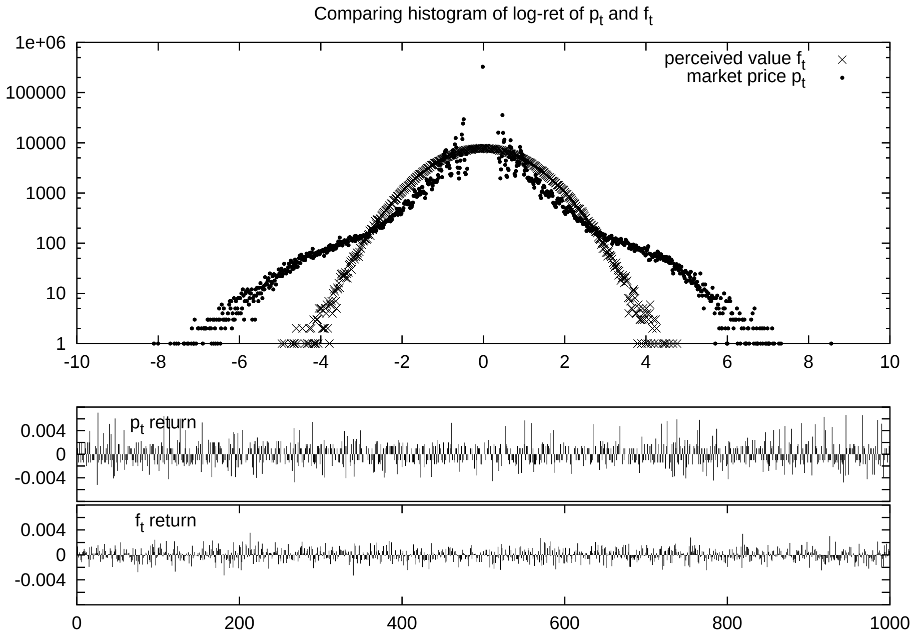


Figure 6.6 Comparing the log return of f_t with that of p_t. The exogenous source of information is transformed into something else by the endogenous dynamics of the market participants. On the X-axis of the upper plot it is reported the price return while on the two plots at the bottom is reported the time step.

6.2.1 Chartist's influence on price dynamics

Empirical studies of large data bases [73] show that the cumulative distribution of the volatility is consistent with a log-normal behaviour, at least for the central part of the histogram. We use here another definition of time average volatility that has also being used in [73]

$$v_t = \left(< r_t^2 >_{\Delta t} - < r_t >_{\Delta t}^2\right)^{1/2} \tag{6.5}$$

where $< \cdot >$ indicates the average over a fixed time interval Δt. This definition of volatility coincides with the estimate of the standard deviation of the log-return on the time interval Δt.

As already stated, the model uses the collective strategy of section 6.1.2 to reproduce imitation or herding behaviour. Although imitation is enough to reproduce fat tails in log-return distribution, it fails to explain the log-normal volatility distribution.

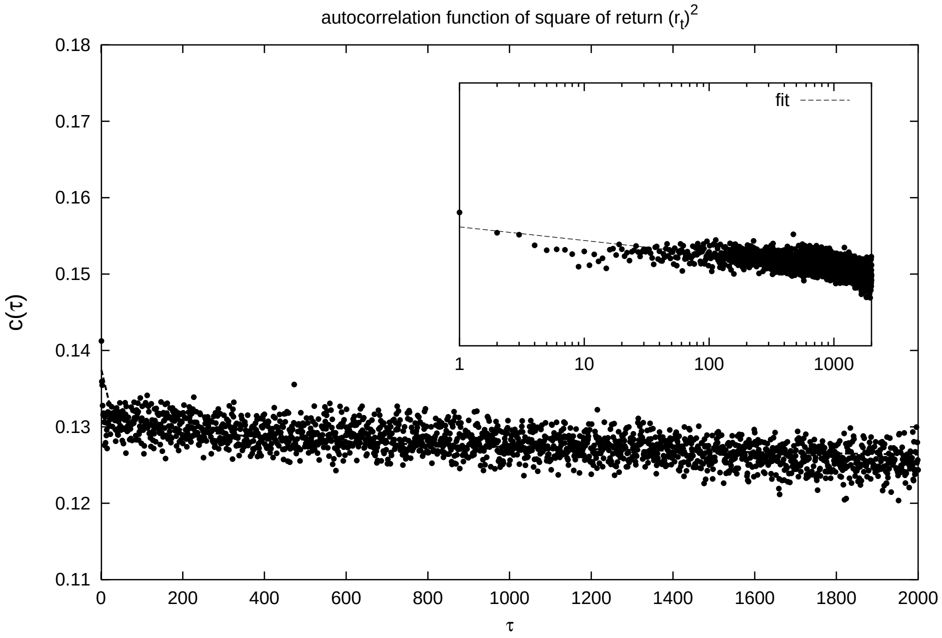


Figure 6.7 Autocorrelation function of volatility of return (eq(4.1)). It shows the correlation of volatility persisting for long time. In the log-log plot the fit has slope -0.013.

To test this conjecture we performed four different runs where in two of them the chartists were absent and in other two the aggregation (collective strategies) was switched off. Figure 6.9 shows the histogram of volatility for simulations with 15360 agents on 16×16 lattice sites, $a = 0.001302$ and $\delta = 3\%$. Figure 6.9 clearly shows that when no aggregation is present the histogram of volatility can not be fitted with a log-normal curve. Same pattern when no chartists are present (same figure panel (b)). Instead, a good fit with a log-normal distribution

$$\frac{1}{\sqrt{2\pi}\sigma x} \exp\left[-\frac{(\log x - < x >)^2}{2\sigma^2}\right]$$

with parameters $\sigma = 0.371$ and $< x >= -6.620$ is found for the run with both aggregation and chartists behaviour. This result is consistent with the empirical findings on the S&P500 [73] only for the central part of the histogram (see figure 6.10). Other empirical studies of large data bases [73] show right tails of the volatility distribution of S&P500 consistent with a power-law asymptotic behaviour characterized by an exponent $\simeq 3$. This power-law right tail is not recovered here. Nevertheless it is worthwhile to mention the work in [92] where a quite good fit with a log-normal distribution is indeed found for the NYSE index.

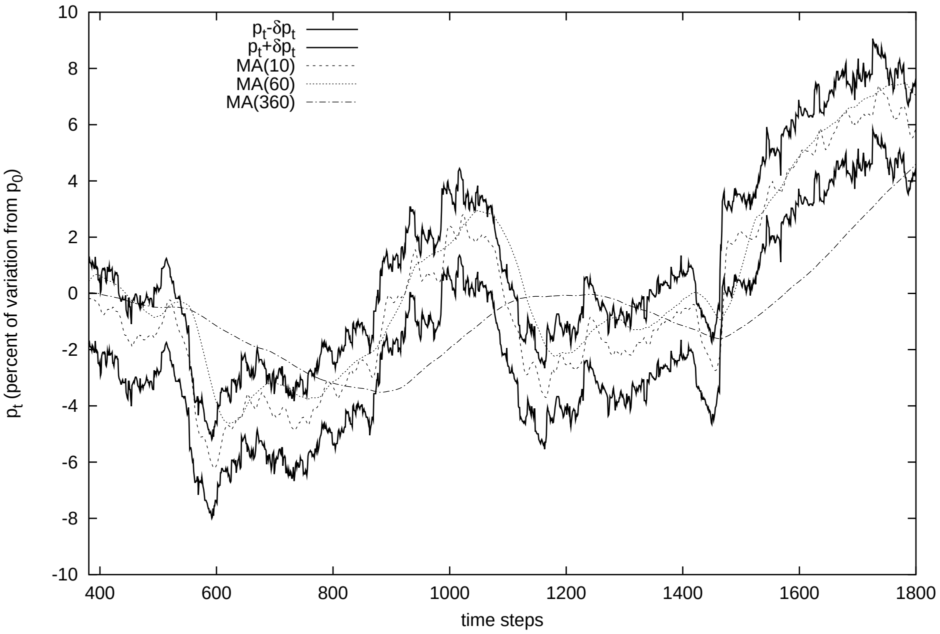


Figure 6.8 Variation from the initial price (in percent). Indicated in figure the filtered price $p_t - \delta p_t$, $p_t + \delta p_t$, and the three moving averages $\overline{m}_t(10)$, $\overline{m}_t(60)$ and $\overline{m}_t(360)$. Large fluctuations are provoked by the $\overline{m}_t(h)$ crossing the price $p_t \pm \delta p_t$. Positive and negative trends are visible.

In general, a log-normal distribution predicts that large "positive" (i.e. greater than the average) jumps (fluctuations) are more frequent than "negative" ones. Our simulations show that a good fit is doable only for the histogram of the run with chartists revealing a positive effect of chartist's trading on the overall dynamics.

The effect of chartists on the price dynamics can be rationalized as follows. According to the trading rule in 6.1 chartists do not influence the price as long as the moving average $\overline{m}(h)$ defined in eq(6.2) stays in the range $(p_t - \delta p_t, p_t + \delta p_t)$. When $\overline{m}(h)$ hits the upper or lower border of the filter the chartists take a position: buy if it goes over $p_t + \delta p_t$ or sell in the other case (see fig. 6.8). Trends are clearly visible as well as large fluctuations (crashes?) in proximity of the points in which one (or more) moving average crosses the filtered price $p_t \pm \delta p_t$.

Let's have a look at the consequences of such behaviour in closer detail. Chartists are equally distributed in three classes, as many as the time horizons over which the moving averages are computed, $N_C = N_{C_{10}} + N_{C_{60}} + N_{C_{360}}$. Let's take for example the simple case in which two different time horizon are computed, $h1 \neq h2$. Also call $N_{C_{h1}}$ and $N_{C_{h2}}$ the number of chartists respectively. Now consider

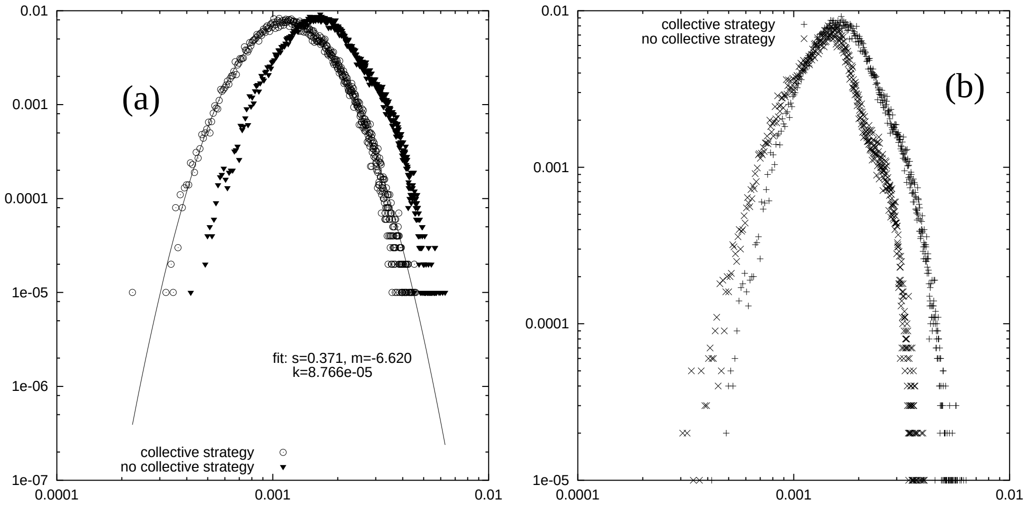


Figure 6.9 Volatility histogram (log-log scale): figure (a) corresponds to runs with chartist traders; figure (b) without chartist traders. Runs of 10^5 time steps. One of the two histograms of each figure indicates a run without imitation or herding behaviour. In plot (a), s indicates the standard deviation and m the mean value of the log-normal fit.

the price following a descending path, whatever has provoked it. What happens is that, a sharp decrease of the price will induce $\overline{m}(h1)$ to decrease first (just to make an example). At a certain point $\overline{m}(h1) > p_t^+(\delta)$ calling $aN_{C_{h1}}$, on average, for a sell order. This synchronous signal to all the chartists $N_{C_{h1}}$ will bring the price to fall accordingly. Eventually, at a later time t' also $\overline{m}(h2) > p_{t'}^+(\delta)$ calling for a second wave of sell orders by $aN_{C_{h2}}$.

This reasoning could be extended at will according to the number of different time horizons considered to compute the moving average. This kind of *domino effect* is valid also the other way around for a rising price, inducing positive trends. Figure 6.8 shows the filtered price with three moving averages computed over time horizons 10, 60 and 360. The corresponding price and volume is shown in figure 6.2.

6.3 Long-term dependencies: Hurst exponent and modified R/S statistics

Evidence of positive correlations in the magnitude of the return (volatility clustering) in real time series in a well accepted and documented fact. The modified

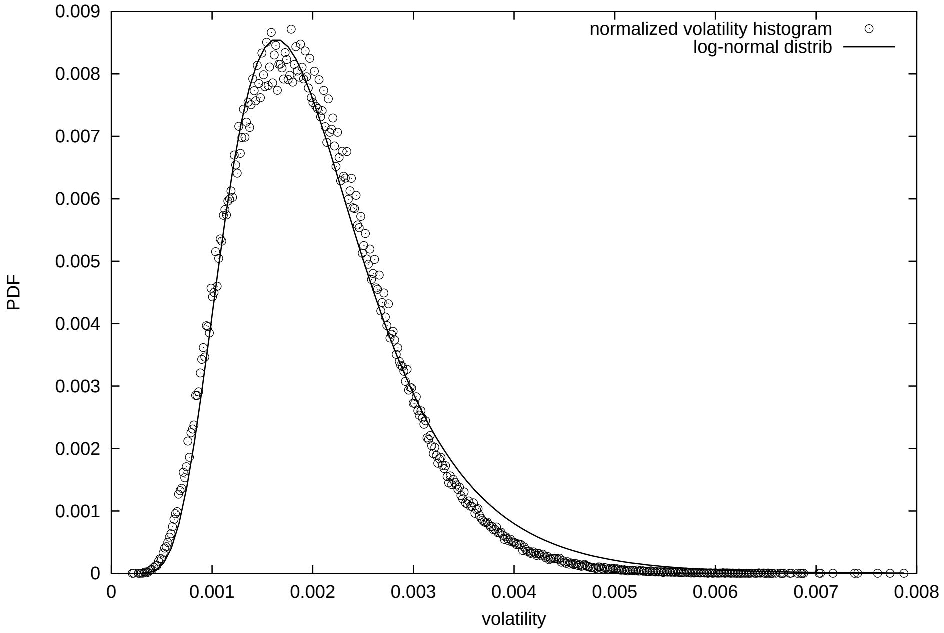


Figure 6.10 Histogram of volatility. The histogram is well fitted in the central part by a log-normal distribution.

R/S statistics (MRS) proposed by Lo (1991, [74]) is a modification of the Hurst-Mandelbrot rescaled range (R/S) statistics [52, 79]. Lo generalized the R/S statistics as he suggested the R/S is sensible to short-range dependencies. Thus, evidence of dependencies using the R/S statistics *may* come merely by short-term dependencies and *not* from the long-term ones.
Given a time series of N values $X_1, \ldots, X_N$, the MRS statistics is defined as follows:

$$Q_n \equiv \frac{1}{\hat{\sigma}_n(z)} \left[\max_{1 \leq k \leq n} \sum_{j=1}^{k} (X_j - \overline{X}_n) - \min_{1 \leq k \leq n} \sum_{j=1}^{k} (X_j - \overline{X}_n) \right] \tag{6.6}$$

where $\overline{X}_n$ is the sample average,

$$\begin{aligned} \hat{\sigma}_n^2(z) &\equiv \frac{1}{n} \sum_{j=1}^{n} (X_j - \overline{X}_n)^2 + \frac{2}{n} \sum_{j=1}^{z} w_j(z) \left\{ \sum_{i=j+1}^{n} (X_i - \overline{X}_n)(X_{i-j} - \overline{X}_n) \right\} \\ &= \hat{\sigma}_X^2 + 2 \sum_{j=1}^{z} w_j(z) \hat{\gamma}_j . \end{aligned} \tag{6.7}$$

$\hat{\sigma}_X^2(z)$ is the sample variance estimation of X computed over n samples and $\hat{\gamma}_j$ are the auto-covariance estimators weighted by $w_j(z) \equiv 1 - j/(z+1)$ for $z < n$. Equation 6.6 computes the range of partial sums of deviations of a time series X_t from its mean $\overline{X}_n$, rescaled by $\hat{\sigma}_n(z)$.
To choose z in eq(6.7) is not a simple problem. The parameter z is called the "truncation lag" and must be chosen with some consideration of the data at hand [7]. A simpler solution compared to that in [7] is given by Phillips [96]. Following his advice we take $z \propto o(N^{1/4})$. Note that when $z = 0$ the MRS statistics is identical to Mandelbrot's R/S.

In our case the time series X_t represents the series of volatility defined as the absolute value of the log-return, $|r_t|$. Our sample consists of $N = 10^6$ points. The procedure to compute the MRS statistics is the following: (i) compute Q_n over N/n non-overlapping intervals of size n; (ii) the value of Q_n is computed as the average value over the N/n non overlapping intervals; (iii) the procedure is repeated for different value of $n = 2^k, k = 6, \ldots, 15$.
Plotting $\log Q_n$ against $\log n$ should show the slope H beyond large n. The slope H is called *Hurst correlation coefficient* by the name of the hydrologist H.E. Hurst who first devised this method [52] in his studies of river discharges.

Figure 6.11 shows $\log Q_n$ against $\log n$ computed for a run with $L^2 = 400, N = 18000$ on 3×10^5 time steps. The slope of the fit in the range $\log n > 6$ is 0.79 revealing long-range positive correlations.

6.4 Conclusions and future developments

We have described a new model to reproduce the price fluctuations of a single stock in an artificial stock exchange whose traders are modeled as chartists, fundamentalists and noisy. Some of them have more influence than others and represent a brokerage agency where people go to ask for advice. They group together inside each lattice site to form a collective system and to follow a common strategy according to proportional voting. Traders are free to diffuse on a two-dimensional lattice to model the tendency to change opinion and to follow a different advisor.

The model is consistent with fat tails of histogram of returns, log-normal distribution and clustering of volatility.

The structure of the model is versatile enough to allow future expansion. We believe that a more realistic description of the agents behaviour and trading may allow to get further insight in the dynamics of price change as well as in the distribution of wealth among traders.

Agents should be able to buy/sell more than just one stock at a time according to, for example, the availability of capital and difference between perceived value and actual price. Besides it is worth (and the model will easily allow it) to develop more

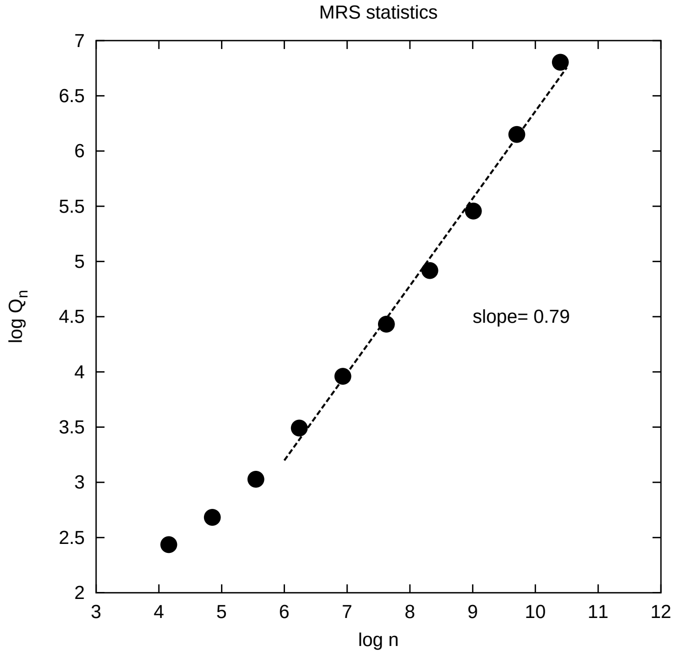


Figure 6.11 $\log Q_n$ against $\log n$ computed over 3×10^5 values for a run with **AMSE**.

realistic trading strategies like trend-following [39] and/or divide the action among different choices (more stocks and/or bonds with fixed income as in [67, 68, 69]).

Chapter

7

Forecasting

Forecasting future values of an asset gives, besides the straightforward profit opportunities, indications to compute various interesting quantities such as the price of derivatives (complex financial products) or the probability for an adverse mode which is the essential information when assessing and managing the risk associated with a portfolio investment.

Forecasting the price of a certain asset (stock, index, foreign currency, etc) on the ground of available historical data, corresponds to the well known problem in science and engineering of time series prediction. While many time series may be approximated with a high degree of confidence, financial time series are found among the most difficult to be analysed and predicted. This is not surprising since the dynamics of the markets following at least the semi-strong EMH should destroy any easy method to estimate future activities using past informations.

7.1 Introduction

Among the methods developed in *Econometrics* as well as other disciplines [1], the artificial *Neural Networks* (NN) are being used by "non-orthodox" scientists as non-parametric regression methods [21, 84]. They constitute an alternative to non parametric regression methods like *kernel regression* [21]. The advantage of using a neural network as non-linear function approximator is that it appears to be well suited in areas where the mathematical knowledge of the stochastic process underlying the analysed time series is unknown and quite difficult to be rationalized. Besides, it is important to note that the lack of linear correlations in the financial price series and the already accepted evidence of an underlying process different from i.i.d. noise point out to the existence of higher-order correlations

---

[1] see the vast bibliography with more than 800 entries at `www.stern.nyu.edu/~aweigend/Time-Series/Biblio/SFIbib.html` reported from [122]

or non-linearities. It is this non-linear correlation that the neural net may eventually catch during its learning phase. If some macroscopic regularities, arising from the apparently chaotic behaviour of the large amount of components are present, then a well trained net could identify and "store" them in its distributed knowledge representation system made by units and synaptic weights [86, 101].

In the following we will see that a well suited NN for each of a set of price time series showing a "surprising" rate of success in predicting the *sign* of the price change on a daily base *can* be found. Not less interesting, we will see that the foretold regularities in the time series seem to be more present on larger time scale than on high frequency data, as the performance of the net degrades if we go from monthly to minutes data.

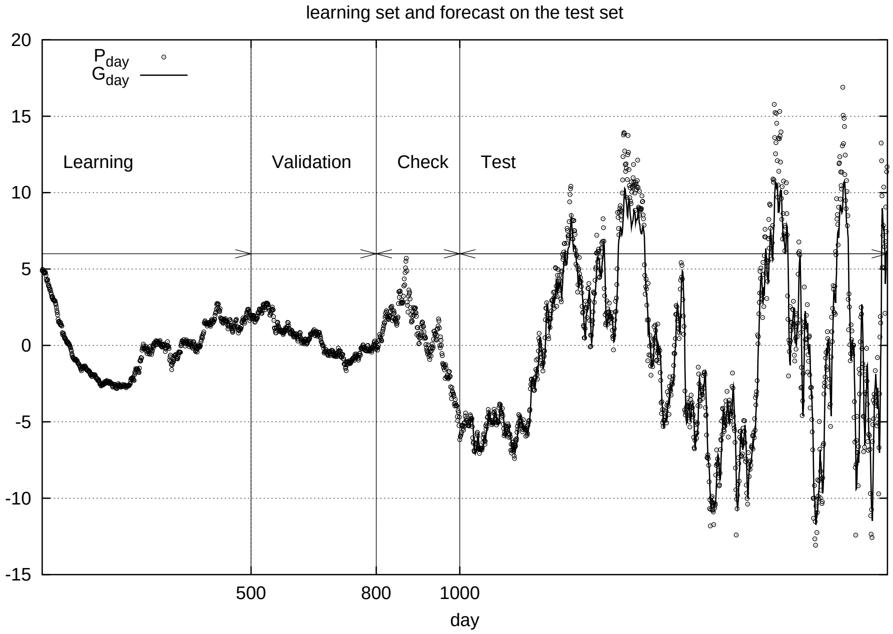


Figure 7.1 Each time series is divided in four data sets: learning, validation, checking and testing (see text for explanation). A difficulty arises from the fact that the oscillations in the test set are much more pronounced than in the learning set. In figure, daily closing price of Intel Corp.

7.2 Multi-layer Perceptron

Multi-layer perceptrons (MLP) are the neural nets usually referred to as function approximators. A MLP is a generalization of Rosenblatt's *perceptron* (1958); n_i

input units, n_h hidden and n_o output units with all feed forward connections between adjacent layers (no intra-layer connections or loops). Such net's topology is specified as n_i-n_h-n_o.

A NN may perform various tasks connected to classification problems. Here we are mainly interested in exploiting what is called the *universal approximation property*, that is, the ability to approximate any nonlinear function to any arbitrary degree of accuracy with a suitable number of hidden units [126, 34].

The approximation is performed finding the set of weights connecting the units. This can be done with one of the available methods of non-parametric estimation techniques like *nonlinear least-squares*. In particular we choose *error back propagation* (EBP) which is probably the most used algorithm to train MLPs [103, 104]. It is basically a gradient descent algorithm of the error computed on a suitable learning set. A variation of it use *bias*, *terms* and *momentum* as characteristic parameters. Moreover we fixed the learning rate $\eta = 0.05$, the momentum $\beta = 0.5$ and the usual sigmoidal [2] as nonlinear activation function.

7.3 Detrending analysis

We have trained the neural nets on "detrended" time series. The detrending analysis was performed to mitigate the unbalance between the *learning set*, and the *test set*. In fact, subdividing the available data in learning set and testing set as specified in the following section (have a look at figure 7.1), we train the nets on a data set corresponding to a periods much back in time while we test the nets on data set corresponding to the most recent period of time. This problem is know in literature as *noise/nonstationarity tradeoff* [84, 86].

It is known that in the 1990's the American market has noticeably changed in that almost all the titles connected to the information technology have not only jumped to record values but also the fluctuations of price today are much stronger than before. [3] Ignoring this fact would lead to a mistake because the net would not learn the characteristics of the "actual situation".

To detrend a time series we performed a nonlinear least squares fit using the *Marquardt-Levenberg* algorithm [21, 98] with a polynomial of sixth degree. Then we just computed the difference of the series with the fitting curve. For each time series considered we ended up with a detrended series composed by 2024 points corresponding to the period from about January 1990 to February 2000. For example, the plot in figure 7.2 shows the detrended time series of the index S&P500 along with the original series and the polynomial fit.

---

[2] the sigmoidal or logistic activation function is $g(u) = 1/(1 + e^{-u})$

[3] p_t is what we use to train our nets. Considering $\log(p_t)$ instead of p_t would mitigate the problem but it would introduce further nonlinearities

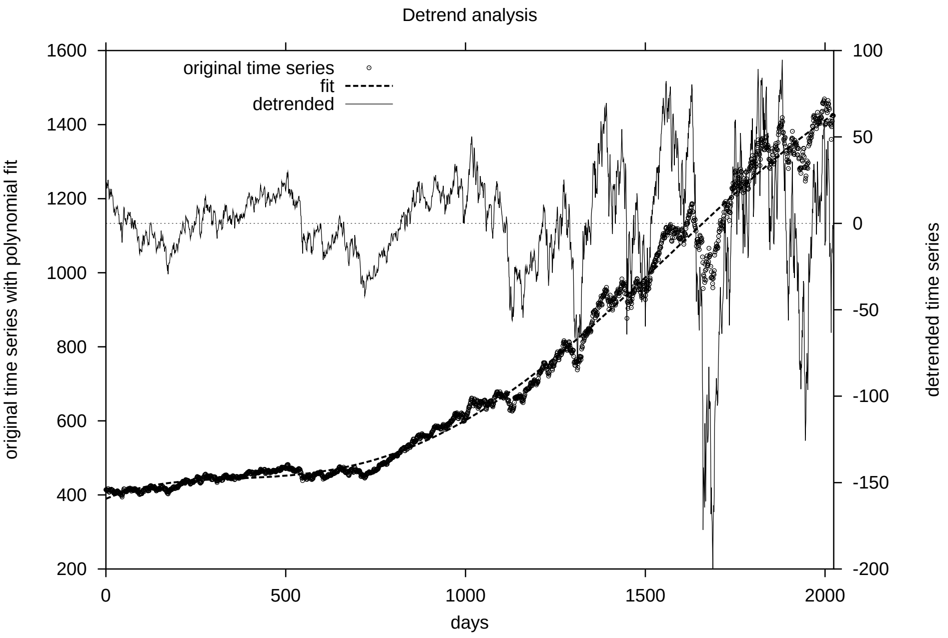


Figure 7.2 S&P500 detrended time series. The plot shows the original series, the polynomial fit and the resulting detrended time series obtained just by difference between the original and the fitting curve. The detrended time series consist of 2024 points.

We choose daily closing of historical series for 3 indices and 14 assets on the NYSE and Nasdaq. In particular the assets were chosen among the most active companies in the field of information technology.

7.4 Determining the net topology

One of the primary goals in training neural networks is to ensure that the network will perform well on data that it has not been trained on (called "generalization" The standard method of ensuring good generalization is to divide our training data into *multiple data sets*. The most common data sets are the *learning L*, *cross validation V*, and *testing T* data sets. (The *checking* set *C* will be explained later in this subsection.) While the learning data set is the data that is actually used to train the network the usage of the other two may need some explanation.

Like the learning data set, the cross validation data set is also used by the network during training. Periodically, while training on the learning data set, the network is tested for performance on the cross validation set. During this testing, the

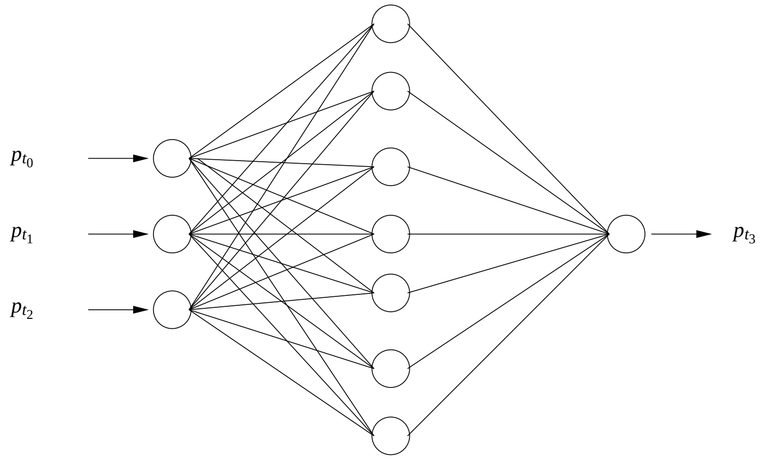


Figure 7.3 A three layer perceptron 3 – 7 – 1 with three inputs, seven hidden and one output units.

weights are not trained, but the performance of the network on the cross validation set is saved and compared to past values. If the network is starting to overtrain on the training data, the cross validation performance will begin to degrade. Thus, the cross validation data set is used to determine when the network has been trained as well as possible without overtraining (e.g. maximum generalization).

Although the network is not trained with the cross validation set, it uses the cross validation set to choose a ”best” set of weights. Therefore, it is not truly an *out-of-sample* test of the network. For a true test of the performance of the network the testing data set T is used. This data set is used to provide a true indication of how the network will perform on new data.

In figure 7.3, an example of MLP with $n_i = 3$, $n_h = 7$ and one output unit takes $p_{t_0}, p_{t_1}, p_{t_2}$ in input and gives the successive value p_{t_3} as forecast. The number of free parameters is given by the number of connections between units $(n_i + n_o) \cdot n_h$.

While the choice of one output unit comes from the straightforward definition of the problem, a crucial question is “how many input and hidden units should we choose?”. In general there is no way to determine apriori a good network topology. It depends critically on the number of training examples and the complexity of the time series we are trying to learn. To face this problem a large number of methods are being developed (recurrent networks, model selection and pruning, sensitivity analysis [84, 86]), some of which follow the evolution’s paradigm (Evolutionary Strategies and Genetic Algorithm).

Because we have observed a critical dependence of the performance of the net from n_i and n_h, and to avoid the great complexity of more powerful strategies [84, 86], we ended up with the decision to explore all the possible combinations of n_i-n_h

in a certain range of values. Our “brute force” procedure consists of training nets of different topologies (varying $2 \le n_i \le 15$ and $2 \le n_h \le 25$) and observe their performance. More precisely we select good nets on the basis of the mean square error (see eq(7.1)) computed on 200 points out of the sample set constituting the test set. Thus, besides the separation in Learning-Validation-Testing of our time series, we further distinguish a subset from the Testing set: the *Checking C* (see fig. 7.1). The reason is that while we train the net to interpolate the time series (minimizing the mean square error) we finally extrapolate to forecast the *sign of the increments* (to be defined later).

To assess the efficiency of the learning and to discard bad trained nets during the search procedure we use the *mean square error* ϱ defined as

$$\varrho = \frac{1}{\sigma} \cdot \frac{1}{|C|} \sum_{t \in C} (g_t - p_t)^2 \tag{7.1}$$

where p_t is the price value, g_t is the forecasted value at time $t \in C$ and σ is the standard deviation of the time series. For good forecasts we will have small positive values of ϱ ($1 \gg \varrho \ge 0$).

We set the threshold 0.015 to discriminate good from bad nets. Only those nets for which $\varrho \le 0.015$ are further tested for sign prediction.

In summary, first we learn on set L, and through validation V we find when to stop learning; then through check on C we see if the learning process worked well, and in case it did, we make predictions in the test phase on set T for ”future” (i.e. previously unused) price changes and compare them with reality.

7.5 Stopping criteria

To avoid overfitting and/or very slow convergence of the training phase, the stopping criteria is determined by the following three conditions, one of which is sufficient to end the training phase (early stopping):

1. Stopping is assured within 5000 iterations of cross validation (see section 7.4);
2. during cross validation the mean square error on the validation set V is computed as $\varrho_V = \frac{1}{2} \sum_{t \in V} (g_t - p_t)^2$; during training ϱ_V should decrease, so a stopping condition is given if ϱ_V increase again more than 20% of the minimum value reached up to then;
3. learning is also stopped if ϱ_V reaches a plateau; this is tested during cross validation averaging 1000 successive values of ϱ_V and checking if the actual value is above this average.

7.6 Results

The plot in figure 7.4 compares the forecasted g_t and the real p_t values for the time series of Apple Corp. on the test set T. It also shows a linear fit for the points $\{p_t, g_t\}$. A raw measure of performance on the test set T can be obtained by the slope of the fitting line (let's call it θ). It will be a value close to one if the fit corresponds to the $y = x$ line, i.e. if $p_t = g_t$. We obtained the following θ's for the time series in table 7.2 and 7.3: $\theta_{\text{S\&P500}} = 0.906$, $\theta_{\text{DJI}} = 0.874$, $\theta_{\text{Nasdaq100}} = 0.860$. $\theta_{\text{AAPL}} = 0.976$, $\theta_{\text{T}} = 0.921$, $\theta_{\text{AMD}} = 0.914$, $\theta_{\text{STM}} = 0.885$, $\theta_{\text{HON}} = 0.885$, $\theta_{\text{INTC}} = 0.874$, $\theta_{\text{CSCO}} = 0.860$, $\theta_{\text{WCOM}} = 0.847$, $\theta_{\text{IBM}} = 0.842$, $\theta_{\text{ORCL}} = 0.824$, $\theta_{\text{MSFT}} = 0.803$, $\theta_{\text{SUNW}} = 0.774$, $\theta_{\text{DELL}} = 0.692$, $\theta_{\text{QCOM}} = 0.488$.

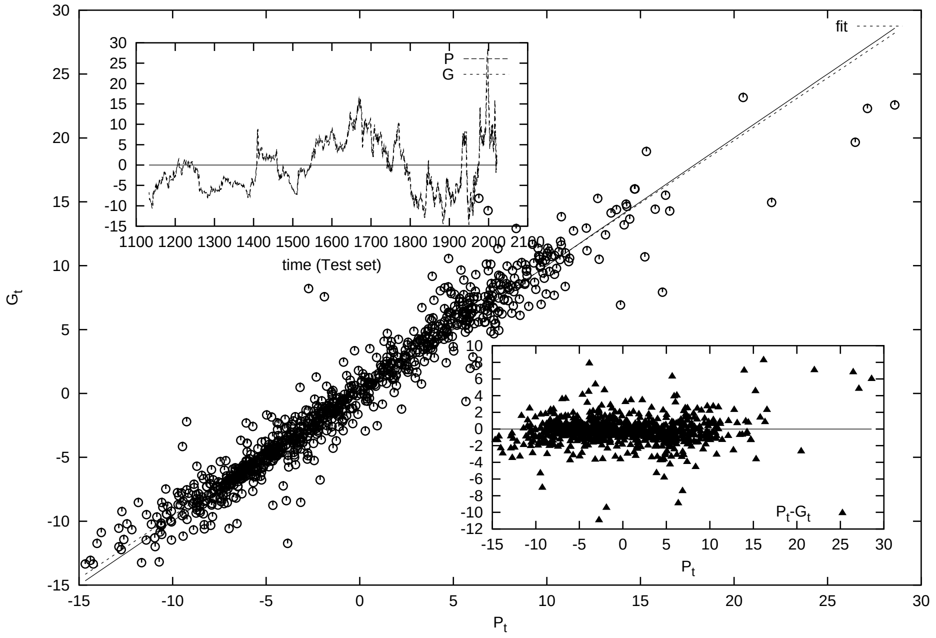


Figure 7.4 Forecast of the time series AAPL. Price is expressed in US$. A perfect forecast will be represented by dots on the $y = x$ line (shown as the continuous line). The dashed line is a linear fit of the points $\{p_t, g_t\}$. A raw measure of the error in forecasting is given by the slope of the fitting line. Values close to one indicate $g_t \simeq p_t$.

The final estimation of the performance in forecasting is made by means of the *one-step sign prediction rate* $\mathcal{S}$ defined on T as follows

$$\mathcal{S} = \frac{1}{|T|} \sum_{t \in T} HS(\Delta p_t \cdot \Delta g_t) + 1 - HS(|\Delta p_t| + |\Delta g_t|) \tag{7.2}$$

where $\Delta p_t = p_t - p_{t-1}$ the price change at time step $t \in T$ and $\Delta g_t = g_t - p_{t-1}$ is the *guessed* price change at the same time step. Note that we assume to know the value of p_{t-1} to evaluate Δg_t. HS is a modified Heaviside function $HS(x) = 1$ for $x > 0$ and 0 otherwise [4]. The argument of the summation in eq(7.2) gives one only if Δp_t and Δg_t are non-zero and with same sign, or if Δp_t and Δg_t are both zero. In other words $\mathcal{S}$ is the probability of a correct guess on the sign of the price increment estimated on T.

In the lower-right inset of figure 7.4 it is shown $p_t - g_t$ as function of p_t. One can see that the difference between the real and the forecasted values clusters for small p_t. Another way see it is to look at the histogram of $\mathcal{S}$ as function of Δp_t. In other words the rate of correct guesses on the sign of the price increment relative to the magnitude of the fluctuation of the real price. To obtain an unbiased histogram we have to normalize it dividing each bin by the corresponding value of the Δp's histogram (the limit of Δp follows a power law so that large fluctuations are much less probable). The resulting distribution is plotted in figure 7.5. It is now clearly visible that the net does not favor large increments over small ones or vice versa. In fact the probability to make a correct guess on the sign of the increment seems independent from the magnitude of the increment itself. This does

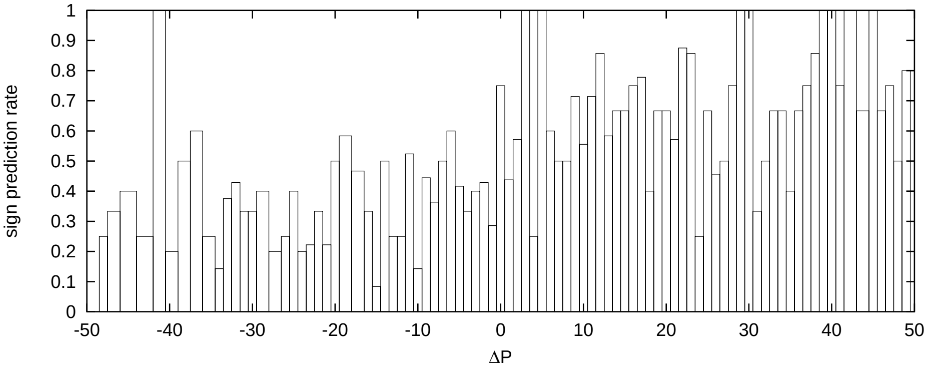


Figure 7.5 Normalized $\mathcal{S}$ as function of Δp (arbitrary units). The the sign prediction rate seems independent from the magnitude of the price change $|\Delta p|$.

---

[4] The usual HS function gives 1 in zero, i.e. $HS(0) = 1$

Series	*ok*/*tot*	Series	*ok*/*tot*
S&P500	32/54	DowJones Ind	189/450
Nasdaq 100	45/86		
SUNW	112/112	DELL	69/69
WCOM	76/76	AAPL	309/311
INTC	46/46	AMD	244/245
STM	33/269	ORCL	35/35
MSFT	21/21	IBM	9/9
CSCO	39/48	HON	22/82
T	6/6	QCOM	43/436

Table 7.1 Here *tot* indicates the number of nets such that $\varrho \geq 0.015$, that is, we judged as good nets, while *ok* is the number of them that gave a sign prediction rate $\mathcal{S}$ above 50 percent.

Symbol	n_i	n_h	$\lvert L\rvert$	$\lvert V\rvert$	ϱ	$\mathcal{S}$(%)
S&P500	8	2	500	300	0.008938	52.272727
DowJones Ind	13	2	700	200	0.012074	51.488423
Nasdaq 100	4	25	700	200	0.014182	50.982533

Table 7.2 For each index the net topology $n_i - n_h - 1$ is specified along with ϱ, $\mathcal{S}$, $|L|$ and $|V|$. $|T| = 2024 - (n_i + |L| + |V| + |C|)$ and $|C| = 200$.

not means that the net forecasts "rare events" (i.e. a profit opportunity) as easily as normal fluctuation, because the statistics here calculated are not significant with respect to extreme events.

To interpret the results that we are going to show we have to concentrate our attention on the way we select a good net to be used to make forecast. For each time series we have performed a search to determine the topology of a good net as specified in the last section. Once we get a pool of candidates the question is "how many of them give a sign prediction rate above fifty percent?"

This question is answered in table 7.1. There, *tot* indicates the number of nets such that $\varrho \leq 0.015$, that is, we judged as good nets, while *ok* is the number of them that gave $\mathcal{S} \geq 50$. This ratio can be seen as an estimation of the confidence that the net will perform a "sufficient" forecast of price change, where sufficient means above fifty percent.

The value of $\mathcal{S}$ together with the specification of the number of units per layer of the best net is reported in table 7.2 and table 7.3 along with the dimension of the learning and validation set.

The sign prediction rates range from 50.29% to 54%. While the smallest values

Company	Symbol	n_i	n_h	$\|L\|$	$\|V\|$	ϱ	$\mathcal{S}$(%)
• Sun Microsys	SUNW	9	7	500	300	0.014435	54.005935
• Dell Computer	DELL	4	18	500	300	0.004315	53.543307
• Mci Worldcom	WCOM	3	2	500	300	0.004024	53.392330
• Apple Comp Inc	AAPL	5	17	700	300	0.013786	53.374233
• Intel Corp	INTC	6	6	500	300	0.009953	53.254438
○ Adv Micro Device	AMD	4	23	500	300	0.012339	52.952756
○ ST Microelectron	STM	6	2	500	300	0.003978	52.465483
• Oracle Corp	ORCL	6	2	500	300	0.006333	52.366864
• Microsoft Cp	MSFT	10	4	500	300	0.008327	52.277228
○ Intl Bus Machine	IBM	10	6	500	300	0.006642	52.079208
• Cisco Systems	CSCO	4	14	500	300	0.008364	51.968504
○ Honeywell Intl	HON	8	2	600	200	0.008506	51.877470
○ AT&T	T	3	22	500	300	0.014920	51.327434
• Qualcomm Inc	QCOM	4	25	500	300	0.009888	50.295276

Table 7.3 Success ratio for the prediction of the sign change. For each asset the net topology is specified along with ϱ, $\mathcal{S}$ and the number of points in the learning and validation set. In the second column is specified the symbols from the respective stock exchange NYSE(○) or Nasdaq(•).

50.29 may be questionable, the larger values above 54 seem a clear indication that the net is not behaving randomly. Instead it has captured some regularities in the nonlinearities of the series.

A quite direct test for randomness can be done computing the probability that such forecast rate can be obtained just by flipping a coin to decide the next price increment. For this purpose we use a random walk ($pr(\text{up}) = pr(\text{down}) = \frac{1}{2}$) as forecasting strategy g_{rw_t} and observe how many, over 1000 different random walks, give a sign prediction rate $\mathcal{S}_{rw}$ defined in eq(7.2) above the value obtained with our net. Note that each random walk perform about 1000 time steps, the same as $|T|$ for that specified time series (see table 7.2 and 7.3). These values are reported in table 7.4. They indicate that except for QCOM the random walk assumption "cannot give" the same prediction rate as the neural net.

In other words, given a neural net which produce $\mathcal{S}$ as prediction rate over a certain time series p_t we may compute the probability at which the *null hypothesis of randomness* is rejected. We use a random walk ($pr(\text{up}) = pr(\text{down}) = \frac{1}{2}$) as forecasting strategy g_{rw_t} and then compute $\mathcal{S}_{rw}$ defined in eq(7.2) on the time series p_t. The random variable $\mathcal{S}_{rw}$ have mean 0.5 and standard deviation $\sigma_{\mathcal{S}_{rw}}$. By definition $\mathcal{S}_{rw}$ is the sample mean of T *i.i.d.* Bernoullian random variables. Thus, assuming that $\mathcal{S}_{rw}$ converges to a Gaussian $\mathcal{N}(\frac{1}{2}, \sigma_{\mathcal{S}_{rw}})$, we can estimate the unknown

Series	$\#rw : \mathcal{S}_{rw} \geq \mathcal{S}$	Series	$\#rw : \mathcal{S}_{rw} \geq \mathcal{S}$
S&P500	78	DowJones Ind	186
Nasdaq 100	258		
SUNW	7	DELL	16
WCOM	13	AAPL	25
INTC	21	AMD	30
STM	50	ORCL	69
MSFT	76	IBM	103
CSCO	98	HON	108
T	194	QCOM	431

Table 7.4 For every sign prediction rate $\mathcal{S}$ reported in table 7.2 and 7.3 it is here shown the number of random walks (over 1000) that have totalized a sign prediction rate $\mathcal{S}_{rw}$ greater or equal $\mathcal{S}$.

variance of $\mathcal{S}_{rw}$ as $\hat{\sigma}^2_{rw} = \frac{1}{N}\sum_{i=1}^{N}(\mathcal{S}_{rw_i} - \frac{1}{2})^2$. To have an estimation of $\sigma_{\mathcal{S}_{rw}}$ we ran $N = 1000$ random walks each giving a value for $\mathcal{S}_{rw}$. Once we estimate σ_{rw}, the null hypothesis becomes “what is the probability $p_{\mathcal{S}_{rw}}[x > \mathcal{S}]$ that the neural net is doing a random prediction on p_t with rate $\mathcal{S}$?” or the other way around ”what is the probability $p_{\mathcal{S}_{rw}}[x \leq \mathcal{S}]$ that the net is not doing randomly?”. In formula, $p_{\mathcal{S}_{rw}}[x \leq \mathcal{S}] = \int_{-\infty}^{\mathcal{S}} \mathcal{N}(\frac{1}{2}, \hat{\sigma}_{rw})(x)dx$ where $\mathcal{N}(\frac{1}{2}, \hat{\sigma}_{rw})$ is a Gaussian and $\hat{\sigma}_{rw}$ is the estimation of the standard deviation σ_{rw} of the random variable $\mathcal{S}_{rw}$. In summary, for every sign prediction rate $\mathcal{S}$ obtained with our neural net on a time series p_t, we first estimate $\hat{\sigma}_{rw}$ as specified above, then we compute the probability $p_{\mathcal{S}_{rw}}[x \leq \mathcal{S}]$ at which the null hypothesis of randomness prediction is rejected. The results tell us that for some bad prediction values (like for QCOM or Nasdaq100) the randomess hyphothesis cannot be rejected but for the majority of the series the probability to reject the null hypothesis is something between 0.01 and 0.1.

7.7 Weekly and intra-day data

It is interesting to ask if the MLP may exploit regularities in the time series of price sampled at a lower/higher rate than daily. Apart from the “scaling behaviour” observed empirically in real price series we are interested in the performance of our procedure (search plus learn) when we change the time scale on which we sample the price of the assets or the index at a stock market.

To answer this question we performed the same search for the good net on the IBM and AMD stock price sampled on weekly basis as well as taking intra-day data with the frequency of one minute. Both series consisted of 2024 points, the same as the daily price series.

The outcome is that intra-day data are much difficult to be forecasted with our MLPs. In fact for both the one-minute-delay data series the search did not succeeded to find a good net; all the good nets (few) have given a sign prediction rate $\mathcal{S} < 40\%$.

On the other hand the forecast of weekly data gave a success rate comparable with that of daily series (e.g. a 4-2-1 net performed $\mathcal{S} = 51.422764$ with $\varrho = 0.004947$).

7.8 Artificially generated price series

As last question, and to further test the correctness of our prediction, we tried to forecast the sign of price changes of an artificially generated time series. This was generated by the the Cont-Bouchaud herding model (see section 5.1 in chapter 5) that seems one of the simplest one able to show fat tails in histogram of returns [33]. This model shows the relation between the excess kurtosis observed in the distribution of returns and the tendency of market participants to imitate each other (what is called *herd behaviour*). The model consists of percolating clusters of agents [109, 108, 115, 32, 112]. At a given time step a certain number of coalitions (clusters) decide what to do: they buy with probability a, sell with probability a or stay inactive with probability $1-2a$. The demand of a certain group of traders is proportional to its size and the total price change is proportional to the difference between supply and demand.

It is clear that such a model generates unpredictable time series, and our networks should not be able to make any predictions. Indeed, when our method was applied to this series it did not succeeded to find a good net as all the tried nets performed bad on the check set C, i.e. $\varrho > 0.015$.

7.9 Discussion

We have shown that a suitable neural net able to forecast the sign of the price increments with a success rate slightly above 50 percent on a daily basis can be found. This can be an empirical demonstration that a good net exists but we do not have a mechanism to find it with “high probability”. In other words we cannot use this method as a profit opportunity because we do not know *a priori* which net to use. Perhaps a better algorithm to search for the good topology (model selection and pruning with sensitivity analysis [84, 86]) would give some help.

As final remark we have found that intra-day data are much more difficult to be forecasted with our method than daily or weekly data.

Summary

The main question addressed above was how to derive "the collective" (i.e. macroscopic) properties of a system starting from the knowledge of the laws ruling the individual (i.e. microscopic) behaviour.
Similar problem gave rise to the field of statistical mechanics although a major difference is present. The macroscopic laws of thermodynamics were derived from the knowledge of the microscopic Newtonian's law of motion which are well understood. In contrast, in finance and in biology and in particular in immunology, the microscopic rules are mostly empirical and nonetheless unable to completely define the system.

In the search for computational models that help to understand the dynamics of complex systems, one can take a great advantage from the impressive acceleration of computer tools and techniques. In fact the very structure of computation on digital computers has inspired the introduction of new class of models (algorithms), where interaction among degrees of freedom are expressed by logical rules acting over a discrete state space – something much closer to "biological language" than to standard (floating point) physical models.

We have presented a unifying approach to model complex systems with large number of degree of freedom. Starting from the definitions of spin systems, with little changes we have defined a new model (called *unbounded lattice gas*) that is well suited to describe two different simulation systems.

The first system (called **CImmSim**) simulates the humoral and cellular immune response. It is based on the original model of F. Celada and P.E. Seiden. This detailed model uses six cellular entities, five molecular entities and various signals like interleukin. Through complex interaction rules among the cells and molecules, the response to a generic antigen or virus is triggered by multiple recognition of the bit-string representing the chemical-physical conformation of the receptors. The computational model **IMMSIM** has been completely re-coded in C language (**CImmSim**) with message passing facilities to run large-scale simulations on parallel machines. Particular attention has been devoted to the efficiency in using both memory and CPU. By means of **CImmSim** we have investigated the complex dynamics of the immune response and described the process of the antigen recognition as a cascade in a suitable (information) state space.

The second system (called **AMSE**) simulates the trading activity of agents in a stock market. It is built borrowing some ingredients from other known models. In addition, a group a traders called moving-average followers represents chartists who base their decision upon the evolution of the price of the asset. Also, traders group together to take collective decision to model the herding behaviour. Simulations reveal the following results: (1) the aggregation behaviour is responsible for the fat-tailed histogram of return; (2) the log-normal distribution and long-range correlation of volatility come from the activity of the chartists following the moving average. Many other features of the market dynamics can be investigated with this model and are actually works in progress.
The architectural schema of **AMSE** is equivalent to the one used for **CImmSim**, i.e. it is also an unbounded lattice gas. Also for this simulation system we have adopted the message passing paradigm to allow large scale simulation.

Two other efforts described in this dissertation are (i) the implementation of the Cont-Bouchaud model using the algorithm of Hoshen and Kopelman as the clustering labeling algorithm and (ii) the realization of a multi-layer perceptron with error-back propagation learning algorithm to forecast financial time series.

Appendix A

Basic immunology

In this appendix is given the cardinal features of the immune system response as needed for chapters 2 and 3. It is subdivided in humoral and cellular response.

A.1 The humoral response

The *humoral* immune response is characterized by the production of antibodies. These are soluble (i.e. free) copies of the B cell receptors which are commonly carried on the cell surface and are responsible for the antigen binding and recognition.

According to the clonal selection theory, the lymphocytes B that proliferate during the antigen attack are those whose receptor is able to bind the antigen. During the proliferation, the B lymphocytes differentiate into memory B cells (same functionality of the parent cell but a slower aging process) and Plasma B lymphocytes. Plasma cells have basically the task of producing and releasing large quantities of antibodies.

Apart from the mutation phenomena briefly mentioned in paragraph 2.1.5, the released antibodies are exact copies of the receptor of the originating Plasma B cell. As a consequence, the antibodies match the antigen with the same strength. The B cells are obviously the main agent in the humoral response. How and when do they start to duplicate? The process begins when a potential antigen (molecule, bacterium or virus) is found by either the B cells or the AP cells. The major difference between these two cells is that the former binds the antigen with its receptor whereas the latter "eats" it without any recognition process. In a broad sense, this is another kind of response to the antigenic invasion. It is quick but not powerful enough to face a massive antigen attack. Actually, the AP cells are not very aggressive with respect to the antigen population, they rather keep catching occasionally some molecules. For this reason they are called the "street sweeper" of the immune system. The best defense is offered by the so-called *acquired* immune system and

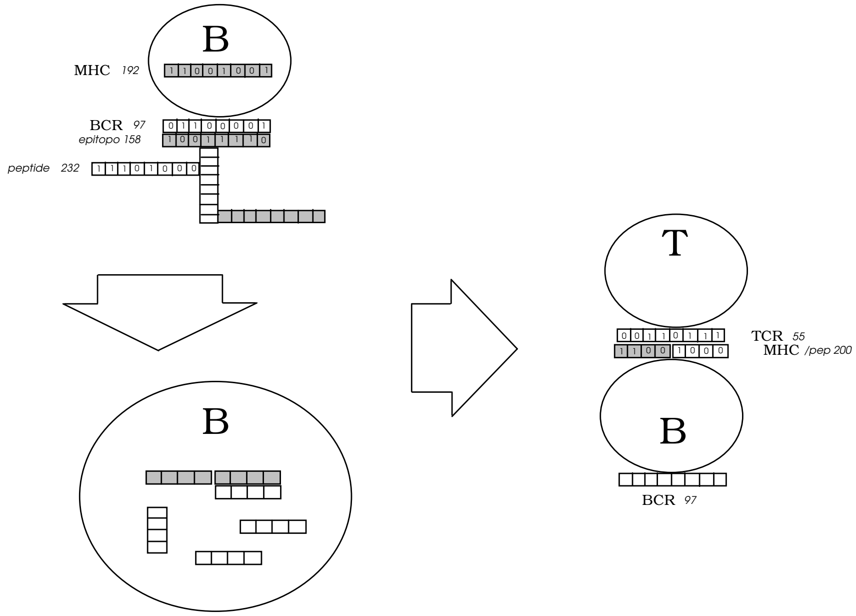


Figure A.1 B-Antigen interaction, endocitosys of B cell and B-T interaction in a system with strings of 8 bits.

it is based on specialized T and B cells.
Both B cells and AP cells process internally the antigens (endocitosys, MHC class II bind antigen-peptides). Upon successful binding, they expose the MHC/peptide groove on their surface. This is a first indication of an antigenic attack but the B's wait for another signal to activate. When a T cell is able to bind the MHC/peptide groove on AP or B cell surface, it gets stimulated and stimulates back the B cells via the emission of certain lymphokines. Figure A.1 shows the various phases of the humoral response.

The lymphokines are soluble molecules. It is known that a large number of different lymphokines exists whose regulatory function is extremely complex.
T helper and B cells stimulate each other. At this time, they start to proliferate. While a proliferating clone of T helper is basically composed by memory cells, some of the clones of the B cells differentiate into Plasma B cells. They produce the antibodies that eliminate the antigen.

A.2 The cellular mediated response

The mechanisms of the cellular response are of special interest in devising and testing vaccines, one of the main applications of immunology.

The simulation of the cellular response requires the modeling of new entities. For instance, the MHC class I (MHCI) was not represented in the previous versions of **CImmSim**. The molecules MHC class I and II play a major role in the process of antigen recognition. They control the binding between the antigen-virus and, respectively, T killer (class I) and T helper (class II) cells. The cells representing the virus targets are modeled as *Generic Epithelial cells* (EP). The virus infects such cells penetrating the cellular membrane. Then it starts to proliferate completely hidden to any antibody produced by the humoral response (see appendix A.1). Virus proliferation inside the infected cell eventually terminates with the *explosion* of the cell. As a result there is the release of a large amount of viruses.

The infection is counteracted by the T killer cells. These lymphocytes are able to recognize MHCI-virus complexes on the infected cells. Upon recognition they kill the infected cells preventing further proliferation of the virus.

Before dying an EP releases a D signal (danger signal) which is here explicitly modeled. The D signal determines the activation of APC from a *resting* state.

During the immune response, both lymphocytes and "accessory" cells (i.e. macrophages and dendritic cells; the latter are not modeled here) produce a number of different molecules named *cytokines*. T helper cells, in this case, release cytokines which help T killer activation. Upon recognition of an MHCI-peptide complex on the surface of an active APC, they produce the interferon-γ (IFN) molecule which determines the probability for a T killer cell to proliferate.

The cytokines have many functions, but most of them act as regulators of the response. In **CImmSim**, for sake of simplicity, just few kinds of cytokines are *explicitly* modeled. We chose to represent the cytokines whose concentration determine the probability for an event to occur. For instance, the aforementioned probability that a T killer duplicates depends on a factor which can be written as $1-\exp(-(\gamma/E)^2)$ where γ is proportional to the number of interferon-γ molecules for a certain constant E. The cytokines which trigger "deterministic" events are not strictly required in our model since, by definition, such events happen regardless of the actual number of molecules.

Appendix
B

CImmSim: optimization techniques and parallel programming

The **CImmSim** code does not just resort to parallel processing to run faster. Data structures and I/O have been optimized as well to limit the (huge) memory and disk space requirements. In this appendix we discuss the solutions adopted to overcome the limitations of the original **IMMSIM** code due to the impractical growth of the APL2-workspace.

B.1 Dynamic memory allocation

In a classic CA there is a simple correspondence between entities of the automaton and sites of the grid. Usually each site contains either one or zero instances of an entity and the number of distinct entities is pretty small. [1] This feature makes the computation of the interactions among entities very simple. Unfortunately it imposes a severe restriction on the number of instances that can be represented. This number can not be greater than the number of lattice points. For a biological model, this is a too strong limitation. As a matter of fact, the IS is a dynamic system with a population that, due to the combination of multiple mechanisms like evolution, learning and memory, may change significantly during the simulation.

One of the most serious issue we had to address has been the huge amount of memory required to store the information which describe the entities and their attributes.

In an "alpha" version of **CImmSim** [25] we adopted, for the sake of simplicity, *static* data structures (arrays) for all the entities of the simulation. This strategy was quickly abandoned because of the exponential growth of the memory requirements for larger simulations. It is easy to show that using arrays the space complexity is

---

[1] In the well known *Game of Life* there is just one possible entity. The Game of Life was introduced in the late 1960's by John Horton Conway [47].

$O(2^{Nl})$ for some constant N, which is obviously unbearable. So we resort to use a more flexible, although more complex, *dynamic* memory allocation technique.

After the choice of the attributes for each cellular entity, the information required for each cell is organized in blocks of variables, *one block for each cell*. In such a way, each cell gets a clear identity (record) in memory and can be easily told apart from the others. The blocks are linked in lists, one for each class of cells: B list, Th list and so on (see table 2.1 for a complete list of the cellular entities). These are *forward* or *single linked* lists. To save memory there is no pointer to the previous element. Only the pointer to the next element is used. The lists are initialized at startup time and are managed dynamically at run time. When a cell dies out it is removed from the list. Likewise, when a virgin cell comes in (from the bone marrow, lymph nodes and/or for clone division of stimulated cells) a block of memory is allocated and the variables-attributes are filled with appropriate values. The new block becomes the head of the corresponding list.
The allocated memory grows linearly with the *number* of cells (there is no direct dependency on the bit string length l) following the clonal growth of cells during the immune response. The growth is limited by an embedded mechanism which emulates the effect of having limited space for cell-proliferation in real bodies.

B.1.1 List permutation

In the real immune system each cell has tens of thousands of receptors on its surface. So, albeit very unlikely, it is possible that it binds more than one entity at the same time. In our simulator this not supported. Each cell has a single receptor, so the binding events follow a *greedy* paradigm: a successful binding event removes the tied entities from the eliciting set to prevent further interactions during the same time step.

It is pretty clear that such mechanism may introduce an artificial bias in the simulation. Indeed, if the scanning order during the interaction phase were simply the cells' lists order, the cells close to the head of the list would get more chances to interact. To alleviate such problem the order of the cells lists changes for each time step. Basically this is a three steps procedure:

1. the pointers to the list elements are copied to an array of pointers;
2. the array is randomly shuffled by swapping pairs of elements;
3. a new list of cells is built with the order defined by the array of pointers.

This technique allows to scan the original list just once. The array is re-ordered in $O(n)$ time. The temporary space required is also $O(n)$ where n is the number of items in the list.

B.1.2 The single antigen case

In a preliminary version of the simulator, the antibodies were stored in static arrays, one for each lattice site. To save space, we introduced a *sparse* data structure and an *hash table* to access it. As a result the amount of memory required to store the antibodies was reduced to $\simeq 9\%$.

Unfortunately, this simple mechanism does not allow to keep track of more than one Ag. A careful analysis shows that changing the hash table structure to support different antigens is too tricky.

A possible alternative is to represent the antibodies by means of a dynamic data structure similar to the lists employed for the cells. The only difference is that, for each lattice site, there is a field in the list to count the number of antibodies that have identical bit strings. With this approach, the memory requirements are reduced to the bare minimum (i.e. in the worst case proportional to $\sum_{k=m_c}^{l} \binom{l}{k}$).

Obviously, the list does not allow immediate access to the single records which is one of the advantages of an hash table. As a consequence, more CPU time is spent updating a record since each access requires a sequential scan of the list. Such drawback is overcome by advantages like code simplicity and generality. As a matter of fact, no change in the data structure will be required when we simulate mutating antigens.

B.2 State flags

All information about the state of a single cell are stored as bit-flags in a packed form. In the previous release [25] just two bits were used. With the introduction of the cellular response the number of possible states was increased. The layout of the byte associated with each cell is currently the following:

byte-flags = b7 b6 b5 b4 b3 b2 b1 b0

- b7 : Memory flag,
- b6 : NOT USED,
- b5 : Ablink flag,
- b4 : NOT USED,
- b3, b2, b1, b0 : State bits (see table B.1).

The use of four state-bits allows to distinguish among 16 states. This is more than actually needed (the number of states is 8) but it leaves space for future developments. Table B.1 gives a short description of each possible state, to be compared with the list in paragraph 2.1.1.

b3 b2 b1 b0	State	Short description	Used for
0 0 0 0	ACT	Ready to interact	APC, B, Th, EP, Tk, PLB
0 0 0 1	INT	Has phagocitated one Ag	APC, B,
0 0 1 0	INF	Ag has infected the cell	EP
0 0 1 1	EXP	Expose Ag-pep on MHCII	APC, B,
0 1 0 0	LOA	Expose Ag-pep on MHCI	APC, EP
0 1 1 0	RES	Inactive (do nothing)	APC
0 1 0 1	STI	Duplicating	B, Th, Tk
0 1 1 1	DEA	Dead (lysis)	EP

Table B.1 State-determining flags.

B.3 Optimized Hamming distance computation

In the CS-model each interaction between molecules or between cell receptors and molecules requires the evaluation of the Hamming distance between two binary strings (see section 2.1.4). The number of such operations per time step and per each lattice site is proportional to the product between the number of interacting cells. This number can be astonishingly high when millions of cells for each entity are simulated. The Hamming distance can be easily evaluated with a simple loop over the bit string length:

```
int dist=0, k, xyxor;
xyxor = x ˆ y;
for(k=0; k<l; k++) dist+=( (xyxor>>k) & 0x1 );
```

Obviously such naive algorithm would have a dramatic impact on the overall performance of the simulator.

A simple but effective alternative is to build a look-up table with 2^l entries which stores the number of bits equal to 1 for each integer between 0 and $2^l - 1$. After that, a single bitwise `XOR` between the two binary strings is required. The result of the `XOR` is used as an index to lookup the table. The value of the corresponding entry is the Hamming distance of the strings.

Even if time is required at startup to build the table, there is a clear advantage in this approach since the evaluation of the distance becomes *almost* independent from the bit string length. The following fragment of C-code fills the look-up table `DHLookUp`:

```
for(n=0; n<(2<<(l-1)); n++){
   for(i=0, ones=0; i<l; i++)
            ones += ( (n >> i) & 0x1 );
```

```
      DHLookUp[n] = ones;
   }
```

The Hamming distance is now given by `DHLookUp[`$x \oplus y$`]`, where $\oplus$ is the bitwise `XOR` operation between the two strings x and y. Note that the size of table grows exponentially with the bit string length l. To avoid the overhead of such large table, for $l > 24$ the evaluation of the Hamming distance is performed in two steps: the strings are divided in two substrings, then the total distance is the sum of the distances of the substrings. In such a way the table has just $2^{l/2}$ entries instead of 2^l.

B.4 Compressed output

In paragraph 2.1.2 we have seen how most of the entities are specified by a integer number in the range $[0, 2^l - 1]$. Information about the number of entities having the same "specificity" must be stored at each time step to allow *off-line* analysis and visualization of the system evolution. Saving such information, for all possible values, requires an array with 2^l entries. Actually, many entries are equal to zero because the expressed *repertoire* is always a (small) subset of the potential *repertoire*. So, there is a danger of wasting a lot of back storage by filling with zeros. Unfortunately, to cope with the general case, there is no way to know in advance *which* entries will be not equal to zero. The number and the position of such entries may vary according to some random events at each time step. As a consequence, we need to store the whole array.

The amount of disk space required just for the B cell can be evaluated as follows: a *long* integer variable is used for each value of the specificity (the number of entities for some specificity may be greater than 65536, so neither a *char* nor a *short* variable can be used). A long integer requires usually four bytes, so we store 4×2^l bytes each time we sample the population state. For a run of 500 time steps, $500 \times 2^{l+2}$ bytes of disk space are required. This means $\simeq$ 8 Gbytes for a simulation with $l = 22$.

To save space we resort to *run time* data compression before the information are stored to the disk in binary format. This means that instead of using the standard C function `fwrite()` directly, there is a call to the `gzipfwrt()` function defined as follows:

```
int gzipfwrt
 (void *ps,size_t size,size_t nitems,FILE *stream){
static char *pc=NULL;
static int pcsize=0;
int compsize,rc;
```

```
 if((size*nitems)>pcsize) {
   if(pc!=NULL) free(pc);
   pcsize=size*nitems*1.01+12+1;
   pc=(char *)malloc((unsigned)pcsize);
   if(pc==NULL) {
     fprintf(stderr,"Can’t get %d bytes\n",pcsize);
     exit(-1);
   }
 }
 compsize=pcsize;
 if((rc=compress(pc, &compsize, ps, size*nitems))) {
   fprintf(stderr,"Error %d in compress\n",rc);
   exit(-1);
 }
 fwrite(&compsize,sizeof(int),1,stream);
 fwrite(pc,(unsigned)compsize,1,stream);
}
```

The function `compress()` is part of *zlib*, a general purpose data compression library [4]. It takes in input a pointer to the original array, the length (in bytes) of the array, a pointer to the buffer which, on return, contains the compressed data and a pointer to a variable which contains the length (in bytes) of the compressed data. When the buffer with the compressed data is ready, its length and the buffer itself are written to disk by means of the `fwrite()` function.

After the simulation ends, the results may be extracted by means of a reverse procedure called `gunzipfrd()`. The length of each data block is read along with the data in compressed format. After that, `uncompress()` is invoked on the specified block of data.
All this machinery will naturally take CPU time but the advantage in saved disk space are much more valuable for long bit string simulations. In any case the compression can be disabled at compiling time.

The compression ratio depends greatly on the number of zeroes to be saved. This number can be roughly estimated given l and m_c. If we do not consider the small fluctuation due to the birth of new cells, the population who reach considerable values are those who may get stimulated to proliferate, that is in other words, those who match the antigen. Given, for example a single antigen with one epitope (i.e. one bit string) the number of possible matches over the threshold m_c are given by the sum $\sum_{k=m_c}^{l}\binom{l}{k}$. The compression ratio can be calculated dividing this number by the total 2^l (assuming the compression reduces to zero the space needed for null values, which is clearly a rough approximation). For a large typical simulation with $l = 20$ and $m_c = 15$ the compression ratio is $\simeq$ 97%. It decreases if we lower the affinity threshold m_c because then we have more interacting cells, that is, less population counters equal to zero. The compression performs better on long bit string simulations. Test cases report a compression rate between about 75% and 82%.

B.4.1 Saving detailed information

Some other information (for example the number of cells for each allowed state) are cumulative. It is possible to store directly their values at run time, even in ASCII format. This facilitates both the debugging and the monitoring of long simulations taking hours, since we can observe the evolution of the different populations just plotting them on the fly.

B.5 The parallel version of the simulator

In **CImmSim** *all* the phases of the simulation run in parallel. The lymphnode is mapped onto a bi-dimensional triangular lattice (six neighbour sites) of $L = L \times L$ sites, with periodic boundary conditions in both directions (up-down, left-right). However, to make the internal management of the lists easier, the lattice sites of the automaton are not arranged as a two-dimensional array but as a linear vector. The transformation is carried out by a simple function $\mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$, $Z = X \times L + Y$ and does not change the global toroidal topology of the body.
The lists which describe the entities are “local” to the processors. In other words, there is no single list split among the processors but as many independent lists as the number of processors in use.
Each Processing Element (PE) works on a subset of lattice sites. In case the Number of PE (*NPE*) is not an exact divisor of L, the reminder R of the sites is spread among the first R PE’s instead of being assigned to a single PE. This technique minimizes the load unbalance among the PE’s. No PE keeps a copy of lists or data structures belonging to other PE’s and in such a way for a fixed bit string length l the memory required on a PE decreases linearly as *NPE* grows.

The problem is not “embarrassingly parallel” because there is a *diffusion* phase in which cells and molecules may migrate from a lattice site towards one of its six nearest neighbors (see figure 2.1).
Elements are deleted from or inserted in the lists of cells and molecules when a cell leaves the “domain” of a PE to migrate to one of the “nearest neighbor PE’s”. Since the sites are distributed as if they were a one-dimensional array, each PE needs to communicate, in a ring topology, with at most with two other PE’s. The element is deleted from the original list and all the attributes are packed in a message sent to the PE which now owns the cell (PE_d). PE_d unpacks the message and inserts the

attributes of the incoming cell in a new element that becomes the head of the list. Actually each PE packs all its outgoing cells in just two messages for performance reasons. The first is directed to $(PE_d - 1) \bmod NPE$, whereas the second goes to $(PE_d + 1) \bmod NPE$.
Fig B.1 shows the various subtasks of *NPE* processes. The horizontal lines represent the communication among PE's.

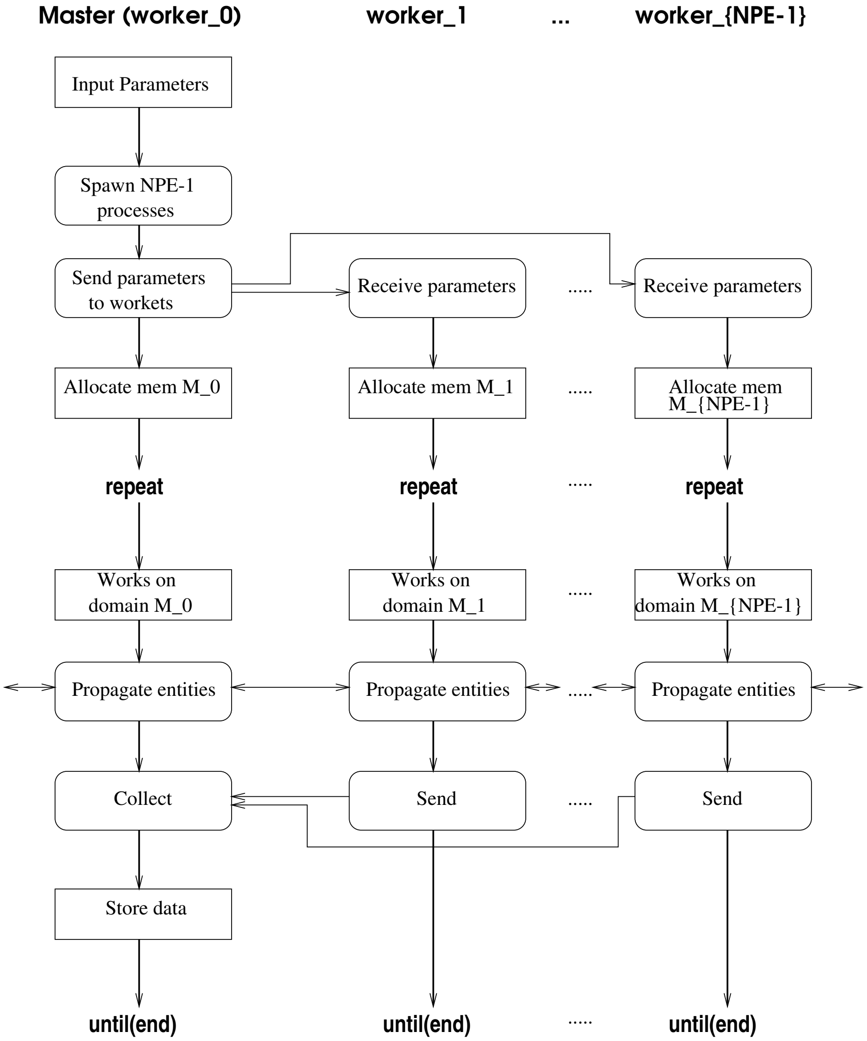


Figure B.1 **CImmSim** communication pattern. *NPE* indicates the number of processing elements.

For **CImmSim** we have resorted to the primitives for parallel programming defined by the PVM software package [2]. The main advantage of PVM is, for our application, the very simple mechanism for packing/unpacking multiple and heterogeneous elements in a single message.

B.6 Computing requirements

For the run presented in section 2.1.6 we have employed 4 nodes of a SUN Ultra Enterprise 10000. These are 333 MHz Ultra SPARC CPU's running SunOS 5.5.1. The run has required almost 11 hours of *elapsed* time; plot in figure B.2 shows the CPU time for each time step on node 0 (the landscape behavior is explained in the caption). The peak of memory allocation has been about 880 MBytes per task whereas the "working set" (that we define as the size shown in the `RSS` column of the `ps` command output) is 260 MBytes for each slave plus 340 MBytes for the master. The permanent storage requirements for the intermediate results have been limited to 225 Mbytes thanks to the on-line compression (see section B.4). At the end of the run, the data have been extracted and filtered to get few Kbytes corresponding to the most significant results.

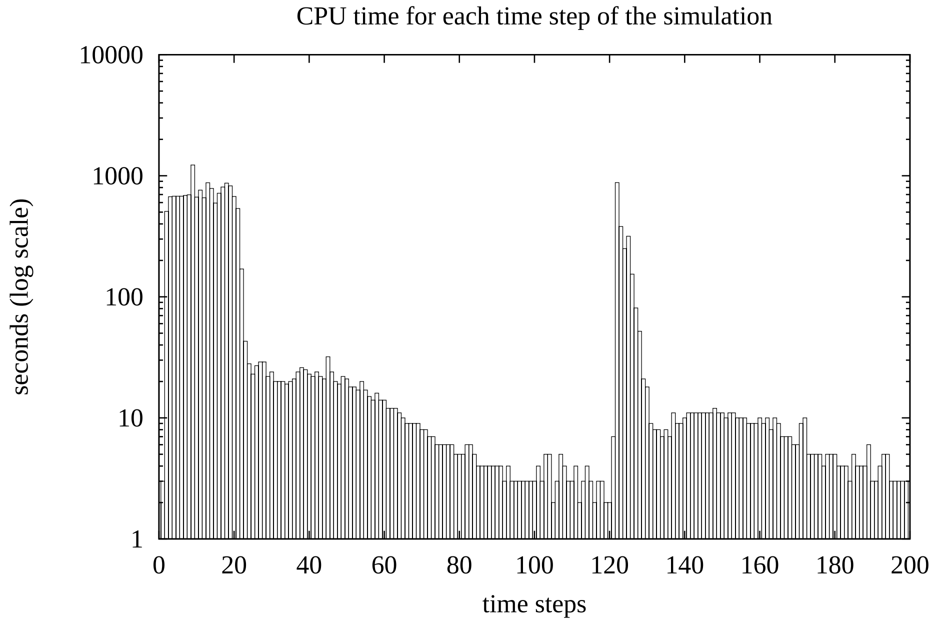


Figure B.2 CPU time for each time step of the 24-bit simulation (see section B.6). The burst of CPU load follows exactly the clonal growth of the cells (see the plots below) because the time required by the interaction procedures depends on the number of cells in each lattice point.

B.7 Timing

Three sets of runs have been performed to measure the efficiency of the parallel code. The quality of these simulations is, from the immunology viewpoint, poor because the set of input parameters is tuned to highlight the impact on the code performance of some "key" parameters. The three set of runs are described in the following paragraphs. All the timings we report are the sum of the user times on all PE's in seconds.

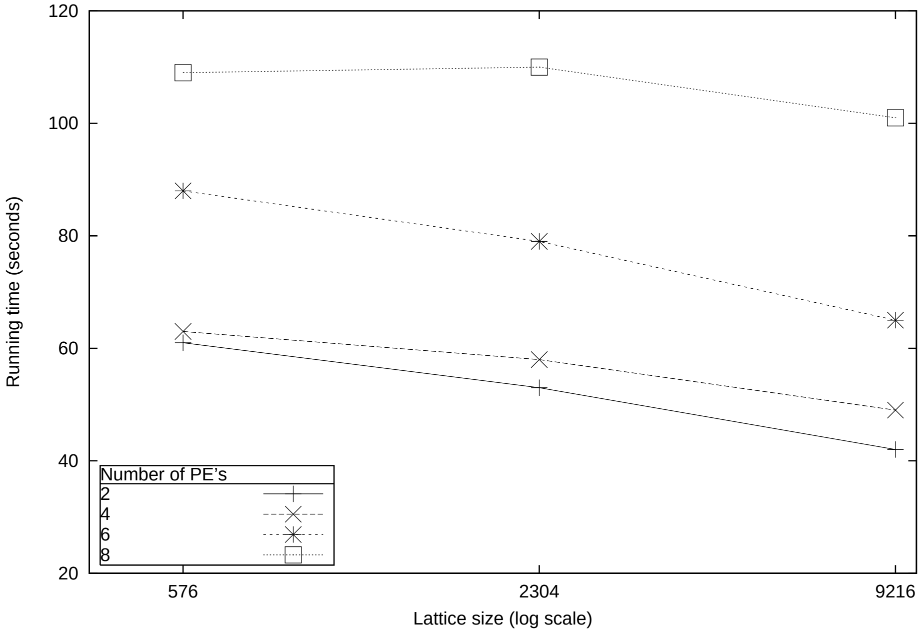


Figure B.3 Scaling of the lattice size L. A set of three runs for 2, 4, 6 and 8 PE's: $L = 24^2$, $L = 48^2$ and $L = 96^2$; 30 time steps; $l = 16$; 15000 initial cells per type (for a total of 75000 cells); one initial injection of 10^4 antigens.

B.7.1 Scaling the lattice size

For this set of simulations we chose the following set of parameters: initial number of cells 15000 per type (i.e. a total of 75000 cells); one injection of 10000 antigens; the lattice size is varied from 24×24 to 48×48 and 96×96.

From figure B.3 it is apparent that the computing time decreases going from the smallest grid to the largest one. The result is less surprising than it looks for a very simple reason: in this model the computational load is determined by the concentration of cells in each lattice site (see paragraph B.7.3) rather than the lattice size as in most of other CA models. The interesting point is that the total time increases by using more processors. The straightforward explanation is that the computational complexity of these test cases does not justify the parallel-machinery. That is, the time the CPU's spend to exchange messages is higher than the time required to compute the interactions among cells.

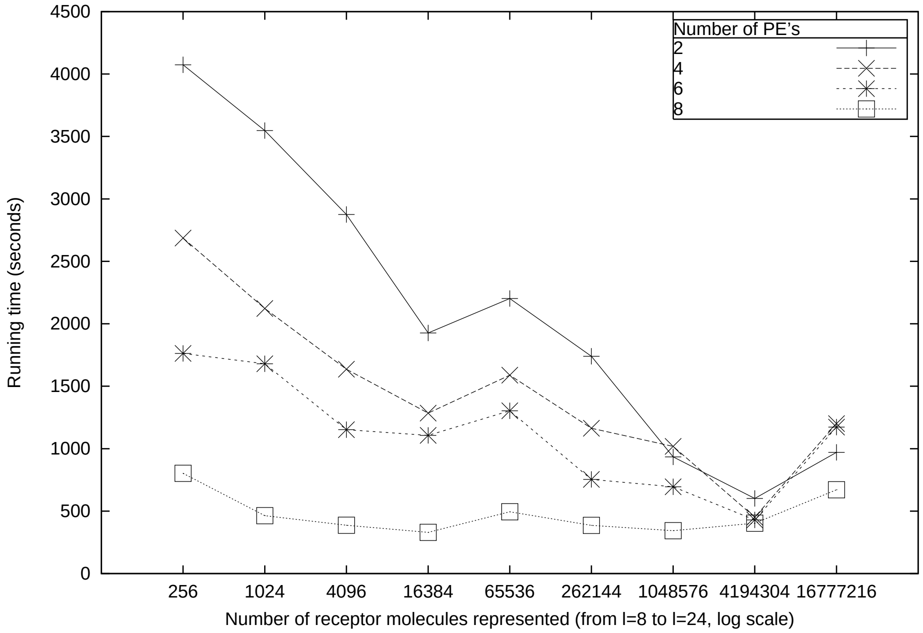


Figure B.4 Scaling of the bit string length. A set of nine runs for 2, 4, 6 and 8 PE's: l from 8 to 24. 100 time steps on a 24^2 grid with 10^4 initial cells per type and no injection of antigens.

B.7.2 Scaling the bit-string length

In the second set of tests we varied the bit string length from 8 to 24 with 5 affinity classes ($l - m_c = 5$). Lattice size is 24^2; 100 time steps; 10000 cells per type but no antigens. In this case the simulation reproduces just the aging process of the cells. The outcomes (figure B.4) show how the code is improved compared to the previous versions [25]. The computing time does not grow anymore when the bit string length increases. Note that the time goes up for $l \geq 22$ since the compression mechanism to save disk space (see section B.4) is not active for $l < 22$.

It is not yet clear why, with few processors, the total time is reduced when *l* increases.

B.7.3 Scaling the initial number of cells

In the last set of runs we have varied the initial number of cells on a 48 × 48 grid. The bit string length *l* is fixed and equal to 16; the number of time steps is 10. The initial number of cells is scaled from 1000 to 512000 for each type (that means up to 2.56 millions of cells) whereas 10^6 antigens are injected for each time step determining a great load which triggers a quick response (cfr figure B.2). Here we obtain a classic scaling law for the elapsed time: growing with the number of cells and decreasing with the number of processors.

This result confirms that the computational complexity of **CImmSim** depends essentially on the number of interacting elements.

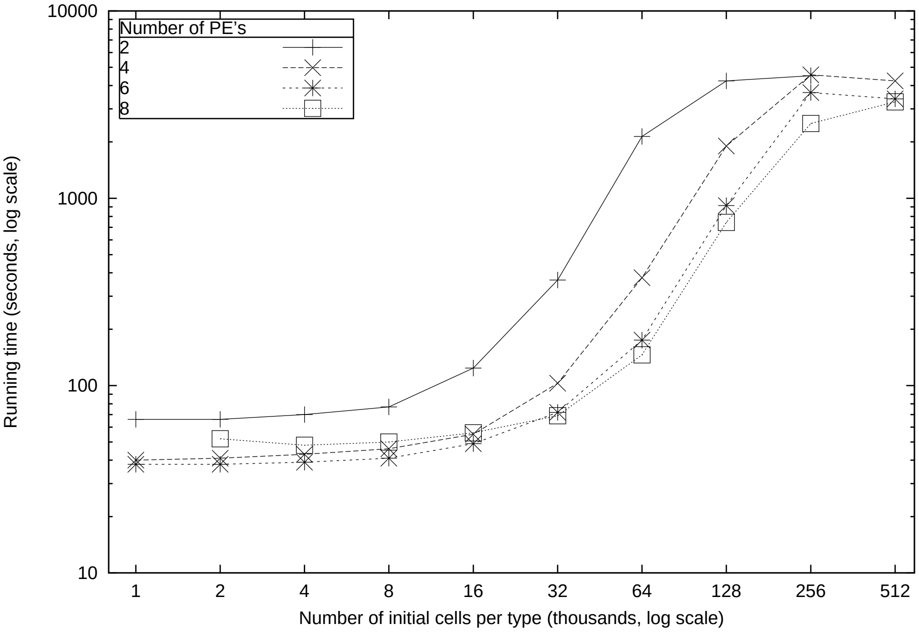


Figure B.5 Scaling of the initial number of cells. It determines the number of total interactions. A set of ten runs for 2, 4, 6 and 8 PE's. One million antigens are injected at each time step for 10 time steps. $L = 48^2$; the number of distinct molecules is 2^{16} (i.e. bit string length $l = 16$).

B.8 Porting to other platforms

The choice of the PVM programming model has made pretty easy the porting of **CImmSim** to other platforms. Indeed, besides the original SP2 version, **CImmSim** runs on a number of other platforms: from NOW (Network Of Workstations) to the Cray/T3E supercomputer. However, the public domain version of PVM has been used just on clusters. For performance reasons, for instance, we have been forced to use our own version of PVM on the Sun Ultra Sparc. This version is similar to PVMe in design but it is specially tuned for shared memory architectures [14].

On the T3E we have resorted to the Cray PVM environment. The only problem of the Cray PVM is that it follows strictly the SPMD philosophy, the `pvm_spawn` primitive is not supported and there is a one to one correspondence between task and processor. These features have required slight changes in the **CImmSim** code.

Appendix

C

AMSE: parallel programming

Since we have more sophisticated agents than in [24], more computing power is required to carry on large scale simulations. To address this issue, the current version of the code (**AMSE** ver 0.6) resorts to parallel processing.

C.1 The parallel version of the simulator

The approach to the parallelization is similar to that described in [26] (see appendix B) for the simulation of the immune system response. However there are some significant differences. First of all we have replaced PVM with MPI [1] mainly to exploit the better support for Collective Communication Primitives (CCP's) that MPI offers. In the present code (**AMSE**), CCP's are used to perform many *reduce* operations in parallel instead of collecting all data on a single node and then process them sequentially.

The scheme which is shown in figure C.1 can be summarized as follows. Each task of a parallel run is in charge of a subset of the total number of agents. All phases of the simulation are executed in parallel and there is no dependency on the total number of tasks. There are two phases in which the tasks interact: the "diffusion" phase and the output phase. During the diffusion, agents may migrate from a task to another and the communication is point-to-point. All receive operations are *posted* in advance, to avoid any dependency on the internal buffering mechanisms of MPI. To evaluate global quantities required by all tasks (e.g. the price change or the total volume) CCP's are employed. The same technique is applied when data are collected from all tasks before writing results in the files.

As shown in figure C.2, the efficiency of the parallel code depends strongly on the number of agents. This is not surprising since the overhead of the implicit synchronization required by the CCP's is, for few agents, greater or equal to the speedup due to the parallel processing.

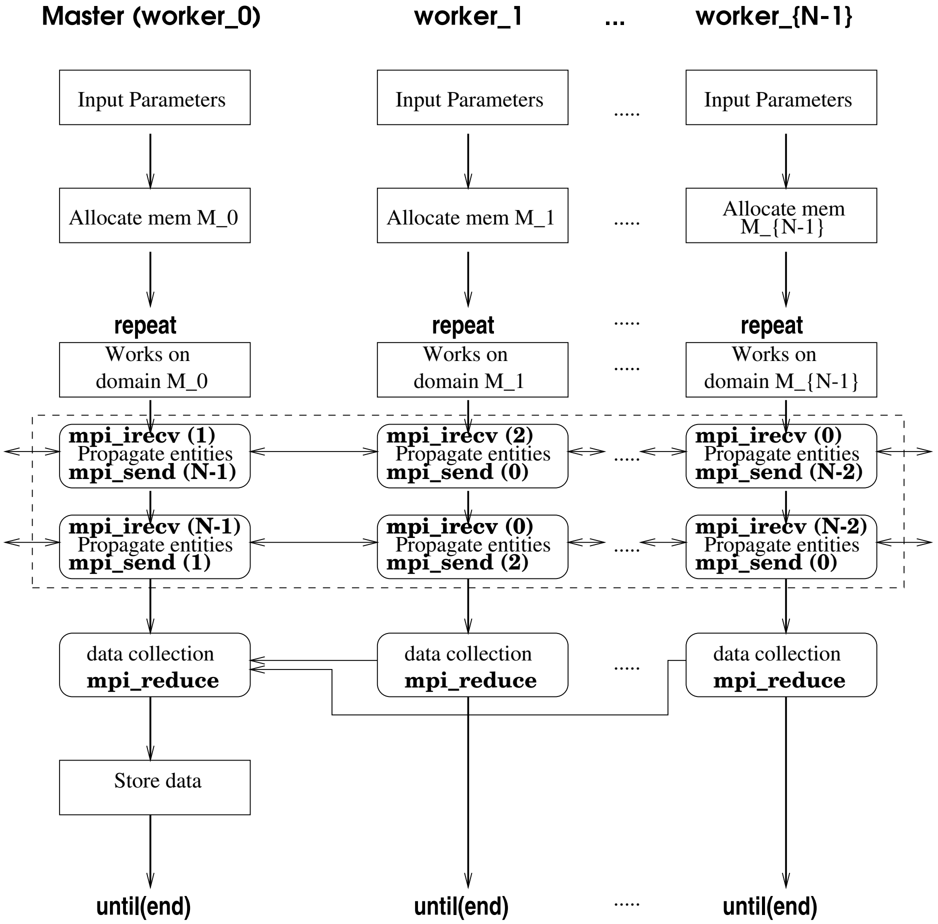

Master (worker_0)
worker_1
...
worker_{N-1}
Input Parameters
Allocate mem M_0
Allocate mem M_1
Allocate mem M_{N-1}
repeat
Works on domain M_0
Works on domain M_1
Works on domain M_{N-1}
mpi_irecv (1) Propagate entities mpi_send (N-1)
mpi_irecv (2) Propagate entities mpi_send (0)
mpi_irecv (0) Propagate entities mpi_send (N-2)
mpi_irecv (N-1) Propagate entities mpi_send (1)
mpi_irecv (0) Propagate entities mpi_send (2)
mpi_irecv (N-2) Propagate entities mpi_send (0)
data collection mpi_reduce
Store data
until(end)

Figure C.1 **AMSE** Communication scheme. The numbers in parentheses represent the source, for the mpi_irecv, or the target, for the mpi_send, of the point-to-point communication operations (e.g., `mpi_irecv(1)` means receive message from worker_1, `mpi_send(N-1)` means send message to worker_{N-1}).

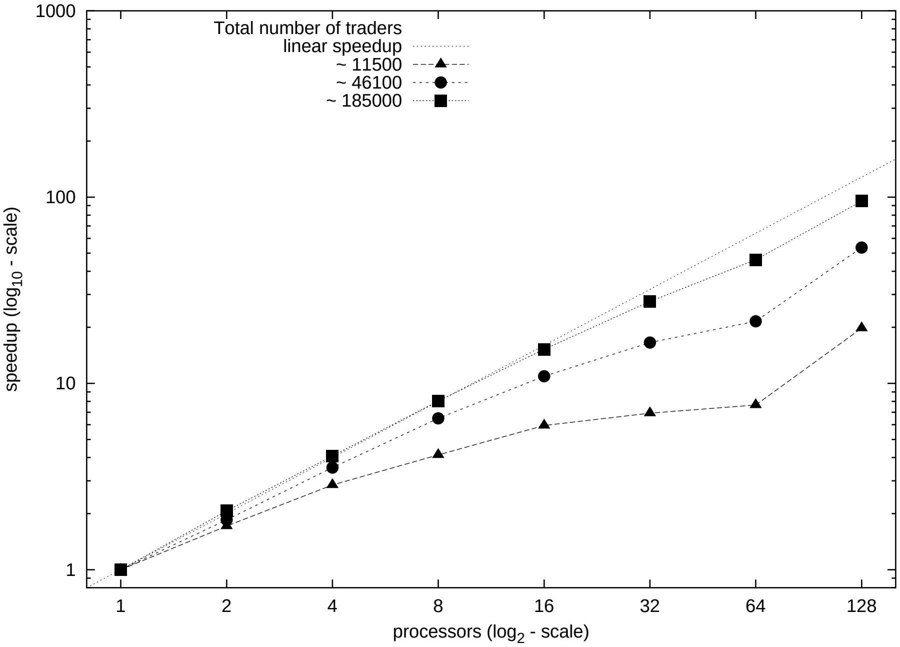


Figure C.2 Speedup for different system size simulations (the speedup is defined as $T(1)/T(N)$, where $T(N)$ is the time with N processors). Timings on a Cray T3E with a number of processors up to 128.

Owing to the MPI portability, exactly the same code runs on pretty different platforms like the Sun Enterprise 10000, the Cray T3E and the IBM SP2.

Appendix D

Glossary of Notation

Symbol	Meaning	See page
MS	microsimulation model	1
L	linear dimension of a lattice	6
s_n	stochastic variable identifying a spin	6
LGA	lattice gas automata	6
ILGA	integer lattice gas automata	7
$r(x,t)$	the number of entities on each lattice site x at time t	7
ULG	unbounded lattice gas automata	8
SFSM	stochastic finite state machine	8
IS	immune system	13
`AND`	logical and	14
`OR`	logical or	14
`NOT`	logical not, negation	14
APL2©	A Programming Language. ©Copyright IBM Corp.	17
CImmSim	C version of **IMMSIM**	18
IMMSIM	Celada-Seiden computational model	18
MHC	major histocompatibility complex	20
B	lymphocyte B	20
Th	lymphocyte T helper	20
Tk	lymphocyte T killer, cytotoxic	20
APC	generic antigen processing cell, macrophage	20
EP	epithelial cell, generic virus-target cell	20
PLB	lymphocyte plasma B	20
IFN	interferon-γ	20
D	danger signal	20

IC	immune-complexes	20
Ag	generic antigen or virus	20
Ab	generic antibody	20
l	bit-string length	21
$\upsilon(m)$	affinity potential	23
m_c	affinity threshold	23
p_b	per-bit mutation probability	26
h_c	hole in the repertoire	27
G_{12}	information gain of the transition from state 1 to state 2	36
$\equiv$	is defined as	37
$f_m,\ f(m,t)$	concentration of m-matching B-cells at time t	37
$\mu(t)$	mean match number	39
$H(t)$	entropy	39
$G(t)$	information gain	39
$\simeq$	roughly equal	40
$N_m(x,t)$	number of B cells in class m at site x at time t	42
$\hat{\upsilon}(t)$	total affinity at time t	45
$\mathbb{N}$	set of natural numbers	53
$\{m_{ij}\}$	mutation probability matrix	54
m_{ij}	probability to jump from class i to j by mutation	54
$\Re(\cdot)$	real part of the variable which appear between parenthesis	57
p_h	error threshold	58
$\ll$	much smaller than	60
$\propto$	proportional to	60
p_t	asset price at time t	69
Δp_t	price change at time t	69
R_t	net return	69
r_t	log-return	69
ε_t	Gaussian increment	69
σ	standard deviation of a probability distribution	69
μ	mean value of a probability distribution	69
NYSE	New York Stock Exchange	70
S&P500	Standard and Poor 500 financial index	70
DJI	Dow Jones Industrial Average financial index	70
i.i.d.	independent identically distributed	70
EMH	efficient market hypothesis	70

PDF	probability distribution function	71
α	exponent of a power law or a Lévy distribution	71
$P(\cdot)$	PDF of the random variable which appears between parenthesis	72
STL	Scale-invariant Truncated Lévy distribution	72
κ	excess kurtosis	72
$\mathcal{N}(\mu, \sigma^2)$	standardized Gaussian distribution with mean μ and standard deviation σ	72
$c(\tau)$	autocorrelation function	74
v_t	volatility at time t	74
a	probability for a trader to be active at each time step	78
p	probability to occupy a site in a percolation lattice	78
p_c	percolation threshold	79
PVM	parallel virtual machine	80
$F_q(\tau)$	structure function	81
$< \cdot >$	time average	81
$r_t^{(\tau)}$	time-lagged return	81
ζ_q	scaling exponent	81
AMSE	A Model of Stock Exchange	87
$x_t^{(i)}$	decision to buy, sell or stay inactive of agent i at time t	88
$c_t^{(i)}$	amount of credits or money of agent i at time t	88
$n_t^{(i)}$	number of owned stock of agent i at time t	88
$w_t^{(i)}$	total wealth (stocks plus money) of agent i at time t	88
D_t	excess demand at time t	88
N_e	number of traders in class e	89
N_F	number of funamentalists	89
N_N	number of noisy	89
N_C	number of chartists	89
$s^{(i)}$	influence strength of agent i	89
$\bar{m}_t(h)$	moving average computed over the time horizon h	89
$p_t^+(\delta)$	filtered price $p_t + \delta p_t$	89
$p_t^-(\delta)$	filtered price $p_t - \delta p_t$	89
$\gg$	much bigger than	91
$pr[\cdot]$	probability of the elementary event between parenthesis	91
V_t	trading volume at time t	94

MRS	modified rescaled range statistics	102
Q_n	MRS statistics	102
NN	neural networks	105
MLP	multi-layer perceptrons	106
EBP	error back propagation learning algorithm	107
η	learning rate of the EBP algorithm	107
β	momentum of the EBP algorithm	107
ϱ	mean square error	110
g_t	forecasted price at time *t*	110
ϱ_V	mean square error on the validation set	110
$\mathcal{S}$	one-step sign prediction rate	111
HS	Heaviside function	112
`XOR`	bitwise logical exclusive `OR`	126
PE	processing element	129
CCP	collective communication primitives	137
MPI	message passing interface	137

Bibliography

[1] The Message Passing Interface standard. *www-unix.mcs.anl.gov/mpi/* .

[2] Parallel Virtual Machine. *www.epm.ornl.gov/pvm/* .

[3] The Quotations Page. *www.quotationspage.com* .

[4] Zlib ver. 1.0.4, a general purpose data compression library.

[5] *Workshop 2000 "Agent-Based Simulations", Passau, Germany, May 2-3, 2000* , 2000. www.or.uni-passau.de/workshop2000/agents.html.

[6] AGUR, Z., AND KERSZBERG, M. The emergence of phenotipic novelties through progressive genetic change. *Am. Nat.* **129**:862–875, 1987.

[7] ANDREW, D. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica* **59**:817–858, 1991.

[8] AUYANG, S. *Foundations of Complex-System Theories in Economics, Evolutionary Biology and Statistical Physics*. Cambridge Univ. Press, Cambridge, UK, 1998.

[9] BAGNOLI, F., AND BEZZI, M. Species formation in simple ecosystems. *Int. J. Mod. Phys. C* **9**:999, 1998.

[10] BALASUBRAMANIAM, V. *e-print archive, cond-mat/9601030* , 1996.

[11] BAVIERA, R., PASQUINI, M., SERVA, M., VERGNI, D., AND VULPIANI, A. Efficiency in foreign exchange markets. *e-print archive, cond-mat/9901225* , 1999.

[12] BECK, C., AND SCHLOGL, F. *Thermodynamics of chaotic systems*. Cambridge Univ. Press, Nonlinear Science Series, Cambridge, UK, 1993.

[13] BERNARDES, A., AND ZORZENON DOS SANTOS, R. Immune network at the edge of chaos. *J. Theo. Biol.* **186**:173–187, 1997.

[14] BERNASCHI, M. Efficient message passing on UNIX shared memory multiprocessor. *Future Generation Computer Systems Journal* **13**:443, 1998.

[15] BERNASCHI, M., AND CASTIGLIONE, F. Effect of technical traders in a synthetic stock market. *Int. J. Mod. Phys. C* **11**:1437, 2000.

[16] BEZZI, M., CELADA, F., RUFFO, S., AND SEIDEN, P. The transition between immune and disease states in a cellular automaton model of clonal immune response. *Physica A* **245**:145–163, 1997.

[17] BOGHOSIAN, B., YEPEZ, J., ALEXANDER, F., AND MARGOLUS, N. Integer lattice gases. *Phys. Rev. E* **55**:4137–4147, 1997.

[18] BOUCHAUD, J., AND POTTERS, M. *Theory of financial risk.* Cambridge Univ. Press, Cambridge, MA, 2000.

[19] BROSA, U., AND STAUFFER, D. Vectorized multisite coding for hydrodynamic cellular automata. *Jour. Stat. Phys.* **57**:399–403, 1989.

[20] BURNET, F. *The Clonal Selection Theory of Acquired Immunity*. Vanderbuil University, Nashville, TN, 1959.

[21] CAMPBELL, J., LO, A., AND MACKINLAY, A. *The Econometrics of Financial Markets.* Princeton Univ. Press, NJ, 1997.

[22] CAMPBELL, J., AND SHILLER, R. The dividend-price ratio and expectations of future dividends and discount factors. *Review of Financial Studies* **1**:195–227, 1988.

[23] CARDARELLI, G., MARSILI, M., AND ZHANG, Y. A prototype model of stock exchange. *Europhys. Lett.* **40**:479, 1997.

[24] CASTIGLIONE, F. Diffusion and aggregation in an agent based model of stock market fluctuations. *Int. J. Mod. Phys. C* **11**:865–880, 2000.

[25] CASTIGLIONE, F., BERNASCHI, M., AND SUCCI, S. Simulating the immune response on a distributed parallel computer. *Int. J. Mod. Phys. C* **8**:527–545, 1997.

[26] CASTIGLIONE, F., BERNASCHI, M., AND SUCCI, S. A high performance simulatior of the immune response. *Future Generation Computer Systems* **15**:333–342, 1999.

[27] CASTIGLIONE, F., MANNELLA, G., MOTTA, S., AND NICOSIA, G. A network of cellular automata for the simulation of the immune system. *Int. J. of Mod. Phys. C* **10**:677–686, 1999.

[28] CASTIGLIONE, F., PANDEY, R., AND STAUFFER, D. Effect of trading momentum and price resistance on stock market dynamics: A Glauber Monte Carlo simulation. *Physica A* **289**:223–228, 2001.

[29] CELADA, F., AND SEIDEN, P. A computer model of cellular interaction in the immune system. *Immunology Today* **13**:56–62, 1992.

[30] CELADA, F., AND SEIDEN, P. Affinity maturation and hypermutation in a simulation of the humoral immune response. *Eur. J. Immunol.* **26**:1350, 1996.

[31] CHEN, S.-H., AND YEH, C.-H. On the emergent properties of artificial stock markets: some initial evidences. *National Chengchi University, Taipeh* , 1999.

[32] CHOWDHURY, D., AND STAUFFER, D. A generalized spin model of financial markets. *Eur. Phys. J. B* **8**:477–482, 1999.

[33] CONT, R., AND BOUCHAUD, J. Herd behaviour and aggregate fluctuations in financial markets. *Macroeconomic Dynamics* **4**, 2000.

[34] CYBENKO, G. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signal and Systems* **2**:303–314, 1989.

[35] DAYAN, I., HAVLIN, S., AND STAUFFER, D. Cellular automata generalisation of the Weisbuch-Atlan model for immune response. *J. Phys. A* **21**:2473–2476, 1988.

[36] DE OLIVEIRA, S., DE OLIVEIRA, P., AND STAUFFER, D. *Evolution, Money, War, and Computers - Non-Traditional Applications of Computational Statistical Physics*. Teubner, Stuttgart-Leipzig, 1999.

[37] EIGEN, W. *Naturwissenschaften* **58**:465, 1971.

[38] ELLIS, R. *Entropy, Large Deviations and Statistical Mechanics*. Springer, Berlin, 1985.

[39] FARMER, J. Market force, ecology, and evolution. *J. Econ. Behavior and Organization* , 1998. e-print archive, adapt-org/9812005.

[40] FARMER, J. Physicists attempt to scale the ivory towers of finance. *e-print archive adapt-org/9912002* , 1999.

[41] FARMER, J., AND LO, A. Frontiers of finance: evolution and efficient markets. *Proc. Natl. Acad. Sci. USA* **96**:9991–9992, 1999.

[42] FARMER, J. D., PACKARD, N., AND PERELSON, A. The immune system, adaptation and machine learning. *Physica D* **22**:187–204, 1986.

[43] FELLER, W. *An introduction to the probability theory and its applications*. John Wiley & Sons, New York, NY, 1971.

[44] FRANCI, F., AND MATASSINI, L. Life in the stock market - a realistic model for trading. *e-print archive, cond-mat/0008466* , 2000.

[45] FRISCH, U. *Turbulence: the legacy of Kolmogorov*. Cambridge Univ. Press, Cambridge, UK, 1995.

[46] FRISCH, U., HASSLACHER, B., AND Y.POMEAU. Lattice-gas automata for the Navier-Stokes equation. *Phys. Rev. Lett.* **56**:1505–1508, 1986.

[47] GARDNER, M. The fantastic combinations of John Conway's new solitaire game of "Life". *Scientific American* **223**:120–123, 1970.

[48] GHASHGHAIE, S., BREYMANN, W., PEINKE, J., AND TALKNER, P. Turbulent cascades in foreign exchange markets. *Nature* **381**:767–770, 1996.

[49] GNEDENKO, B., AND KOLMOGOROV, A. *Limit Distribution for Sums of Independent Random Variables*. Addison Wesley, 1954.

[50] GOPIKRISHNAN, P., PLEROU, V., AMARAL, L. N., MEYER, M., AND STANLEY, H. Scaling of the distribution of fluctuations of financial market indices. *Phys. Rev. E* **60**:5305–5316, 1999.

[51] HAMMING, R. W. Error detecting and error correcting codes. *Bell Syst. Tech. J.* **29**:147–160, 1950.

[52] HURST, H. Long term storage capacity of reservoirs. *Transaction of the American Society of Civil Engineers* **116**:770–799, 1951.

[53] IORI, G. A microsimulation of traders activity in the stock market: the role of heterogeneity, agents' interactions and trade frictions. *e-print archive, adap-org/9905005* , 1999.

[54] IORI, G. Scaling and multi-scaling in financial markets. *e-print archive, cond-mat/0007385* , 2000.

[55] IVANOVA, K., AND AUSLOOS, M. Low q-moment multifractal analysis of gold price, Dow Jones Industrial average and BGL-USD exchange rate. *Eur. Phys. J. B* **8**:665–669, 1999.

[56] KAUFFMAN, S. *The origins of order*. Oxford Univ. Press, New York, NY, 1993.

[57] KAUFMAN, M., URBAIN, J., AND THOMAS, R. Towards a logical analysis of the immune response. *J. Theor. Biol.* **114**:527, 1985.

[58] KEPLER, T., AND PERELSON, A. Somatic hypermutation in B cells: an optimal control treatment. *J. theor. Biol.* **164**:37, 1993.

[59] KIM, G., AND MARKOWITZ, H. Investment rules, margin, and market volatility. *The Journal of Portfolio Management* **16**:45–52, 1989.

[60] KINCHIN, A. *Mathematical foundations of Information Theory*. Dover Publications, Inc., New York, 1957.

[61] KOHRING, G. Parallelization of short- and long-range cellular automata on scalar, vector, SIMD and MIMD machines. *Int. J. Mod Phys.* **2**:755–772, 1991.

[62] LEBARON, B. Agent based computation finance: suggested readings and early research. *Journal of Economic Dynamics and Control* . to appear.

[63] LEBARON, B., ARTHUR, W., AND PALMER, R. Time series properties of an artificial stock market. *Journal of Economic Dynamics and Control* **23**:1487–1516, 1999.

[64] LEFÈVRE, O. IMMSIM 2.1© a user's guide, 1995. ©Copyright IBM Corp. 1992-94, All right reserved.

[65] LEVIN, S., GRENFELL, B., HASTINGS, A., AND PERELSON, A. Mathematical and computational challenges in population biology and ecosystem science. *Science* **275**:334–343, 1997.

[66] LEVY, H., LEVY, M., AND SOLOMON, S. *Microscopic Simulations of Financial Markets*. Academic Press, New York, NY, 2000.

[67] LEVY, M., LEVY, H., AND SOLOMON, S. A microscopic model of the stock market: cycles, booms, and crashes. *Econ. Lett.* **45**:103, 1994.

[68] LEVY, M., LEVY, H., AND SOLOMON, S. Microscopic simulation of the stock market: the effect of microscopic diversity. *J. Physique I* **5**:1087, 1995.

[69] LEVY, M., LEVY, H., AND SOLOMON, S. New evidence for the power law distribution of wealth. *Physica A* **242**:90, 1997.

[70] LÉVY, P. *Théorie de l'addition des variables aléatoires*. Gauthier Villars, Paris, 1937.

[71] LIEBREICH, J. Influence of finite capital in the Cont-Bouchaud-model for market fluctuations. *Int. J. Mod. Phys. C* **10**:1317–1325, 1999.

[72] LIPPERT, K., AND BEHN, U. Modeling and immune system: architecture and dynamics of idiotypic networks. In *Ann. Rev. Comp. Phys. vol. V* (1997), D. Stauffer, Ed., World Scientific, p. 287.

[73] LIU, Y., GOPIKRISHNAN, P., CIZEAU, P., MEYER, M., PENG, C.-K., AND STANLEY, H. Statistical properties of the volatility of price fluctuations. *Phys. Rev. E* **60**:1390–1399, 1999.

[74] LO, A. Long-term memory in stock market prices. *Econometrica* **59**:1279–1313, 1991.

[75] LUX, T. Multi-fractal processes as model for financial returns: a first assessment. *Uni-Bonn preprint* , 1999.

[76] LUX, T., AND AUSLOOS, M. Market fluctuations I: scaling, multi-scaling and their possible origins. 2000. (review) preprint.

[77] LUX, T., AND MARCHESI, M. Scaling and criticality in a stochastic multi-agent model of financial market. *Nature* **397**:499–500, 1999.

[78] MANDELBROT, B. *J. Business, Univ. Chicago* **36**:394, 1963.

[79] MANDELBROT, B. *The Fractal geometry of Nature*. W.H. Freeman and Co., New York, NY, 1977.

[80] MANDRIOLI, D., AND GHEZZI, C. *Theoretical Foundations of Computer Science*. Krieger Publishing Company, Melbourne, Florida, 1993.

[81] MANTEGNA, R., AND STANLEY, H. Scaling behaviour in the dynamics of an economic index. *Nature* **376**:46, 1995.

[82] MANTEGNA, R., AND STANLEY, H. *An introduction to Econophysics: correlation and Complexity in Finance*. Cambridge University Press, Cambridge, MA, 1999.

[83] MOLGEDEY, L., AND EBELING, W. Local order, entropy and predictability of financial time series. *Eur. Phys. J. B* **15**:733–737, 2000.

[84] MOODY, J. *Forecasting the Economy with Neural Nets: A survey of Challenges and Solutions*. In Orr and Müller [88], 1998.

[85] MURRAY, J. *Mathematical Biology*. Springer, Berlin, 1989.

[86] NEUNEIER, R., AND ZIMMERMANN, H. *How to Train Neural Networks*. In Orr and Müller [88], 1998.

[87] NOWAK, A., SZAMREJ, J., AND LATANÈ, B. From private attitude to public opinion: a dynamic theory of social impact. *Psychological Review* **97**:362–376, 1990.

[88] ORR, G., AND MÜLLER, K.-R., Eds. *Lect. N. Comp. Sci. 1524, "Neural Networks: tricks of the trade"*. Springer, Heidelberg, 1998.

[89] PANDEY, R., AND STAUFFER, D. Immune response via interacting three dimensional network of cellular automata. *J. de Physique* **50**:1, 1989.

[90] PANDEY, R., AND STAUFFER, D. Metastability with probabilistic cellular automata in an HIV infection. *J. Stat. Phys.* **61**:235, 1990.

[91] PARETO, V. *Cours d'Economique Politique* **2**, 1897.

[92] PASQUINI, M., AND SERVA, M. Clustering of volatility as a multiscale phenomenon. *e-print archive, cond-mat/9903334* , 1999.

[93] PERELSON, A., Ed. *Theoretical Immunology, Part Two*. Addison Wesley, Redwood City, CA, 1988.

[94] PERELSON, A., AND WEISBUCH, G., Eds. *Theoretical Immunology, Part One*. Springer, Berlin, 1988.

[95] PERELSON, A., AND WEISBUCH, G. Immunology for physicists. *Rev. Mod. Phys.* **69**:1219–1267, 1997.

[96] PHILLIPS, P. Time series regression with δ unit root. *Econometrica* **55**:277–301, 1987.

[97] PODOBNIK, B., IVANOV, P., LEE, Y., AND STANLEY, H. Scale invariant truncated Lévy process. *e-print archive, cond-mat/9906381* , 1999.

[98] PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, Cambridge, UK, 1994.

[99] PRIGOGINE, I. *From being to becoming*. Freeman and Co., New York, NY, 1980.

[100] QUINN, M. *Parallel Computing : Theory and Practice*. McGraw-Hill, New York, NY, 1993.

[101] REFENES, A., BURGESS, A., AND BENTZ, Y. Neural networks in financial engineering: a study in methodology. *IEEE Transactions on Neural Networks* **8**, 1997.

[102] ROTYIS, J., AND VATTAY, G. Statistical analysis of the stock index of the Budapest stock exchange. *e-print archive, cond-mat/9711008* , 1997.

[103] RUMELHART, D., HINTON, G., AND WILLIAMS, R. Learning internal representation by error propagation. In *Parallel Distributed Processing: Exploration in the Microsctructure of Cognition. Volume I: Foundations* (Cambridge, MA, 1986), D. Rumelhart and J.L.McClelland, Eds., MIT Press/Bradford Books, pp. 318–362.

[104] RUMELHART, D., HINTON, G., AND WILLIAMS, R. Learning representation by back-propagation-error. *Nature* **323**:533–536, 1986.

[105] SEIDEN, P. *personal communication* .

[106] SEIDEN, P., AND CELADA, F. A model for simulating cognate recognition and response in the immune system. *J. Theor. Biol.* **158**:329–357, 1992.

[107] SHANNON, C. The mathematical theory of communication. *Bell Syst. Techn. J.* **27**:379–423;623–656, 1948.

[108] SORNETTE, D., STAUFFER, D., AND TAKAYASU, H. Market fluctuations II: multiplicative and percolation models, size effects and predictions. *e-print archive, cond-mat/9909439* , 1999.

[109] STAUFFER, D. Can percolation theory be applied to the stock market? *Annalen der Physik (Leipzig)* **7**:529, 1998.

[110] STAUFFER, D., AND AHARONY, A. *Introduction to Percolation Theory, 2nd ed.* Taylor & Francis, 1992.

[111] STAUFFER, D., DE OLIVEIRA, P., AND BERNARDES, A. Monte Carlo simulation of volatility clustering in market model with trading. *Int. J. Theor. and Appl. Finance* **2**:83–94, 1999.

[112] STAUFFER, D., AND JAN, N. Sharp peaks in the percolation model for stock markets. *Physica A* **277**:215, 2000.

[113] STAUFFER, D., AND PANDEY, R. *Computers in Physics* **6**:404, 1992.

[114] STAUFFER, D., AND PENNA, T. Crossover in the Cont-Bouchaud percolation model for market fluctuations. *Physica A* **256**:284–290, 1998.

[115] STAUFFER, D., AND SORNETTE, D. Self-organized percolation model for stock market fluctuations. *Physica A* **271**:496, 1999.

[116] STEIGLITZ, K., HONIG, M., AND COHEN, L. Market-based control: a paradigm for distributed resource allocation. In *Market-Based Control: A Paradigm for distributed resource allocation* (Hong Kong, 1996), S. Clearwater, Ed., World Scientific.

[117] STIGLER, G. Public regulation of the securities market. *J. Business* **37**:117–142, 1964.

[118] SUCCI, S., APPERT, K., MUSCHIETTI, L., VACLAVIK, J., AND WERSINGER, J. *Phys. Lett. A* **106**:137, 1984.

[119] SUCCI, S., CASTIGLIONE, F., AND BERNASCHI, M. Collective dynamics in the immune system response. *Phys. Rev. Lett.* **79**:4493–4496, 1997.

[120] VANDEWALLE, N., AND AUSLOOS, M. Multi-affine analysis of typical currency exchange rates. *Eur. Phys. J. B* **4**:257–261, 1998.

[121] von Newmann, J., and Burks, A. *Theory of Self-Reproducing Automata*. Urbana: U. Ill. Press, 1966.

[122] Weigend, A., and Gershenfeld, N. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, Reading, MA, 1994.

[123] Weigert, M., Cesari, I., Yonkovitch, S., and Cohn, M. Variability in the light chain sequences of mouse antibody. *Nature* **228**:1045–1047, 1970.

[124] Weinand, R. Somatic mutation, affinity maturation and the antibody repertoire: a computer model. *J. Theor. Biol.* **133**:409–428, 1990.

[125] Weisbuch, G., and Atlan, H. *J. Phys. A* **21**:189, 1988.

[126] White, H. *Artificial Neural Networks approximation and Learning Theory*. Blackwell Publishers, Cambridge, MA, 1992.

[127] Wolfram, S. *Cellular Automata and Complexity*. Addison Wesley, New York, NY, 1994.

[128] Wolfram, S. *The Mathematica Book*. Cambridge Univ. Press, Cambridge, UK, 1996.

[129] Zhang, Y. Toward a theory of marginally efficient markets. *Physica A* **269**:30–44, 1999.

[130] Zorzenon Dos Santos, R. Immune responses: getting close to experimental results with cellular automata models. In *Ann. Rev. Comp. Phys. vol. VI* (1999), D. Stauffer, Ed., World Scientific.

Index

A

affinity potential . . . 23
algorithm
    error back propagation . . . 107
    Marquardt-Levenberg . . . 107
antibody . . . 20
antigen . . . 20
APL2 . . . 17
autocorrelation function . . . 74

B

Bachelier, Louis . . . 69, 82
bit-string
    length . . . 21
    receptor . . . 21
Brownian motion . . . 82

C

cellular automata . . . 6
collective comm. primitives . . . 137

D

danger signal . . . 20
distribution
    binomial . . . 26
    Gaussian . . . 69
    Lévy stable . . . 71, 72
    log-normal . . . 99
    Maxwellian . . . 35
    Pareto . . . 96
    Poisson . . . 26
    scale-inv. truncated Lévy . . . 72
    truncated Lévy . . . 72
Dow Jones Industrial Average . . . 70

E

Econometrics . . . 105
Econophysics . . . 69
Efficient Market Hypothesis
    semi-strong . . . 71
    strong . . . 71
    weak . . . 71
efficient market hypothesis . . . 70
epithelial cell . . . 20
epitopes . . . 20
error threshold . . . 53, 59
excess kurtosis . . . 72

F

function
    Heaviside . . . 112
    logistic . . . 107
    sigmoidal . . . 107

G

grand challenge . . . 2

H

herding behaviour . . . 75, 78
Hurst correlation coefficient . . . 103
Hurst, H.E. . . . 103
hyper-mutation . . . 26

I

immune system . . . 13
immune-complexes . . . 20
information gain . . . 36
interferon-γ . . . 20

K
Kullback information 36
Kullback relative entropy 43
kurtosis 72

L
lattice 5
  integer lattice gas 7
  lattice gas 6
  unbounded lattice gas 8
learning cascade 39, 43
lymphocyte
  B 20
  plasma B 20
  T helper 20
  T killer 20
lymphokines 19

M
macrophage 20
Major Histocompat. Complex . . . 20
Mandelbrot, Benoit 71
message passing interface 137
Message Passing Library 18, 80
model
  IMMSIM 18
  AMSE 87
  CImmSim 18
  agent-based 87
  bit-string 19
  Celada-Seiden 18
  cellular automata 6
  Cont-Bouchaud 78
  Farmer-Packard-Perelson 16
  Frisch-Hasslacher-Pomeau 6
  Ising 6
  Kaufman-Urbain-Thomas . . . 14
  lattice gas 6
  Levy-Levy-Solomon 77
  microsimulation 1, 5
  Pandey-Stauffer 15
  Potts 6
  spin 5
  Weisbuch-Atlan 14
  XY 6
mono-affinity 80
mono-fractality 80
mono-scaling 73
Moore’s law 2
multi-affinity 80
multi-fractality 80
multi-layer perceptrons 106
multi-scaling 73
mutation 26

N
net return 69
neural networks 105
New York Stock Exchange 70
nonlinear least squares fit 107

O
overfitting 110

P
parallel virtual machine . 18, 80, 130
peptides 20
percolation 79
processing element 129

R
random walk 69
relative price change 69
return 69

S
scale invariance 73
scaling exponent 81
self-affinity 80
Shannon, C.E. 35
spin 5
Standard and Poor 500 70
statistics

    modified rescaled range 101
    R/S 101
stochas. finite state machine . . . 8, 20
structure function 81

T
theory
    clonal selection 17
    idiotypic networks 17
    information 35
thymus 13

V
volatility 74, 97, 98
volume 93

Deutsche Zusammenfassung

Die Enwicklung der Computertechnologie hat die Art und Weise zu Forschen stark beinflusst und verändert. Betrachtet man zum Beispiel physikalische Systeme, so werden diese üblicherweise mit Hilfe mathematischer Modelle untersucht. Diese Modell sind häufig analytisch sehr schwer oder gar nicht lösbar, so dass approximative, d.h. numerische oder simulative, Verfahren zum Auffinden einer Lösung notwendig sind.

Bei der Untersuchung numerischer Modelle ist der Computer auf Grund der rasanten und stetigen Steigerung der Rechnerleistung ein effizientes Hilfsmittel, um die Dynamik komplexer Systeme zu verstehen. Darüber hinaus hat die Arbeitsweise digitaler Rechner sogar zu einer vollkommen neuen Klasse von Modellen (Algorithmen) geführt, in der die Wechselwirkungen zwischen den Freiheitsgraden eines Systems durch Regeln zwischen diskreten Zustandsvariablen ausgedrückt werden. Diese Vorgehen ist oft sehr viel näher an der intuitiven Vorstellung über das Systemverhalten als physikalisch–mathematische Modellgleichungen im kontinuierlichen Zahlenraum.

In dieser Arbeit wird ein einheitlicher Rahmen vorgestellt, in dem sich komplexe Systeme mit einer großen Zahl von Freiheitsgraden abbilden lassen. In Anlehnung an die Beschreibung von Spinsystemen lässt sich ein Modell definieren, mit dem sich unterschiedliche Simulationssysteme beschreiben lassen. Dieses neue Modell, “Unbounded–Lattice–Gas” genannt, ist ein Gittergasmodell, bei dem sich auf jedem Gitterplatz eine unbegrenzte Anzahl von “Teilchen” (Zellen, Agenten, etc) aufhalten kann. Die Teilchen gehören zu unterschiedlichen Klassen und können – in Abhängigkeit von ihrer Zugehörigkeit zu einer Klasse – verschiedene Mikrozustände annehmen. Anstatt wie bei Zellularautomaten mit Teilchen auf benachbarten Gitterplätzen zu interagieren, findet die Wechselwirkung *lokal* zwischen Teilchen an demselben Gitterplatz statt. Zusätzlich können sich die Teilchen frei über das Gitter bewegen, ähnlich zum klassischen *Brown'schen* Diffusionprozess.

Jedes Teilchen (*Entität*) des Systems besitzt eine komplexe innere Dynamik und lässt sich als *Stochastic Finite State Machine* repräsentieren. Zustandswechsel des

Teilchens werden durch stochastische Ereignisse ausgelöst, die durch die Wechselwirkung mit anderen Entitäten (wie im Falle des immunologischen Modells) oder äusseren Feldern (wie im Falle des Börsenmodells) entstehen.

Der erste Teil der vorliegenden Arbeit, bestehend aus Kapitel 1 – 3, beschäftigt sich mit der numerischen Immunologie ("computational immunology"), speziell mit der Mikrosimulation der Reaktion des Immunsystems auf Antigene.

Ein sehr differenziertes Mikrosimulationsmodell ("Celada–Seiden Modell") wird detailliert vorgestellt und die Antwort des Immunsystems auf Antigene mit Methoden der statistischen Mechanik untersucht. Der Mechanismus, der zur kollektiven Erkennung von Anitgenen durch das System führt (d.h. der Erkennung der Information, die die Antigene tragen), ist als Lernprozess beschreibar, der sich kaskadenartig im Zustandsraum des Gesamtsystems ausbreitet ("learning cascade"). In Kapitel 3 wird ein anderer Zugang zum Verständnis dieses komplexen Vorgangs gewählt. Ein System gekoppelter Differentialgleichungen wird in Form von "coupled-maps" behandelt. Es wird gezeigt, dass sich hiermit die Lernkaskaden, die zuerst im Mikrosimulationsmodell beobachtet wurden, reproduzieren und erklären lassen.

Der zweite Teil der Arbeit, Kapitel 4 – 7, beschäftigt sich mit dem noch jungen Forschungsgebiet der Wirtschaftsphysik ("Econophysics"). Zunächst werden in Kapitel 4 einige Grundbegriffe von Finanzmärkten eingeführt und empirische Beobachtungen beschrieben. Das anschliessende Kapitel 5 stellt kurz bereits existierende Modelle vor, wobei der Schwerpunkt auf der Beschreibung des "Cont-Bouchaud Perkolations Modells" liegt (Abschnitt 5.1). Schließlich wir in Kapitel 6 das im Rahmen dieser Arbeit entwickelte neue Mikorsimulationsmodell zur Abbildung von Aktivitäten an Finanzmärkten diskutiert.

Das letzte Kapitel der Arbeit (Kapitel 7) weicht von der ansonsten gewählten Vorgehensweise der Arbeit ab und beschäftigt sich mit dem Problem der Vorhersage von Zeitreihen auf Finanzmärkten. Auch hier wird jedoch ein Modell zur Anwendung auf wirtschaftsphysikalische Probleme verwendet, das durch biologische Systeme inspiriert wurde.

Durch die parallelisierte Implementierung beider Modelle konnte die Rechenzeit soweit reduziert werden, dass es möglich ist, Systeme realistischer Größe zu simulieren. Beide Implementierungen nutzen "Message Passing" und können so auf Rechnerarchitekturen mit gemeinsamen und verteiltem Speicher betrieben werden. Konkret verwendet **CImmSim** die Bibliothek "Parallel Virtual Machine" (PVM) und **AMSE** die "Message Passing Interface" (MPI). Auf Grund der unterliegenden Modellarchitektur des Unbounded Lattice Gas ergibt sich ein enormer Geschwindigkeitzuwachs (“Speedup”) für beide Implementierungen bereits mit einem “relativ einfachen” Schema der Parallelisierung.

Acknowledgments

My best acknowledgments go to

Prof. Dr. Dietrich Stauffer and Prof. Dr. Rainer Schrader for supervising my work at the ZPR/ZAIK and for many interesting discussions, sharp criticisms but also many encouragements.

Sauro Succi for making this happen, Massimo Bernaschi for his constant “virtual presence”, encouragements and last (but not least!) for teaching me innumerable “computer tricks”.

The Graduiertenkollegs “Scientific Computing” for support.

The Regional Computing Center of the University of Cologne (RRZK) and the Italian Center for Parallel Computing (CINECA) are kindly acknowledged for providing most of the computational resources.

R.B. Pandey (thanks for supporting my visit to the University of South Mississippi), P.E. Seiden, F. Celada, S. Motta, M. Bezzi (for letting me browse his code on the cellular response) and Z.-F. Huang for stimulating discussions.

All the colleagues at the ZPR/ZAIK for providing a very nice atmosphere to work with. In particular Nils Eissfeld, not only for helping me to translate the summary but also for having shared these years of daily life.

C. Teuner for her kindness and prompt aid. She helped me to solve many practical problems especially during the first period in Cologne.

All my friends in Cologne. Life would have been much more “complex” without them. I will miss them all.

Finally I am grateful to my family for all the rest!

Erklärung

Ich versichere, daß ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; daß diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; daß sie – abgesehen von unten angegebenen Teilpublikationen – noch nicht veröffentlicht worden ist sowie, daß ich eine solche Veröffentlichung vor Abschluß des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. R. Schrader und Prof. Dr. D. Stauffer betreut worden.

Filippo Castiglione

Teilpublikationen

M. Bernaschi and F. Castiglione, Design and implementation of an Immune System Simulator, *Computers in Biology and Medicine*, to appear, 2001

S. Succi, F. Castiglione, and M. Bernaschi, Collective dynamics in the immune system response, *Phys. Rev. Lett.* **79**:4493–4496, 1997

M. Bernaschi, F. Castiglione, and S. Succi, Large-scale Cellular Automata simulations of the Immune System response, *Phys. Rev. E* **61**:1851–1854, 2000

F. Castiglione, Antigen recognition and evolution in a bit-string population model, *Int. J. Mod. Phys. C* **10(6)**:989–1002, 1999

F. Castiglione and D. Stauffer, Multi-scaling in the Cont-Bouchaud microscopic stock market model, submitted to *Quantitative Finance*, Jan. 2001

F. Castiglione, Diffusion and Aggregation in an Agent Based Model of Stock Market Fluctuations, *Int. J. Mod. Phys. C* **11(5)**:865–880, 2000

M. Bernaschi and F. Castiglione, Effect of technical traders in a synthetic Stock Market, *Int. J. Mod. Phys. C* **11(7)**:1437, 2000

F. Castiglione, Forecasting price increments using an artificial Neural Network, *Advances in Complex Systems* **3(1)**, Hermes-Oxford, 2001

Short summary

The main question addressed in this dissertation is how to derive “the collective” (i.e. macroscopic) properties of a system starting from the knowledge of the laws ruling the individual (i.e. microscopic) behaviour. We have presented a unifying approach to model complex systems with large number of degree of freedom. Starting from the definitions of spin systems, with little changes we have defined a model that is well suited to describe two different simulation systems. The first simulates the humoral and cellular immune response and is based on the model of F. Celada and P.E. Seiden. The second simulates the trading activity of agents in a stock market. It is built borrowing some ingredients from other known models. In addition, a group a traders called moving average followers represents chartists who base their decision upon the evolution of the price of the asset. Also, traders group together to take collective decision to model the herding behaviour.

Kurzzusammenfassung

Bei der Untersuchung numerischer Modelle ist der Computer auf Grund der rasanten und stetigen Steigerung der Rechnerleistung ein effizientes Hilfsmittel, um die Dynamik komplexer Systeme zu verstehen. Darüber hinaus hat die Arbeitsweise digitaler Rechner sogar zu einer vollkommen neuen Klasse von Modellen (Algorithmen) geführt, in der die Wechselwirkungen zwischen den Freiheitsgraden eines Systems durch Regeln zwischen diskreten Zustandsvariablen ausgedrückt werden. Diese Vorgehen ist oft sehr viel näher an der intuitiven Vorstellung über das Systemverhalten als physikalisch–mathematische Modellgleichungen im kontinuierlichen Zahlenraum. In dieser Arbeit wird ein einheitlicher Rahmen vorgestellt, in dem sich komplexe Systeme mit einer großen Zahl von Freiheitsgraden abbilden lassen. Ein sehr differenziertes Mikrosimulationsmodell (“Celada–Seiden Modell”) wird detailliert vorgestellt und die Antwort des Immunsystems auf Antigene mit Methoden der statistischen Mechanik untersucht. Der zweite Teil der Arbeit, beschäftigt sich mit dem noch jungen Forschungsgebiet der Wirtschaftsphysik (“Econophysics”). Zunächst werden in Kapitel 4 einige Grundbegriffe von Finanzmärkten eingeführt und empirische Beobachtungen beschrieben. Das anschliessende Kapitel 5 stellt kurz bereits existierende Modelle vor, wobei der Schwerpunkt auf der Beschreibung des “Cont-Bouchaud Perkolations Modells” liegt (Abschnitt 5.1). Schließlich wir in Kapitel 6 das im Rahmen dieser Arbeit entwickelte neue Mikorsimulationsmodell zur Abbildung von Aktivitäten an Finanzmärkten diskutiert.

Lebenslauf

Persönliche Daten

Name	Filippo Castiglione
Adresse	Wilhelm–Backhaus–Straße 23, 50931 Köln
geboren am	17.02.1969 in Catania, Italien
Staatsangehörigkeit	italienisch
Familienstand	ledig

Schulbildung

1974 – 1979	Grundschule in Catania
1979 – 1987	Gymnasium in Catania

Zivildienst

1994 – 1995	beim Patronato ACLI, Arezzo, Italien

Studium

1987 – 1993	Studium an der "Università Statale di Milano", Mailand, Italien. Abschluß "Laurea in Scienze dell'Informazione".
08.1991 – 12.1991	ERASMUS-Projekte an der "Universiteit van Amsterdam", Holland
02.1994 – 06.1994	Kurs: Industrielle Modellierung (Uni-Catania mit IBM, Italsiel)
09.1995	Kurs: Paralleles Programmierung Techniken, Italian Center of Parallel Computing (CINECA), Bologna, Italy
08.1996	Forschungaufenthalt beim IBM Watson Labor, New York, und Princeton University, NJ, USA
1998 – 2001	Promotionsstudium der Informatik an der Universität zu Köln. Stipendiat des Graduiertenkollegs *Scientific Computing*

Berufstätigkeit

04.1997 – 03.1998	Ingenieur für Informationstechnik bei ST – Mikroelektronik, Catania
Seit 11.2000	Wissenschaftlicher Assistent am "Dipartimento di Scienze dell'Informazione" der Universität "La Sapienza" zu Rom