Hybrid Simulation Algorithms for an Agent Based Model of the Immune Response

Johannes Textor, Björn Hansen

Institute for Theoretical Computer Science

University of Lübeck

Ratzeburger Alle 160, 23538 Lübeck, Germany

textor@tcs.uni-luebeck.de

Abstract

The immune system is of central interest for the life sciences, but its high complexity makes it a challenging system to study. Computational models of the immune system can help to improve our understanding of its fundamental principles. In this paper, we analyze and extend the Celada-Seiden model, a simple and elegant agent-based model of the entire immune response, which, however, lacks biophysically sound simulation methodology. We extend the stochastic model to a stochastic-deterministic hybrid, and link the deterministic version to continuous physical and chemical laws. This gives precise meanings to all simulation processes, and helps to increase performance. To demonstrate an application for the model, we implement and study two different hypotheses about T cell-mediated immune memory.

Introduction

Researchers who want to find the cure for an auto-immune disease – say, diabetes – would not start right away to experiment on humans. They will experiment on mice first, and this may give them insight that leads to an idea for a new drug. However, the drug will ultimately need to be tested on human volunteers, and it might turn out that what worked for the mouse does not work for the human – which is why biologists speak of the “mouse model”. This work is

about going one step beyond the mouse, and creating a model of the immune system in a computer.

The immune system is highly interesting for researchers in medicine and pharmacology, since immunological discoveries often lead to the development of new drugs and vaccines. It is also a source of inspiration for engineers because of its fault-tolerance, adaptability, and learning capabilities (Stepney et al., 2005). Computational models can contribute in two ways to our understanding of the immune system:

1. *Developing hypotheses* and concepts that are not mature enough to be tested by experiment, or not testable for technical reasons.
2. *Making predictions* that can be verified of falsified by experiment.

Agent-based models of the immune system have been used in both ways (Forrest and Beauchemin, 2007). To make quantitative predictions, models need to achieve a high level of detail, which is difficult because there are big gaps in our knowledge about how the immune system works in detail. These gaps have to be filled by making assumptions that can compromise the accuracy of the predictions (Sol et al., 2003).

The model that we study in this paper falls into the first category. It is a model of the entire immune response, from the first contact with the foreign intruder until the formation of immune memory. However, it is not is a detailed and faithful reproduction of how all these processes work in reality. Even if it were possible to build such a detailed model, it would perhaps be too complex to be of any use. Thus we have to allow for abstractions and simplifications, and we must be aware that our results will always need to be validated by "real" experiments – just like a drug for humans cannot be developed solely by experimenting on mice. Still there are good applications for such a model, and we demonstrate one at the end of this paper.

Our work does not start from scratch, but is based to a large extent on the *Celada-Seiden model* (Seiden and Celada, 1992; Puzone et al., 2002). Several groups have implemented and

used this model, and continue to use it until today (Castiglione et al, 2004, 2007). It covers many important aspects of the immune system in surprisingly simple ways, and we see this simplicity as one of its main strengths. Our goal is to improve its methodological foundation: Based on a precise characterization of its simulation architecture, we will identify ways to improve the realism and performance of the model without sacrificing its simplicity.

But before we go on to explain the contributions of this work in more detail, one essential question needs to be answered: Why do we use an agent-based approach, rather than, for instance, a continuous partial differential equation (PDE) model? Our choice is based on two key aspects of immunity, which lend themselves better to agent-based modelling:

1. *Diversity:* To keep up with the huge, continuously evolving environment of pathogens that the immune system is challenged with, its cells have evolved sophisticated mechanisms to generate diversity. Immune cells carry receptors that are generated randomly by recombination of DNA fragments and additionally fine-tuned by point mutations of the coding sequence. Such highly diverse and dynamic properties are difficult to describe in a PDE system of tractable size, but they can be implemented straightforwardly in an agent-based model.
2. *Randomness:* As a consequence of the high diversity of immune cell receptors, the number of cells that can detect a previously unseen pathogen is very low – Blattmann et al. (2002) estimated that a given antigen is only recognized by 1 in 200 000 T cells. Thus, stochastic factors like the spatial distribution of immune cells at the time of infection can influence the entire infection's course. This fact can be accurately represented in agent-based models using Monte Carlo simulation, while continuous models usually assume that stochastic effects do not play any role because of the large size of the system.

However, agent-based simulations of large biological systems are computationally demanding. In immune system simulations, performance issues typically arise in the late

phase of an immune response, when cells that have detected the antigen create thousands of clones of themselves, which in turn secrete millions of identical antibody molecules. Stochastic effects are no longer important at this stage, and it is no longer necessary to treat each cell and molecule separately. Deterministic models are both more efficient and more descriptive in such situations. The strengths of stochastic and deterministic methods can be combined by a *hybrid* approach: Treat agents as individuals when numbers are low and diversity is high (early phase of the immune response), and treat them in bulk when numbers of agents are large, but the number of different agent *types* is low (late phase of the immune response).

Most implementations of the Celada-Seiden model, which is entirely stochastic, use some hybrid elements for performance reasons. For example, there is an algorithm that iterates through all agents at a given position and moves each agent to a random location in the neighbourhood. Most implementations only actually do this if the number of agents is low. Otherwise, the agents are divided evenly among the neighbouring locations. We will see in the next section that this process is nothing but a finite difference version of the continuous *diffusion equation*. Hence, through the extension the stochastic algorithm to a stochastic and deterministic *hybrid*, a link to a *continuous* physical law is established, which gives us a better understanding of what the stochastic algorithm does. We will show that this does not apply to the other simulation algorithms in the Celada-Seiden model: Either the expected outcome is not known, or the corresponding continuous model is not meaningful. In these cases, we will supply new algorithms, so that the whole simulation can run in stochastic (individual) and in deterministic (bulk) mode, where the latter is in turn a discrete approximation to a set of underlying continuous models.

To demonstrate this link works, we will apply our algorithms to a bistable chemical system that is structurally simple but has interesting dynamics. This system was studied in the literature using both continuous models and discrete stochastic simulations. We will test if our

stochastic simulation framework can reproduce the behaviour predicted by the system's continuous description. Then we discuss our implementation of the humoral immune response, analyzing in particular the impact of the hybrid mode.

Finally, we will demonstrate a concrete application for our model: A new hypothesis about the mechanisms behind immune memory has recently been postulated (Bell and Westermann, 2008). We implement this hypothesis and compare it to the textbook theory. The paper concludes with an evaluation of our results and perspectives for future research.

Immunological foundations of the Celada-Seiden model

We can't cover all the immunological details of the Celada-Seiden model (CS model) in this paper, but it is important to convey the basic idea of how the immune system can be modelled in a computer. We start with a view of the immune system as a classifying system, which helps to understand the functions of the three important types of cells: T cells, B cells, and antigen presenting cells (APC). The second part of this section defines specificity and affinity, and explains the important role of randomness, which is closely related to these concepts.

Note that the facts presented here are stylized and simplified – we present the immunology of the CS model, which is a simplification of reality. The model is based on the standard, "textbook" view of the immune system. There are other theories about the governing principles of immunity, and these are also studied by computational modellers (Aickelin and Greensmith, 2007).

The Immune System as a Classifying System

Borrowing concepts from engineering, we can describe the immune system as a classifying system that tells apart *self* (molecules and cells belonging to the organism) from *nonself* (anything else). While self is tolerated, detected nonself is attacked, which indirectly achieves the desired effect of clearing pathogens and other dangerous substances. However, as a side

effect, benign intruders (like transplanted organs) are neither tolerated by the immune system. Anything that provokes an immune response is called an *antigen*.

Antibody is an effective and highly specific defence mechanism against antigen. It is a family of molecules that are specifically produced to bind to the surface of a target antigen. Once an antigen has been tagged by antibody, non-specific scavenger cells can localize and attack it efficiently. The subsystem of the immune system that produces antibody is called the *humoral immune response*. This subsystem is the focus of the Celada-Seiden model.

To perform the classification of substances into self and nonself, the immune system employs two complementary sets of detectors: *B cells* and *T cells* (Figure 1). While B cells detect structures on the antigen *surface*, T cells are mainly concerned with structures *inside* the antigen. But since T cells cannot reach inside an antigen, they need the help of so-called *antigen presenting cells* (APC), which phagocyte antigen and present digested parts of it on their surface, where T cells can access it. In addition to dedicated APCs, B cells themselves can also present antigen.

During the early phase of a humoral immune response, only a few T and B cells react to the antigen. To initiate a large-scale reaction, these cells have to meet and exchange signals that stimulate them to divide (costimulation). Through several cell division cycles, a large number of cell clones bearing antigen-matching receptors are generated. Finally, antibody is produced by secretion of B cell receptors. Antigen-secreting B cells are also called *plasma cells.*

Specificity, Affinity, and Randomness

Since antigen can have just about any shape and size, the immune system must employ a huge number of different receptors to cover the antigen shape space. It would be impossible to encode all these receptors directly into the DNA. Instead, they are generated randomly by a process involving DNA recombination and mutation. According to Janeway et al. (2005), this

process can generate about 10^8 different B cell receptors. The term *specificity* refers to this covering of the antigen shape space with a large number of specific receptors.

This leads to a high sensitivity of the immune response to random effects: the number of cells that can detect a pathogen is often so low that the localization of these cells in the organism at the time of infection can determine whether a successful response is mounted or not. Hence, B and T cells are highly motile and continuously circulate through the organism, a fact that is reflected in the CS model by letting the agents execute a *random walk* through the simulation space.

The ability of an antibody (B cell receptor) or a T cell receptor to bind to a given antigen is called *affinity*. The binding strength is determined by the shapes of receptor and target (lock-and-key-principle), but also by electrical and chemical forces. A sound representation of the concepts of specificity and affinity is an essential part of any model of the immune system. In the CS model, this is achieved using a simple but powerful concept: Antigen and receptor shapes are modeled as *bit strings* (Figure 2).

Formally, the affinity between two bit strings $s, t \in \{0,1\}^N$ is calculated as follows using a function α of their hamming distance $d = \sum_i s_i \oplus t_i$:

$$\alpha(d) = \begin{cases} 0 & d < d_{\min} \\ \alpha_{\min} & d = d_{\min} \\ \min\left(1, \alpha_{inc}\alpha(d-1)\frac{d}{N-d+1}\right) & d > d_{\min} \end{cases} \quad (1)$$

Hence, s and t must have a hamming distance of at least $d_{\min}$ for nonzero affinity. The affinity then increases with each additional complementary by a factor proportional to α_{inc}, and reaches its maximum when s is the bit-wise complement of t.

In this bit string model, it is straightforward to implement random receptor generation and receptor mutations. The size of the shape space is determined by N. At $N = 27$ bits, we obtain a size that is comparable to the human B cell repertoire (approximately 10^8).

The Simulation Environment

A model of the immunological processes described above will contain cells and molecules that move around, get in contact to each other, sometimes ingest each other, and create new cells and molecules by cell division and secretion. In this section, we leave the immunology aside and focus on how these processes are organized. For instance, can cells move around as they please, or do we restrict their movement? How do we decide which cell meets which, and which is allowed to ingest the other? What if several cells wish to clone themselves, but there is not enough room available? The task of the *simulation environment* is to resolve such conflicts, and generally to orchestrate the movement, actions, and interactions of agents. This means that our model has a central point of control, which implies a rather weak notion of an agent. In fact, the CS model is also sometimes referred to as a "generalized" cellular automaton and not as an agent-based model (Seiden and Celada, 1992). However, this characterization is misleading: cellular automata lack the notion of autonomously acting individuals. In the CS model, there is a clear conceptual difference between actors and environment. The constraints imposed by the environment are a necessary reflection of physical reality, but the actual immunology is not implemented in the environment, but in the concrete actions and interactions of the agents. This will be illustrated in the next section, where we use our simulation environment to model a process that has nothing to do with immunology; in the subsequent section, we return to immunology and the actions and interactions between agents in our immune system simulation. The regulation function of the environment in agent-based modelling in general is currently being investigated (Schumacher and Ossowski, 2006; Bandini and Vizzari, 2007).

In the following subsections, we analyze and modify the algorithms that organize agent motility, interactions, and cell division in the CS model. Our requirements to such algorithms are (1) that both a stochastic and a deterministic mode is available, and (2) that the

deterministic mode corresponds to a meaningful continuous model. Some of the analyzed algorithms do not meet these requirements, and we will replace them with our own versions. To distinguish the original CS model algorithms from our replacements, we mark the former with a suffix: INT_{CS} is the CS model's interaction algorithm, and INTMA is our replacement (the suffix MA is explained later). Our primary source is the informal description of the model as published by Celada and Seiden (1992), but this source does not contain all details. Hence, we use the most recent and complete implementation of the CS model, CImmSim version 6.3 (Castiglione et al., 1997; Bernaschi et al., 2001), for additional reference.

The CS model is lattice-based, and we adopt its two-dimensional hexagonal lattice with wrap-around boundary conditions, although an extension to three dimensions might be useful in the future. The properties of agents can be visible to other agents (such as receptors), or internal (age). The simulation alternates between an *action phase*, in which agents can perform actions and interact with other agents on the same site, and a *diffusion phase*, in which agents are moved to neighbouring lattice sites.

The complete information that defines an agent's state, i.e. both external and internal properties, are called its *type* T. All algorithms treat agents grouped by their type, or pairs of agents grouped by pairs of types. There are a number of global parameters and utility functions, which all algorithms may use (Table 1). Of these, the parameters $C_{\max}$, S_T and D_T do not exist in the original CS model, but are natural extensions to represent an object's size and diffusivity more explicitly. The orginal CS model does not make type-dependent distinctions (S_T and D_T are constants independent of T), and the capacity of each site is unlimited.

Motility of Cells and Molecules

Agents in the CS model cannot move around the lattice autonomously, but are moved to random neigbouring sites during the diffusion phase. This is implemented using algorithm

DIFF$_{CS}$ (Figure 3). The link between the discrete random walk carried out by the agents on the lattice and the continuous notion of diffusion is established by considering the expected outcome of algorithm DIFF$_{CS}$. Let $N(s)$ denote the set of neighbours of the lattice site s, which has 6 elements on a hexagonal 2D lattice. On average, executing DIFF$_{CS}$ changes the number of agents $C_T(s,t)$ as follows:

$$C_T(s+\Delta t, s) - C_T(s,t) = \Delta t \frac{D_T}{6(\Delta s)^2} \sum_{s' \in N(s)} C_T(s',t) - C_T(s,t) \quad (2)$$

The expression on the right hand side is well known as the discrete Laplacian operator. Dividing by Δt and letting $\Delta t, \Delta s \to 0$, we obtain

$$\frac{\partial C_T}{\partial T} = \frac{D_T}{4} \nabla^2 C_T \quad (3)$$

where C_T is now a continuous function in space and time giving the *concentration* of agents with type T. This equation is identical to the physical law of diffusion. Thus, algorithm DIFF$_{CS}$ is in fact a discrete stochastic realization of a diffusion process.

Interactions between Cells and Molecules

The interactions between agents and their outcomes are very diverse – for example, one of the interacting agents could start a complex internal process, while the other one is killed (phagocytosis). The only common feature of all interactions is that the agents need to be co-located. Obviously, it is not possible to let all agents at a lattice site interact with all others at the same time. Algorithm INT$_{CS}$ (Figure 4) choses pairs of agents for interaction using the following simple procedure: For each pair of agents on a given lattice site, a coin is thrown. On success, the current pair of agents is set aside to perform the interaction, and is not eligible for further interactions. Despite the simplicity of INT$_{CS}$, it is surprisingly difficult to determine the expected number of interactions. The problem is that each iteration of the inner loop

depends on the outcomes of all previous iterations. Until now, we were unable to derive a closed formula for the expected outcome of INT_{CS}.

In addition to this difficulty, the algorithm is unrealistic because it is not aware of the number of agents located at the current site. One would expect the rate of actions that require collocation of two entities to increase with the concentration of these entities. The chemical *law of mass action* states that the rate of interactions is proportional to each of the interacting species' concentration. This law is often used in continuous models of the immune system (Carneiro, 2005).

The law of mass action is implemented as a discrete stochastic version by Algorithm INTMA (Figure 5). It is easy to see that the expected number of calls to the interaction procedure is equal to $I_{T,T'} \cdot C_T \cdot C_{T'}$ and thus proportional to the concentrations of the interacting agent types. In addition to the number of agents, their relative sizes and the spatial model resolution are taken into account.

Cell Population Growth

Cell division occurs in the CS model when immune cells get activated. The normal homeostatic turnover of cells does not involve cell division –new are inserted and existing cells deleted at constant rates, defined by a cell type dependent half-life parameter. The antigen, on the other hand, divides at a constant rate.

In the CS model, proliferation of activated cells is completely unlimited, and each lattice site can contain an arbitrary number of cells, which is not realistic. In CImmSim, the growth rate of cells is damped using a Gaussian kernel to slow proliferation down as lattice sites become more populated, but there is no mechanism that removes cells when a site becomes overpopulated.

Murray (2002) discusses several continuous models for constrained growth of populations. Algorithm PROLIFLOG (Figure 6) is a discrete stochastic version of the *logistic equation:*

$$\frac{\partial C_T}{\partial T} = P_T \cdot C_T \cdot \frac{C_{\max} - C_T}{C_{\max}} \tag{4}$$

This way, PROLIFLOG imposes a capacity limit $C_{\max}$ on the number of cells per site. Temporal overpopulation of a site is still possible by diffusion from neighbouring sites, but in this case cells are killed at a rate proportional to the amount of overpopulation until the maximum capacity is no longer exceeded. This is important for our diffusion and interaction algorithms to provide meaningful results.

Switching between Stochastic and Deterministic Mode

We have argued above that randomness is an important part of the model's design. However, the computational cost is high: For instance, algorithm INT_{CS} needs $O(n^2)$ random numbers, where n is the number of agents. It is thus desirable to use these stochastic algorithms only when the number of agents are low, and replace them by equivalent deterministic versions when numbers are large. This can be done for the algorithms DIFF_{CS}, INTMA, $\text{PROLIF}_{\text{CS}}$, and PROLIFLOG, since we know their expected outcomes. The switching is technically realized as follows: If the number of agents of all types treated by the algorithm exceeds a user-controlled threshold, then the deterministic version is used instead of the stochastic one. The deterministic version of each algorithm directly produces the expected outcome of its stochastic counterpart; however, since agent numbers are still discrete, a small part of the agents may still have to be treated stochastically for a consistent result. Figure 7 shows the deterministic version of algorithm INTMA to illustrate this procedure.

Simulating a Bistable Chemical System

The simulation environment can now be implemented as a stochastic and deterministic hybrid, but it remains entirely discrete, and the connection to the underlying continuous models is not explicit. Thus, it remains to show that this connection actually works. In this

section, we take a simple system for which both a continuous description in terms of differential equations and a discrete description in terms of interacting particles exist, and simulate this system with our model. Our goal is to show that our discrete stochastic algorithms reproduce the behaviour predicted by the system's continuous description. Such a test cannot be done with our simulation of the immune system itself, since we can't entirely describe this simulation with PDEs. The bistable chemical system we use was introduced by Malevanets and Kapral (1997), who studied it using a Monte Carlo simulation that is structurally very similar to our model.

The System

We consider four hypothetical chemical species $A, A^*, B,$ and B^* and the chemical reactions between them (Figure 8). The species A and A^* can be seen as two alternative states of one species. Both A and A^* contribute to their own enrichment: A converts A^* to A and vice versa. Without additional reactions, the system would not be very interesting – from any initial state, either A or A^* would ultimately become extinct, and the system would end up in a trivial state. The species B and B^* are coupled with A and A^* in a cyclic manner, and thus remove these dead ends from the system. Because B and B^* diffuse faster than A and A^*, the system attains the ability to form *locally* stable states, which can form interesting spatial patterns.

A continuous description of these chemical reactions is derived by applying the law of mass action to the reaction equations and adding diffusion terms, as discussed by Malevanets and Kapral (1997). This results in a nice "sandbox" model that is well suited to study the interplay between molecular interactions and diffusion. Note that in contrast to our simulation framework, the one used by Malevanets and Kapral is designed to allow a rigorous mathematical analysis, but is computationally more demanding. For instance, at most one

particle per lattice site is allowed to move per diffusion phase and at most one pair or triple of particles is allowed to interact during an action phase.

Experiments and Results

Since the reactions between A and A^* involve triples rather than pairs of agents, we needed to create a variant of algorithm INTMA that works with three types of agents (Figure 9). A^* can be converted to A even if only one agent of type A is present, but the algorithm does follow the law of mass action. The corresponding adaptation of algorithm INT_{CS} simply loops through all possible triples of agents instead of all possible pairs (algorithm not shown).

We simulated the described system using three sets of algorithms: (1) those described by Malevanets and Kapral, (2) DIFF_{CS} and INT_{CS}, and (3) DIFF_{CS} and INTMA. The simulations were carried out on a 300x300 square lattice, using the general simulation framework of alternating diffusion and interaction phases described above. With algorithm INT_{CS}, each agent was only allowed to interact once per interaction phase (as in the CS model). This limitation was not used with algorithm INTMA.

The results are shown in Figure 10. All algorithm sets form stable Turing patterns in certain parameter regions. The parameters given by Malevanets and Kapral (1997) can be used directly with algorithms DIFF_{CS} and INTMA, while we had to determine the corresponding values for algorithm INT_{CS} by isolating the diffusion and interaction parts and tuning algorithm INT_{CS} to produce similar results as INTMA.

The results reveal an important property of algorithm INT_{CS}: for diffusion to play any role at all, it has to be used with very low parameter values. Typical parameter values from CImmSim, for example, range from 0.01 (unspecific phagocytosis of antigen by macrophages) to 1 (specific phagocytosis of antigen by maximum-affinity B cells). In this parameter range, the expected number of interactions is close to maximal, and the diffusion algorithm is no longer able to maintain a locally smooth distribution of agents.

Conclusion

The fact that Turing patterns can be generated with all studied algorithms shows that the addition of randomness does not break the link to the continuous system description, at least in certain parameter ranges. The randomness causes “noise” in the Turing patterns, but does not destroy them. It is interesting to see that this works even with the CS algorithms, which were not designed to be biophysically meaningful. However, it remains problematic that the parameters for the CS algorithms have to be determined by tuning, which makes it cumbersome to use parameters from the literature.

Simulating the Immune Response

Our next goal is to assess how our simulation algorithms perform when applied to simulating the immune response. The first part of this section describes our implementation of the Celada-Seiden model using the modified simulation environment. We perform some experiments to test if the qualitative outcome of the simulation remains valid (for no meaningful quantitative comparison is possible), and to assess the impact of the hybrid simulation mode. The second part of this section illustrates a potential application of the model: We compare two different hypotheses on how T cell-mediated immune memory works.

Experiments and Results

Our model of the immune response is essentially a “port” of a subset of CImmSim, version 6.3, to our simulation environment. It is implemented in C++, and the cellular and molecular agents are arranged in an object hierarchy, which facilitates extending the model by new types of molecules and cells. As a technical improvement, the simulation can be split in multiple threads. For reference, we made the implementation available under the terms of the GNU General Public License. It can be downloaded from http://www.tcs.uni-

luebeck.de/forschung/software/limmsim/ or obtained by e-mail from the corresponding author.

The simulation includes B cells, T cells, antigen presenting cells, antigen and antibody and the interactions between them. Motility, interactions, and proliferation are implemented using algorithms DIFF$_{CS}$, INTMA, and PROLIFLOG, respectively. A complete list of the implemented processes and their rate parameters is shown in Tables 3 and 4. The capacity of each lattice point was limited to 64 cells per kind (B cells, T cells, APC and antigen), excluding antibody, on which no limit was imposed. The parameters used for the affinity function were $N = 16$ for B cells, $N = 8$ for T cells, $d_{\min} = N - 2$, $\alpha_{\min} = 0.05$, and $\alpha_{inc} = 0.77$.

Figure 11 shows the distribution of antigen and antibody during the course of an example simulation run on a 64x64 lattice. The antigen is injected at $t = 100$ and starts spreading across the lattice. Around $t = 800$, antibody starts to appear in three different regions on the lattice, where specific B and T cells have met and exchanged costimulatory signals, resulting in the generation of plasma cells. The antigen is almost cleared at $t = 1500$. The last image shows the beginning degradation of antibody molecules ($t = 3000$). Thus, our simulation is capable of producing the same qualitative behaviour as the original Celada-Seiden model.

To analyze the effects of the hybrid mode on performance and results, we compared the results of several simulation runs using (1) no thresholding, (2) deterministic mode for ≥ 64 agents, and (3) deterministic mode for ≥ 32 agents. Note that a threshold of 64 almost exclusively affects the interactions and diffusion of antibody, for which the capacity limit does not apply (the number of cells at a site can only temporarily exceed its capacity, and only by diffusion). For all simulation runs, we measured: the total running time, the time at which the first antibody appeared, the time at which the last antigen was cleared, and the maximum total number of antibodies. The simulations were run for 3000 time steps. The results are shown in Figure 12: Setting the threshold to 64 agents decreased the mean simulation time by about 40%, but thresholding at 32 particles did not increase the performance further. The

quantitative measurements were not significantly affected by the choice of the threshold, but interestingly, the standard deviation increased when the switching was used.

A Case Study: T cell immune memory

Immune memory is thought to be formed by conversion of short-lived effector T- and B cells to long-lived *memory cells*. When an already known antigen re-enters the organism, there is already a reservoir of mature, specific T and B cells available that can quickly and efficiently mount a secondary response.

There is one problem with this commonly accepted theory (Bell and Westermann, 2008): While the existence of memory B cells is well supported, there is no convincing evidence yet that T cells can be permanently altered by antigen stimulation. Memory T cells differ from other T cells only in features that are known to be reversible. Bell and Westermann suggest that immunological memory does not need a memory T cell: antigen could persist in the organism in processed (presented) form, continue to stimulate naive T cells, and thereby indirectly maintain a population of matching effector T cells. This hypothesis requires that T cells can go back to the naive state, since otherwise the persistence of antigen would eventually result in programmed cell death of all matching T cells. The persistent antigen could be stored in a *memory antigen presenting cell* (Figure 13).

The two theories can be modelled by enhancing the simulation as follows:

1. T memory cell hypothesis:
    - B cells can become memory cells instead of plasma cells (rate=0.05)
    - Effector T cells can become memory cells (rate=0.00125),
2. Persistent antigen hypothesis:
    - B cells that have gone through all proliferation cycles can become memory cells (rate=0.05)
    - Proliferating T cells can revert to the naive state (rate=0.00125)

- APCs that present antigen can become memory cells (rate=0.00125). Memory APCs stick with their presented antigen, and do not remove it like normal APCs do.

The rates were chosen by exploration of the parameter space. Figure 14 shows the results of two simulation runs to illustrate an observation that would be interesting to investigate further: Both memory models cleared a previously seen antigen more efficiently, but the persistent antigen model achieved immunological memory with lower total number of T cells. The reason could be that due to the persistent antigen, the proliferation of specific T cells always keeps going at a low intensity, which might cause a lesser extent of T cell "overproduction" at the second infection. However, it would be premature to draw immediate conclusions from this observation – further work is necessary.

Discussion and Conclusions

Recently, there has been considerable activity in the field of computational immunology that involved agent-based or similar models (Beltman et al., 2006; Meier-Schellersheim et al., 2006; Maini et al., 2008). In contrast to the early work of Celada and Seiden, these models focus on more specific parts of the immune system and achieve a higher level of detail, which reflects a general trend in immunology to focus more on the details than on the "big picture". However, the debate on the underlying principles of the immune system is still far from settled, and some fundamental concepts such as the existence of T memory cells lack experimental support. Continuous modelling techniques dominate the analysis of such fundamental questions (Carneiro et al., 2005), but there are good reasons to adopt an agent-based approach, as explained in the beginning of this paper. The Celada-Seiden model is simple and elegant, but nevertheless includes the most important aspects of the humoral immune response, from the first infection to the formation of immune memory. These were

our reasons to adopt the Celada-Seiden model; our main problem with it was its lack of biophysically sound simulation methodology.

Our methodological analysis focused on the simulation environment, which organizes the motility, interactions, and reproduction of agents. Seeking to implement these processes in a more efficient and meaningful way, we added the ability to switch between stochastic and deterministic simulation modes, and established conceptual links to continuous models. We tested this link by modelling a bistable chemical system, and found that our stochastic simulation environment reproduced the characteristic behaviour (Turing patterns) predicted by the system's continuous description.

We then moved on to discuss our implementation of the immune response. An analysis of the hybrid mode confirmed our expectation that performance should increase, but further work is necessary to fully understand the hybrid mode's impact. For example, it is unclear why switching to the deterministic mode for large numbers of agents leads to an increase of the standard deviation of all measured quantities. Generally, the switching itself does not lead to a significant improvement of the simulation effort – the important part is the resulting link to continuous models.

Analyzing the T cell memory models gave us some insight by forcing us to think about these models in a different way. However, it is unsatisfactory that most parameters had to be "guessed". A recent discovery in immunology could improve this situation substantially: The search of T cells and B cells for antigen does indeed seem to be realized by a random walk (Miller et al., 2002; Halin et al., 2005), and so the Celada-Seiden model is closer to reality than previously thought. The quantitative characteristics of this random walk are being investigated, and some estimates for parameters that also exist in the Celada-Seiden model have already been made – for example, the number of contacts between T cells and APCs per hour (Beltman et al., 2006). Our link to quantitative chemical and physical laws makes it

possible to incorporate such results into the model, which would be a promising direction for future work.

Acknowledgements

JT thanks Filippo Castiglione for making CImmSim available for this research, Roberto Puzone for his valuable comments about the simulation space in the Celada-Seiden model, and Jürgen Westermann for advice on the T cell memory models.

References

Aickelin, Uwe, and Julie Greensmith. 2007.

Sensing Ranger: Innate Immunology for Intrusion Detection.

Information Security Technical Reports 12: 218-27.

Bandini, Stefania, and Giuseppe Vizzari. 2007.

Regulation Function of the Environment in Agent-Based Simulation.

In *Environments for Multi-Agent Systems III*, ed. Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, 157-69. Springer Lecture Notes in Computer Science 4389.

Bell, Eric B., and Jürgen Westermann. 2008.

CD4 memory T cells on trial: immunological memory without a memory T cell.

Trends in Immunology, in press.

Beltman, Joost B., Stan Marée, and Rob J. de Boer. 2007.

Spatial modelling of brief and long interactions between T cells and dendritic cells.

Immunology and Cell Biology 85: 306-14.

Blattmann, Joseph N., Rustom Antia, David J.D. Sourdive, Xiaochi Wang, Susan M. Kaech, Kaja Murali-Krishna, John D. Altman, and Rafi Ahmed. 2002.
Estimating the precursor frequency of naive antigen-specific CD8 T cells.
Journal of Experimental Medicine 195: 657-64.

Bernaschi, Massimo, and Filippo Castiglione. 2001.
Design and implementation of an immune system simulator.
Computers in Biology and Medicine 31: 303-31.

Carneiro, Jorge, Tiago Paixão, Dejan Milutinovic, João Sousa, Kalet Leon, Rui Gardner, and Jose Faro. 2005.
Immunological self-tolerance: Lessons from mathematical modeling.
Journal of Computational and Applied Mathematics 184: 77-100.

Castiglione, Filippo, Massimo Bernaschi, and Sauro Succi. 1997.
Simulating the immune response on a distributed parallel computer.
Int J Mod Phys C 8: 527-45.

Castiglione, Filippo, Filippo Poccia, Gianpiero D'Offizi, and Massimo Bernaschi. 2004.
Mutation, fitness, viral diversity and predictive markers of disease progression in a computational model of HIV-1 infection.
AIDS Research and Human Retrovirus 20: 1316-25.

Castiglione, Filippo, Francesco Pappalardo, Massimo Bernaschi, and Santo Motta. 2007.
Optimization of HAART with genetic algorithms and agent-based models of HIV infection.
Bioinformatics 23: 3350-55.

Efroni, Sol, David Harel, and Irun R. Cohen. 2003.

Toward rigorous comprehension of biological complexity: Modeling, evaluation, and visualization of thymic T-cell maturation.

Genome Research 13: 2485-97.

Figge, Marc Thilo, Alexandre Garin, Matthias Gunzer, Marie Kosco-Vilbois, Kai-Michael Toellner, and Michael Meyer-Herrmann. 2008.

Deriving a germinal center lymphocyte migration model from two-photon data.

Journal of Experimental Medicine 205: 3019-29.

Forrest, Stephanie, and Catherine Beauchemin. 2007.

Computer immunology.

Immunology Reviews 16: 176-97.

Halin, Cornelia, J. Rodrigo Mora, Cenk Sumen, and Ulrich H. von Andrian. 2005.

In vivo imaging of lymphocyte trafficking.

Annual Review of Cell and Developmental Biology 21: 581-603.

Hansen, Björn. 2007.

Partikelsimulation einer Reaktions-Diffusionsgleichung.

BSc Diss., University of Lübeck, Germany.

Janeway, Charles A. jr., Paul Travers, and Mark Walport. 2005.

Immunobiology (6th edition).

Garland Science.

Malevanets, Anatoly, and Raymond Kapral. 1996.

Links, knots, and knotted labyrinths in bistable systems.

Physical Review Letters 77: 767-70.

Miller, Mark J., Sindy H. Wei, Ian Parker, and Michael D. Cahalan. 2002.

Two-photon imaging of lymphocyte motility and antigen response in intact lymph node.

Science 296: 1869-73.

John D. Murray. 2002.

Mathematical Biology (3rd edition).

Springer.

Meier-Schellersheim, Martin, Xuehua Xu, Bastian Angermann, Eric J. Kunkel, Tian Jin, and Ronald N. Germain. 2006.

Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method.

PLoS Computational Biology 2: e82

Puzone, Roberto, Brynja Kohler, Philip E. Seiden, and Franco Celada. 2002.

IMMSIM, a flexible model for in machina experiments on immune system responses.

Future Generation Comp. Syst. 18: 961-72.

Schumacher, Michael, and Sascha Ossowski. 2006.

The Governing Environment.

In *Environments for Multi-Agent Systems II,* ed. Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, 88-104. Springer Lecture Notes in Computer Science 3830.

Seiden, Philip E., and Franco Celada. 1992.
A model for simulating cognate recognition and response in the immune system.
Journal of Theoretical Biology 158: 329-57.

Stepney, Susan, Robert E. Smith, Jonathan Timmis, Andy M. Tyrrell, Mark J. Neal, and Andrew N. W. Hone.
Conceptual frameworks for artificial immune systems, 2005.
Journal of Unconventional Computing 1: 315-38.

Textor, Johannes. 2006.
Ein hybrides Automatenmodell zur Simulation des Immunsystems.
MSc Diss., University of Lübeck, Germany.

Figures

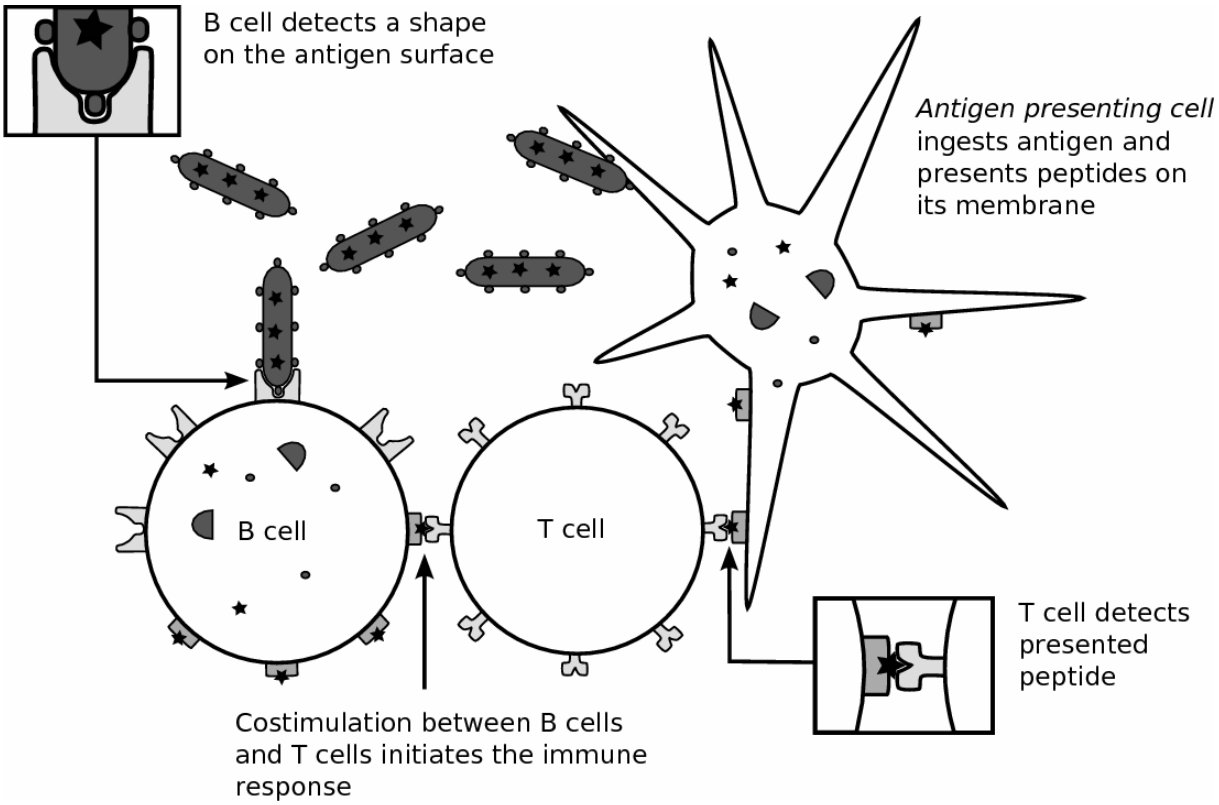

Figure 1: Antigen detection by cooperation of T cells, B cells, and antigen presenting cells during the humoral immune response.

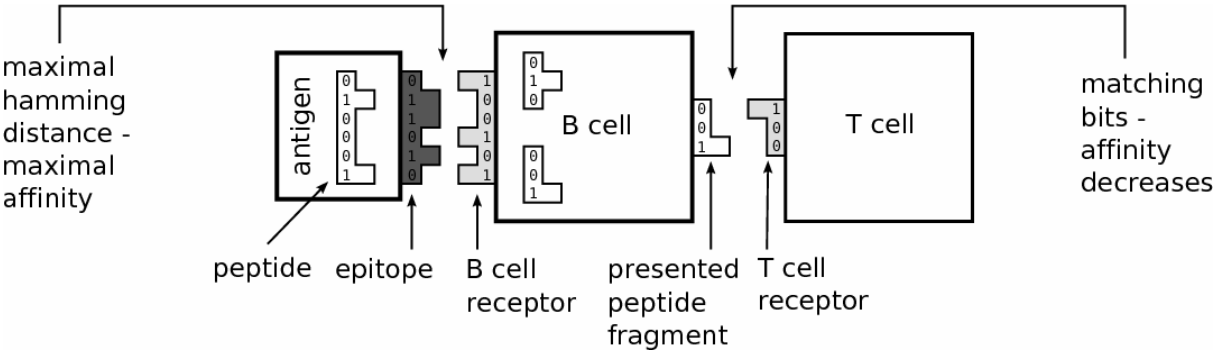

Figure 2: The bit string model for affinity between receptors and antigen.

algorithm DIFF$_{CS}$

---

1 **argument** T : agent type
2 **do for each** site s
3 **do for each** agent a with type T at s
4 **if** $COIN\left(\frac{\Delta t}{(\Delta s)^2} D_T\right)$
5 target[a] $\leftarrow$ random neighbour of s
6 **else**
7 target[a] $\leftarrow$ s

Figure 3: Diffusion algorithm from the CS model. The actual translocation of agents is not part of this algorithm: it is done simultaneously after all targets have been determined.

algorithm INT_{CS}

```
1   arguments T , T′: agent types
2   do for each site s
3       L ← list of agents with type T at s
4       L′ ← list of agents with type T′ at s
5       do for each a ∈ L
6           do for each a′ ∈ L′
7               if COIN(I_{T,T′})
8                   interact(a, a′)
9                   remove a′ from L′
10                  continue using next a
```

Figure 4: CS model algorithm for interactions between agents. Note that the order of agents in the lists L and L' does not matter, since all agents of one type are completely identical.

algorithm INTMA

```
1   arguments T , T′: agent types
2   do for each site s
3     L ← list of agents with of type T at s
4     L′← list of agents with type T′ at s
5     I_max ← size( L )
6     ρ ← size( L′) · (S_T′ / C_max )
7     do I_max times
8       if COIN(ρ · Δt · I_T,T′ )

9           interact(first( L ), first( L′))
10          remove first element from L
11          remove first element from L′
```

Figure 5: Proposed interaction algorithm that reproduces the law of mass action. For simplicity, it is assumed that size(L) ≤ size(L') holds at line 5 such that the maximum number of interactions does not exceed the possible number of interactions.

algorithm PROLIFLOG

```
1   argument T : particle type
2   do for each site s
3       ρ ← 0
4       do for each agent a at s
5           ρ ← ρ + S_type(a) / C_max

6       L ← list of cells of type T at s
7       if Δt · P_T · (1 − ρ) > 0
8           do for each c ∈ L
9               if COIN(P_T · (1 − ρ))
10                  clone(c)
11      else
12          do for each c ∈ L
13              if COIN(−P_T · (1 − ρ))
14                  kill(c)
```

Figure 6: Proposed algorithm for cell division (proliferation). The capacity of each site is limited: Cells can only divide when enough space is left (lines 7-10), otherwise cells may die due to overpopulation (lines 11-14).

algorithm INTMA (deterministic version)

```
1   arguments T , T′ : agent types
2   do for each site s
3     L ← list of agents with type T at s
4     L′ ← list of agents with type T′ at s
5     I_ex ← I_{T,T′} · Δt · size(L) · size(L′) · (S_{T′} / C_max)
6     do floor(I_ex) times
7         interact(first(L), first(L′))
8         remove first element from L
9         remove first element from L′
10    if COIN(I_ex − floor(I_ex))
11        interact(first(L), first(L'))
```

Figure 7: Deterministic version of algorithm INTMA. As in Figure 5, we assume that size(L) ≤ size(L') holds at line 5. Lines 10 and 11 are necessary to keep the expected outcome consistent with the stochastic version.

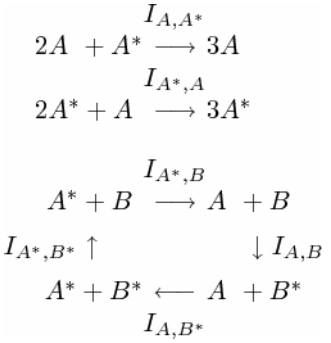

Figure 8: Chemical equations of the studied bistable system. Each reaction has an associated rate parameter $I_{T,T'}$ where T and T' are the interacting species.

algorithm INTMA3

---

1 **do for each** site s
2 $L \leftarrow$ agents of type A at s
3 $L' \leftarrow$ agents of type A^* at s
4 $I_{\max} \leftarrow \text{size}(L')$
5 $\rho \leftarrow (\text{size}(L) \cdot (S_A / C_{\max}))^2$
6 **do** $I_{\max}$ **times**
7 **if** $COIN\left(\rho \cdot \Delta t \cdot I_{A,A^*}\right)$
8 **move agent from** L' **to** L

Figure 9: Extension of algorithm INTMA to interactions between three agents. The interactions shown here correspond to the chemical equation $2A + 2A^* \rightarrow 3A$.

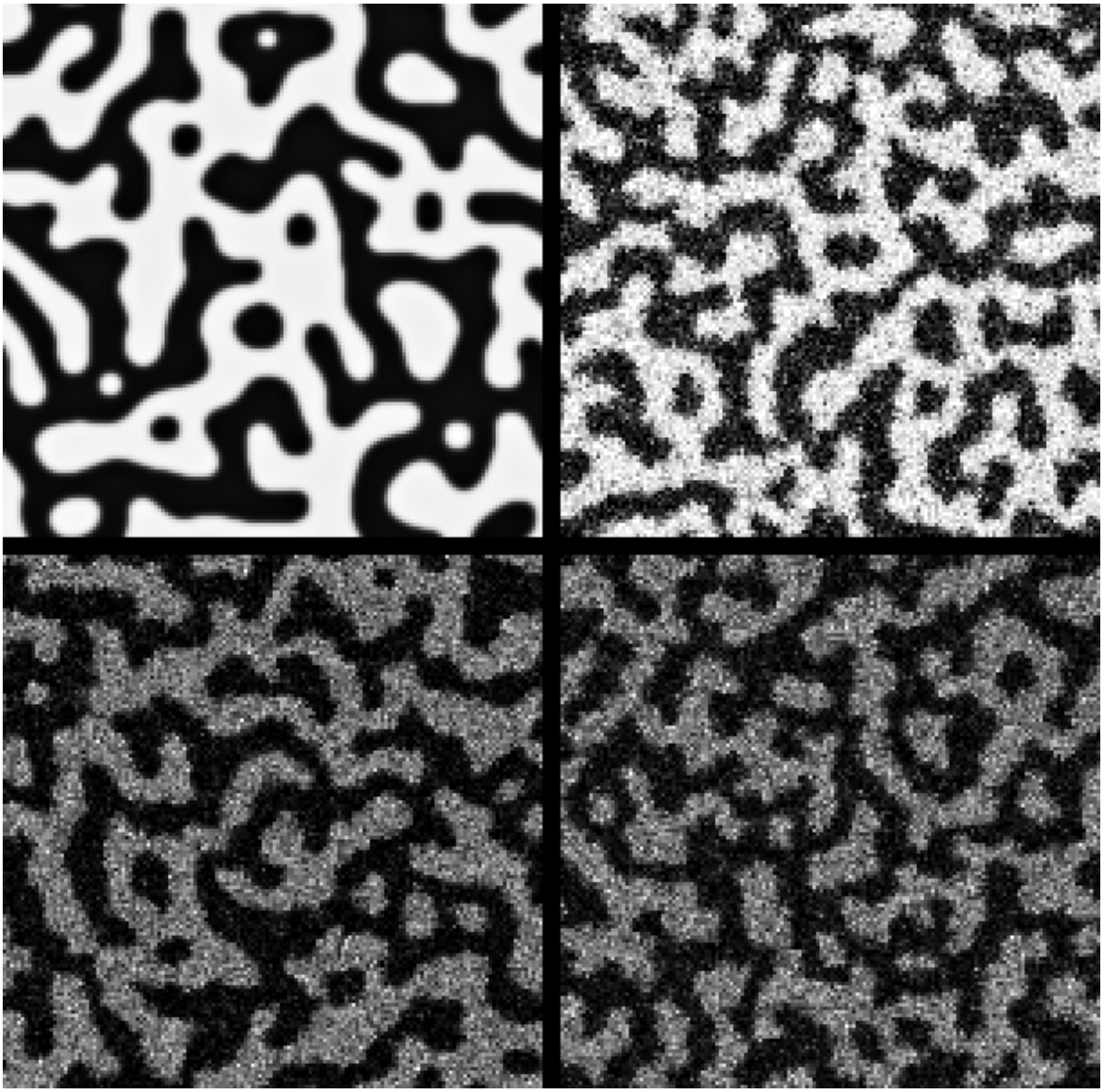

Figure 10: Simulation results for the bistable chemical system. Top left: Solution of the system equations for random initial conditions.Top right: Particle simulation as of Malevanets and Kapral. Bottom left: Simulation using algorithms DIFF_{CS} and INT_{CS}. Bottom right: Algorithms DIFF_{CS} and INTMA.Colors are mapped to the concentrations of A (white: maximal); the bottom images appear darker because of a larger variance of concentrations.

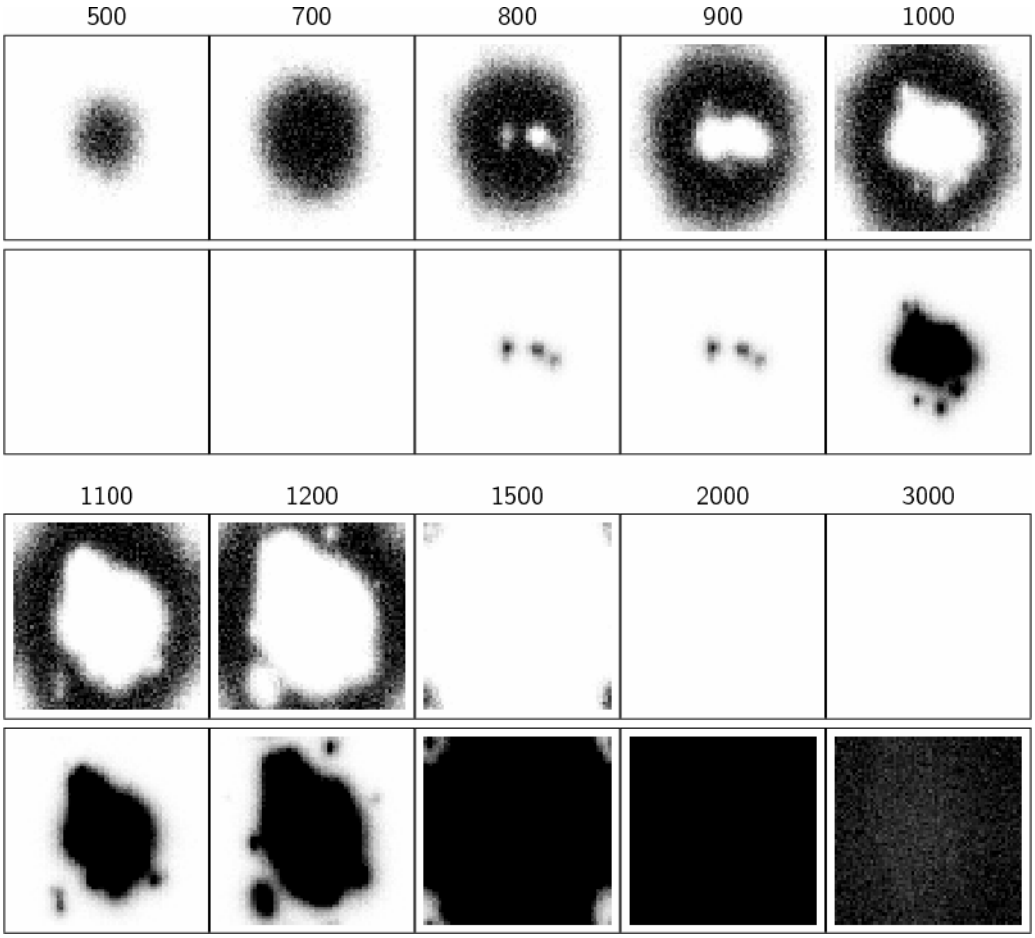

Figure 11: Distribution of antigen (rows 1 and 3) and antibody (rows 2 and 3) during a simulation on a 64x64 hexagonal lattice that was run for 3000 time steps (white: no agents, black: 64 or more). The antigen was injected at $t = 100$.

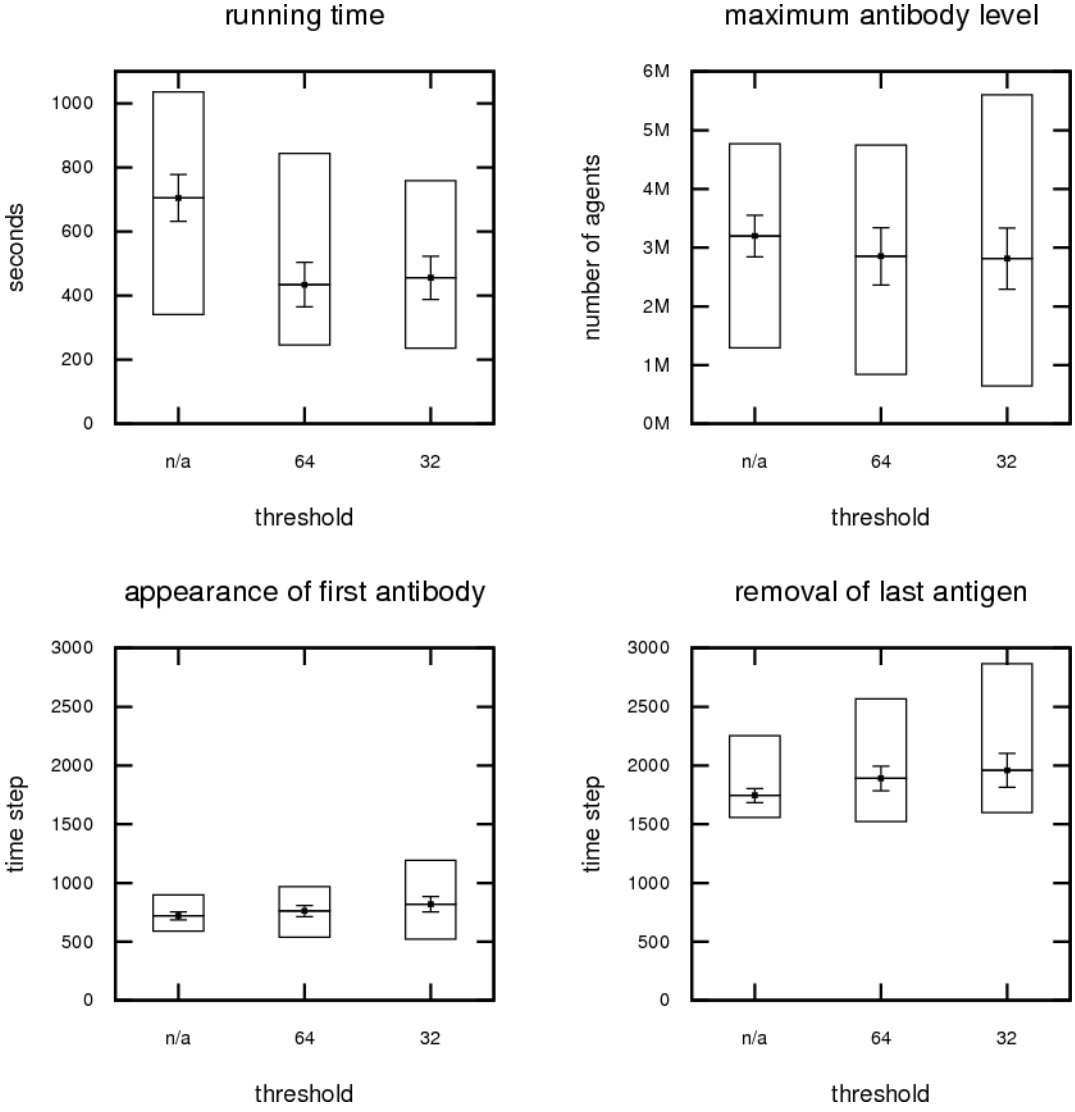

Figure 12: Measurements of running time and characteristics of the simulated immune response on a 64x64 lattice. Mean, minimum, maximum (boxes), and standard error of the mean (error bars) are shown for 10 runs per value of the threshold between deterministic and stochastic mode (n/a: stochastic mode only).

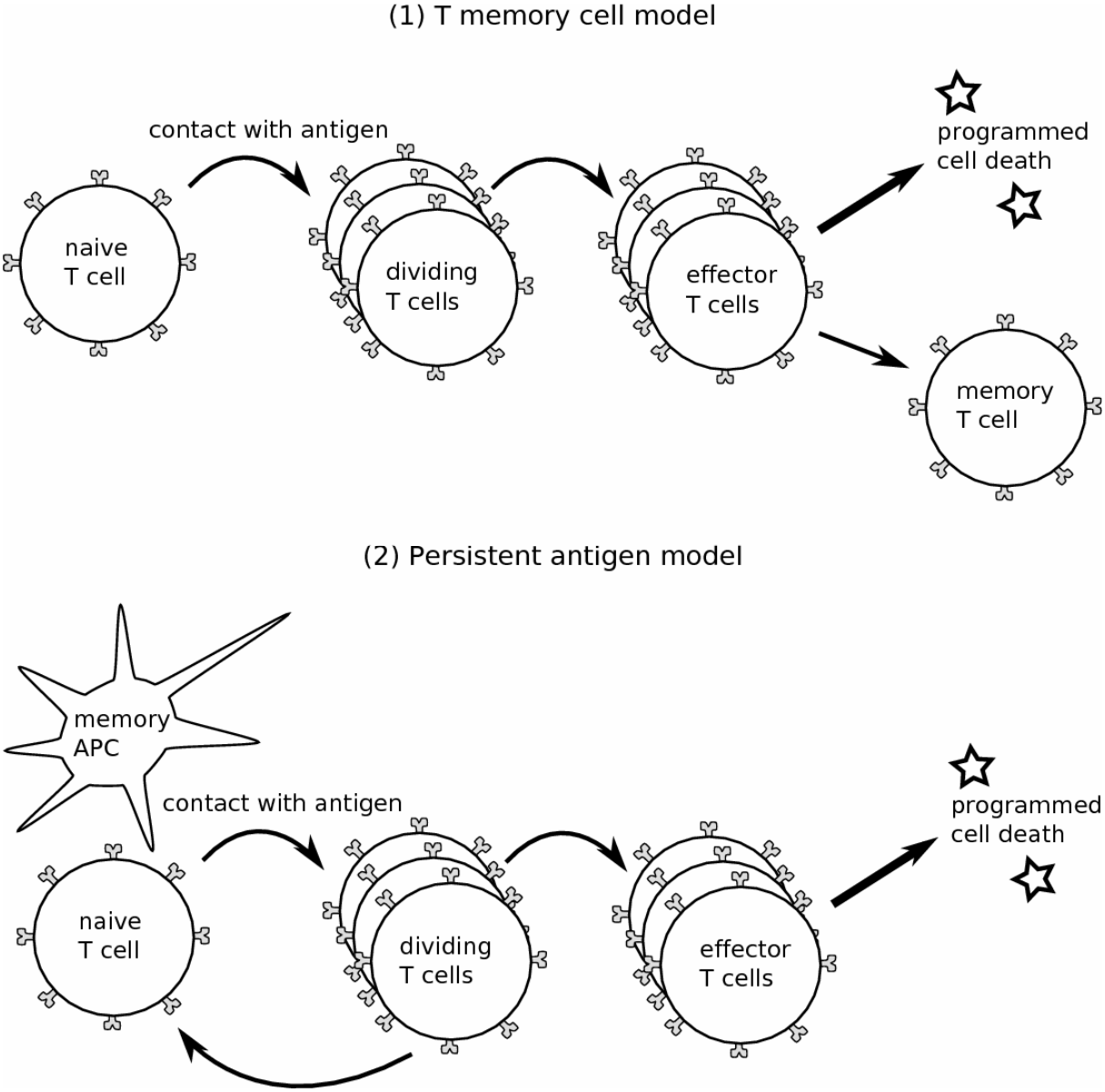

Figure 13: Two hypotheses about T cell immune memory. (1) After contact to antigen, T cells proliferate and become effector T cells that stimulate B cells. Effector T cells are short-lived, but some of them become long-lived memory cells. (2) There is no memory T cell. Processed antigen is kept in long-lived antigen presenting cells and continues to stimulate naive T cells, which can revert to the naive state after proliferating. This keeps a constant supply of specific T cells available.

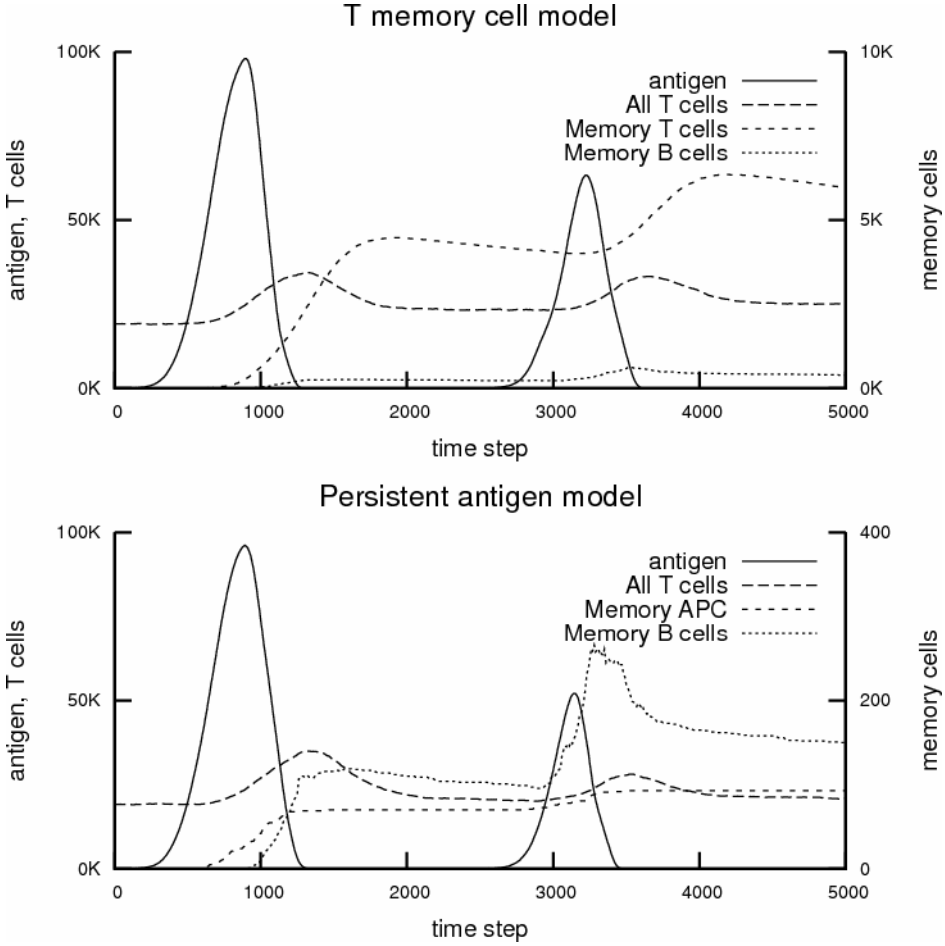

Figure 14: Simulation results for the two memory models on a 40x40 lattice.

Tables

Symbol	*Meaning*
$\Delta t > 0$	Temporal resolution
$\Delta s > 0$	Spatial resolution
$C_{\max} > 0$	Capacity of one site of the lattice
$S_T > 0$	Size of agents with type T
$D_T \in [0,1]$	Motility (diffusion) coefficient of T
$I_{T,T'} \in [0,1]$	Rate of interactions between T and T'
$P_T \in [0,1]$	Proliferation rate of T
$C_T(s,t)$	Number of agents with type T on lattice site s at time t. C_T without parameters refers to the current site at the current time.
COIN(p)	Coin throw with success probability p

Table 1: Global parameters and functions of the simulation framework.

Parameter	*CS algorithms*	*Modified algorithms*
Δt	1	1
Δs	1	1
$C_{\max}$	32	32
$S_A = S_{A^*} = S_B = S_{B^*}$	1	1
$D_A = D_{A^*}$	0.032	0.032
$D_B = D_{B^*}$	0.32	0.32
$I_{A,A^*} = I_{A^*,A}$	0.0007	0.182
$I_{A^*,B} = I_{A,B^*}$	0.0006125	0.0104
$I_{A,B} = I_{A^*,B^*}$	0.001255	0.0208

Table 2: Parameters for the simulations of the bistable chemical system. The values given in are adopted from Malevanets and Kapral (1996), and can be used directly with our hybrid algorithms. The corresponding values for the CS algorithms had to be determined by manual tuning (left column).

Process	*Rate*
Antigen proliferates	0.02
APC phagocytes antigen	0.0005
B cell phagocytes antigen	$\alpha(receptor, epitope)$
APC or B cell processes ingested antigen	$\alpha(mhc, peptide)$
APC or presents antigen to T cell	$0.8\,\alpha(peptide, receptor)$
Costimulation between B and T cell	$3\,\alpha(peptide, receptor)$
Stimulated B cell proliferates	0.05
Stimulated T cell proliferates	0.025
B cell or APC lose presented antigen	0.01
B cell secretes antibody	16
Antigen binds to antibody	$0.5\,\alpha(antibody, epitope)$
APC phagocytes antigen-antibody complex	1

Table 3: Actions and interactions and their default rates as implemented in our simulation of the immune response. Stimulated B and T cells go through 6 division cycles each, and antigen never stops to proliferate. The values are set to mimic the relations between the corresponding processes in the real immune system, but are not directly interpretable as realistic quantities.

Agent type	*Half-life*	*Diffusion coefficient*
Antigen	∞	0.1
Labelled antigen	1 000	0.1
Antibody	690	0.5
T cell	200	0.02
Effector T cell	50	0.02
B cell	200	0.02
Plasma B cell	100	0.02
Memory cell	10 000	0.02
APC	∞	0.01

Table 4: Agent half-lifes (in time steps) and diffusion coefficients used in our simulation of the immune response.