---

**Article:**
Tamrakar, S., Richmond, P. and D'Souza, R.M. (2016) PI-FLAME: A parallel immune system simulator using the FLAME graphic processing unit environment. Simulation, 93 (1). pp. 69-84. ISSN: 0037-5497

https://doi.org/10.1177/0037549716673724

---

Journal Title
XX(X):1–15

DOI: 10.1177/ToBeAssigned
www.sagepub.com/

# PI-FLAME : A Parallel Immune System Simulator using the FLAME GPU environment

**Shailesh Tamrakar[1], Paul Richmond[2] and Roshan M. D'Souza[1]**

**Abstract**

Agent-based models (ABMs) are increasingly being used to study population dynamics in complex systems such as the human immune system. Previously, Folcik et al. developed a Basic Immune Simulator (BIS) and implemented it using the RePast ABM simulation framework. However, frameworks such as RePast are designed to execute serially on CPUs and therefore cannot efficiently handle large model sizes. In this paper, we report on our implementation of the BIS using FLAME-GPU, a parallel computing ABM simulator designed to execute on Graphics Processing Units (GPUs). To benchmark our implementation, we simulate the response of the immune system to a viral infection of generic tissue cells. We compared our results with those obtained form the original RePast implementation for statistical accuracy. We observe that our implementation has $13\times$ performance advantage over the original RePast implementation.

**Keywords**

Agent Based Models; Innate Immune System; Adaptive Immune System; FLAME GPU;

## 1. Introduction

The immune system consists of biological structures and processes that protect an individual against various pathogens. In order to function effectively, the immune system should be capable of detecting and eliminating various pathogens such as virus, bacteria, and parasitic multi-cellular organisms while at the same time not harming the individual's healthy tissue. The immune system in higher organisms such as humans is the result of evolution for over 300 million years. It consists of a complex system of specialized cells, signalling chemicals, and pathogen destroying antibodies. It provides a multi-layered protection against pathogens with each layer providing increasing specificity.

The first line of defence against pathogens are barriers such as the skin. If this physical barrier is breached, there are two main classes of immune responses. The Innate immune system is the first line of defence. It provides a non-specific Alberts et al. (2002) immediate maximal response. The response is cell-mediated Matzinger (2002a) with humoral components. However, there is no memory associated with the innate immune response. If the innate response fails to clear the pathogen, vertebrates in particular posses a second layer of immune system called the adaptive immune system that is activated by the innate response. The adaptive immune system is responsible for exponential proliferation of antigen-specific cells. Unlike innate immune cells, the anitgen-specific adaptive immune cells develop an immunological memory so that host response is much faster when the same pathogen infects the functional cells of the body at a later date Pancer and Cooper (2006).

### *Mathematical Modelling of the Immune system*

The immune system is a complex system with a vast network of interactions between different cells and chemical signals. The shear complexity of the dynamics associated with these interactions makes experimental studies very challenging. While *in vitro* experimentation with a few cells might show some local interactions, separation of these cells from their natural environment causes non-physiological behavior. On the other hand *in vivo* studies may enable study of the overall behavior, but the local interactions are difficult to resolve. The main challenge is to connect individual level interactions and connect them to large-scale phenomena. It is here that mathematical modelling can provide some insight.

Among the first mathematical tools to model immune system were through a system of ordinary differential equations (ODEs) Merrill (1981); Varela and Stewart (1990); Fouchet and Regoes (2008). This system of ODEs described the temporal dynamics of various immune/pathogen cells as well as the bio-chemicals (cytokines, chemokines, cytotoxins etc.). The main advantage of ODEs is the fact that they allow analysis of the dynamics of the system at a level of abstraction similar to how experimentalists conduct their investigation, i.e., monitoring the dynamics of

[1]Department of Mechanical Engineering, UW-Milwaukee
3200 N Cramer St., Milwaukee, Wisconsin,53211, USA
[2]Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello, Sheffield, UK

**Corresponding author:**
Roshan M. D'Souza
Dept. of Mechanical Engineering
UW - Milwaukee
Email: dsouza@uwm.edu

concentrations and density of various chemicals and cells respectively. Moreover, fairly sophisticated techniques exist to solve and analyze these ODEs. The disadvantage is the lack of representation of spatial dynamics. Furthermore, local interactions between cells is not accounted for properly. Systems of Partial differential equation (PDEs) resolve the spatial dynamics Antia et al. (2003); Onsum and Rao (2007). The downside being that these systems are difficult to solve as well as to analyse. Most of these PDEs take the form of reaction-diffusion-decay type PDEs. The origins of these methods can be traced to the work by Alan TuringTuring (1952). As in the case of ODEs, PDEs can only resolve physiological-level behavior, albeit with space. Local interactions between cells cannot be captured. To capture the stochastic nature of some of the processes in the immune dynamics, some researchers have used stochastic ODEs and PDEs Figge (2009); Figueredo et al. (2014); Yuan and Allen (2011). Once again, only physiological-level stochasticity can be captured.

All the methods described above operate under continuum/bulk assumptions, i.e., cell populations are large and therefore the effective measure of population sizes is concentration which are represented as real numbers. In quite a few situations, this assumption may not be true, especially in case of modelling an immune response in the early stages of an infection. Some researchers have used the Gillespie algorithm Gillespie (1977), a variation of the Monte Carlo, which has traditionally been used to model chemical kinetics in bio-chemical networks. The Gillespie algorithm is applicable in cases where the number of cells are finite and the interaction/reaction between these cells is stochastic Figueredo et al. (2014). Spatial dynamics have been incorporated in compartmentalized versions of this algorithm Stundzia and Lumsden (1996).

A shortcoming of the above methods is the difficulty in representing memory which is essential for modelling adaptive immune response. Models based on cellular automata (CA) Seiden and Celada (1992); Mallet and De Pillis (2006); Puzone et al. (2002); Castiglione and Bernaschi (2004) and agent-based modelling (ABMs)Chiacchio et al. (2014); Noble (2002); Van Dyke Parunak et al. (1998); Folcik et al. (2007) address this shortcoming. Both CA and ABMs are designated as "individual-based models" in ecology because of the fact that both models are bottom-up. The global/system level behaviors emerge from the local interactions of discrete individual entities comprising the system Clarke (2014). Local interactions are modelled in terms of rules which are typically derived from experimental observations. The CA models are used when space is represented in terms of a grid. The individuals in this case are grid nodes. ABMs on the other hand are not restricted to a grid architecture. In ABMs the individual/agent is an autonomous generic finite state machine that is mobile and can locally interact with other agents and/or environment and effect state transitions. The term "local" is not restricted to spatial locality but can include networks. Furthermore, space can be represented continuously or discretely as in CAs. Therefore, ABMs are preferred for complex dynamic systems that include mobile interacting entities. Due to the emergent nature of CAs and ABMs, model sizes (agent populations) must reflect numbers in the representative volume element in the real world Bonabeau (2002). Moreover, due to the stochastic nature of the models, several model runs have to be computed to generate sufficient data for statistical analysis. Traditional serial computing on central processing units (CPUs) cannot handle this computational load. Increasingly, researchers have developed parallel computing techniques to address computational complexityBernaschi et al. (1998); Collier and North (2013); Massaioli et al. (2005).

## *Parallel Execution of ABMs on Graphics Processing Units (GPUs)*

Beginning in 2007, a new off-the-shelf commodity parallel computing platform, the graphics processing unit (GPU), has emerged as an attractive hardware platform for parallel computing on desktop computers. GPUs were initially developed for accelerating 3D graphics rendering. Computational scientists used this rendering functionality to accelerate scientific computing Owens et al. (2007). Subsequently, GPU vendors developed APIs to directly access the parallel hardware without invoking the graphics pipeline. Several recent efforts have investigated parallel acceleration of agent-based models on GPUs Richmond et al. (2009); DSouza et al. (2007, 2009).

GPUs follow the data-parallel model where the same set of instructions act on individual elements of a data

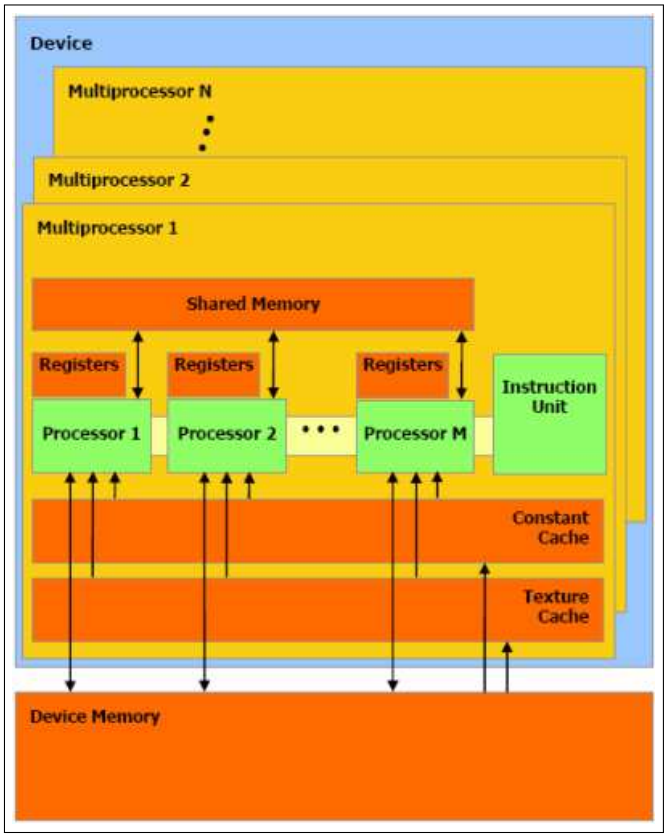



**Figure 1.** Execution hardware and memory hierarchy on GPUs. Each serial processor (SP) executes one thread and therefore processes one data element. Threads are grouped into warps to be executed in lock-step. Multiple thread warps are scheduled on a time sharing basis to execute on the multi-processor. The order of execution of thread warps on the SMs is random.

set. Furthermore, there are no interdependencies between different data set elements at a given processing step. This greatly simplifies the memory access and control. Therefore, most of the GPU transistors are allocated to computation as opposed to control. The simplified memory architecture also lends itself to much higher bandwidth as compared to CPUs. At the hardware level, several serial processors (SPs) are organized into a streaming multiprocessor (SMs) (Figure 1). The SPs in a SM share on-chip user-controlled cache called shared memory, special function units, instruction dispatch units, and warp schedulers. The texture cache and constant cache are read only automatic cache for read-only texture data (used in graphics rendering), and for storing system wide constants respectively. The lowest execution unit is a thread and is executed on a SP. Threads are organized into thread-warps which are executed in lock-step on SPs in a single SM. In software, threads are organized into thread blocks (TBs). Threads in a TB can communicate through shared memory and can be synchronized. All threads in a TB, split into appropriate number of warps, are executed on the same SM. TBs are further arranged into a grid. Threads across TBs can only communicate through off-chip random access memory (RAM). Threads across all TBs are automatically synchronized at the end of the execution of the parallel program called a *kernel*. As opposed to current generation multi-core CPUs, GPUs can have thousands of cores (SPs) and operate efficiently only when executing thousands of threads.

### *Basic Immune Simulator (BIS)*

The Basic Immune Simulator Folcik et al. (2007) employs an ABM to study the interaction between different cells in the immune system. In this particular model, immune cells are modelled as agents which produce chemical signals based on what they detect from their proximal location and interact with other cells in a computer simulated environment. The previous version of Basic Immune Simulator was implemented using an open source software library called Recursive Porous Agent Simulation Toolkit (RePast) North et al. (2006). It used serial JAVA code to simulate the interaction between immune cells. Consequently, this implementation cannot efficiently handle models with population sizes exceeding 20000.

### *ABM Simulation on GPU*

GPUs are well suited for executing ABMs efficiently. The state attributes of the agents are maintained in a data structure. The state transition functions are mapped to *kernels* that execute in parallel. In this paper, we have employed a highly parallel framework designed especially for agent based model called Flexible Large-scale Agent Modeling Environment on the GPUs (FLAME GPU) Richmond et al. (2010) to implement a GPU-based parallel version of the BIS. As opposed to RePast, FLAME GPU leverages the computational power of GPUs and greatly increases the model sizes that can be handled. We have implemented a simplified viral infection that infects a generic tissue and observed the response of immune cells upon infection of generic tissue by a virus. We have compared execution time for this model on RePast and FLAME GPU. Additionally, we carried out tests to verify the statistical accuracy of immune simulator in FLAME GPU when the same initial conditions as in the RePast implementation, for the immune win condition were used.

We begin by briefly discussing about the FLAME GPU framework and its features. Next, we describe the simulation domain and different agent types participating in simulation along with their algorithms. Finally, we conclude our paper with discussion of results i.e. statistical comparison and a performance benchmark between FLAME GPU and RePast.

## The FLAME GPU Agent Simulation Framework

Unlike the traditional frameworks for ABMs such as Repast which are based on algorithms that are executed serially on CPUs, FLAME GPU utilizes the data parallelism of GPUs to improve computational performance. A key mechanism to leveraging this performance is abstracting the underlying architecture through use of a high level modelling syntax and an automated technique for code generation Richmond and Romano (2011) which avoids users having to understand architectural complexities, the mapping of agents to GPU memory, or even the ideas of parallelism. The FLAME GPU technique results in highly efficient simulations allowing model scale to be increased beyond sizes which can be computed in reasonable time in CPU targeted frameworks such as RePast. Additionally, visualization is relatively inexpensive to achieve since agent data is located in GPU memory where it can be rendered directly without any additional computational overhead.

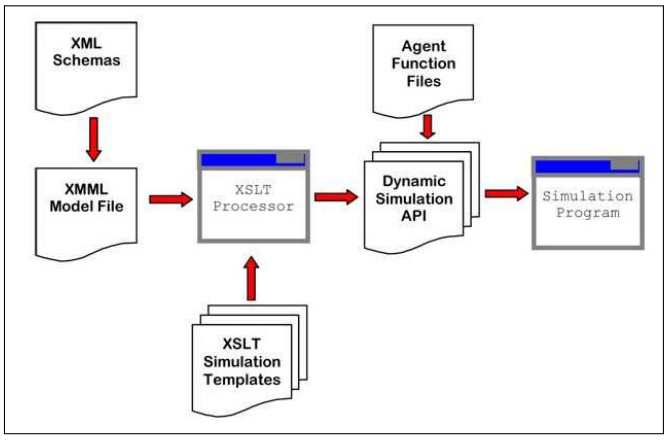



**Figure 2.** FLAME GPU process flow: XSLT processor generates compilable simulation code from XMML templates and a model file.

FLAME GPU is based on a concept called X-machine Eilenberg and Tilson (1974); Holcombe (1988) which is an extension of Finite State Machine (FSM) with internal memory. The X-machine consists of X-machine agents and transition functions. X-machine agents are the state machines which communicate with other agents by iterating through message lists from a message board stored in global memory of the GPU. Transition functions, on the other hand, handle the state changes of agents based on their state transition rules. After the transition function causes the state change,

agents update their internal memory which is either used as an input to subsequent transition functions or as an output which is passed as messages for other agents to read.

Figure 2 illustrates the process flow for the generation of compilable FLAME GPU C simulation code Richmond et al. (2010). XMML model file along with extensible style-sheet language transformation (XSLT) templates is processed through XSLT processor to generate compilable simulation code along with agent data structures, agent and message API functions. Two XMML schemas are used to validate the syntax the of the XMML model files. First, a XMML base schema which verifies the syntax of base XMML model file and the second, a GPUXMML schema which adds any additional GPU specific model parameters. The simulation can be run in either visualization or console mode. The visualization mode allows a user to view the simulation in real time while console mode allows a user to generate XML output for predefined number of iterations.

## *Agent model specification*

The variables, functions and layers associated with X-machine agent are defined within XMML model file. An example of specification of one cell type i.e. generic tissue (Parenchymal cell) is demonstrated in Figure 3. The memory XML element contains the variables and its type associated with Parenchymal cell. Figure 4 shows the initial value of variables for Parenchymal cells.

The function XML element includes an agent function definition (Figure 5) which handles the state change of agents, input or output of messages. Agent function file incorporates script for the rules which are logical statements that lead to behavior of agents based on information gathered from their local environment. For example, agent function `state_pcells` determines whether Parenchymal cell is healthy or infected by a virus. The message names (`agent_bcells` and `agent_pcells`) defined within `inputs` and `outputs` create message functions to access message variables from message board for the purpose of reading and writing respectively. Layer defines the order in which the agent functions are executed.

## *Agent communication*

Agents communicate by iterating the input message list through optimized message iteration function.The message type is defined within XMML model file which can be either non partitioned or spatially partitioned message communication. The method we used in our implementation for agent communication is the spatially partitioned technique. This method is based on interaction of particle systems Green (2008) which divides the simulation domain into uniform sized grids. Agent messages are binned based on the grid index and are sorted using fast radix sort algorithm Satish et al. (2009).

A scatter kernel is used to find the first and last index (Figure 6) of the agent messages in the sorted list which is later used to iterate through messages in all 27 neighboring partitions (for 3-D environment). The agent

```
<xagents>
  <gpu:xagent>
    <name>Parenchymal_cell</name>
    <memory>
      <gpu:variable>
     <type>int</type>
       <name>id</name>
    </gpu:variable>
      <gpu:variable>
       <type>float</type>
       <name>x</name>
    </gpu:variable>
      <gpu:variable>
       <type>float</type>
       <name>y</name>
    </gpu:variable>
      <gpu:variable>
       <type>float</type>
       <name>z</name>
    </gpu:variable>
      <gpu:variable>
       <type>int</type>
       <name>state</name>
    </gpu:variable>
      <gpu:variable>
       <type>int</type>
       <name>bearAb</name>
    </gpu:variable>
      <gpu:variable>
       <type>float</type>
       <name>signal_PK1</name>
    </gpu:variable>
      <gpu:variable>
       <type>float</type>
       <name>signal_virus</name>
    </gpu:variable>
      <gpu:variable>
       <type>float</type>
       <name>signal_apoptic</name>
    </gpu:variable>
      <gpu:variable>
       <type>float</type>
       <name>signal_necrotic</name>
    </gpu:variable>
      <gpu:variable>
       <type>int</type>
       <name>stressedTime</name>
    </gpu:variable>
    </memory>
    <functions>...</functions>
    <states>
      <gpu:state>
        <name>default_pcells</name>
      </gpu:state>
      <initialState>default_pcells</
          initialState>
    </states>
    <gpu:type>continuous</gpu:type>
    <gpu:bufferSize>5457</
        gpu:bufferSize>
  </gpu:xagent>
 </xagents>
```

**Figure 3.** XMML model file definition for Parenchymal cells: variables (position coordinates, chemical signals, stressed time), agent function definitions, agent population, state variables and layers are defined within XMML model file.

message within a predefined radius of influence is considered for communication.

```
<xagent>
    <name>Parenchymal_cell</name>
    <id>0</id>
    <x>-1</x>
    <y>-1</y>
    <z>0</z>
    <state>4</state>
    <bearAb>0</bearAb>
    <signal_PK1>0</signal_PK1>
    <signal_virus>0</signal_virus>
    <signal_apoptic>0</signal_apoptic>
    <signal_necrotic>0</signal_necrotic
        >
    <stressedTime>0</stressedTime>
</xagent>
```

**Figure 4.** State initialization values of variables for one Parenchymal Cell.

```
<xagents>
  <gpu:xagent>
    <name>Parenchymal_cell</name>
    <memory>...</memory>
    <functions>
      <gpu:function>
        <name>state_pcells</name>
        <currentState>default_pcells</
            currentState>
        <nextState>default_pcells</
            nextState>
        <inputs>
          <gpu:input>
            <messageName>agent_bcells</
                messageName>
          </gpu:input>
        </inputs>
        <outputs>
          <gpu:output>
            <messageName>agent_pcells</
                messageName>
            <gpu:type>single_message</
                gpu:type>
          </gpu:output>
        </outputs>
        <gpu:reallocate>false</
            gpu:reallocate>
        <gpu:RNG>false</gpu:RNG>
      </gpu:function>
    </functions>
  </gpu:xagent>
</xagents>
```

**Figure 5.** Agent state transition function definitions for Parenchymal Cells which determines whether they are healthy, challenged, stressed or dead based on influence of other agents.

## Simulation Domain

The simulation domain implemented in FLAME GPU is adapted from previous version of Basic Immune Simulator (BIS) to preserve the qualitative nature of the simulation. The simulation domain is divided into three zones namely: Zone 1, Zone 2, and Zone 3 Folcik et al. (2007). Figure 7 shows the virtual representation of three different zones. Zone 1 is a representation of site for viral infection of generic parenchymal cells. The healthy parenchymal cells are represented by yellow spheres and the infected parenchymal cells are represented by green spheres as shown in Figure 7a. These cells are arranged in a rectangular grid. Zone 1 is also a residence for tissue surveilling innate immune cells called Dendritic cells (DCs). Zone 2 (Figure 7b) is a residence for adaptive immune cells (B-cell agents, T-cell agents and Cytotoxic T Lymphocyte agents) and represents lymph nodes or spleen. Zone 3 (Figure 7c) is a representation of blood circulation system which acts as a media to supply adaptive immune cells to site of viral infection i.e. Zone 1.

Adaptive immune cells move randomly in Zone 3 for some period of time before they move to Zone 1. Zone 3 also contains granulocyte agents which are the type of white blood cells for fighting the infection. Each zone contains portal agents which represent lymphatic and blood vessels. These portal agents are responsible for transferring immune cells from one zone to the other.

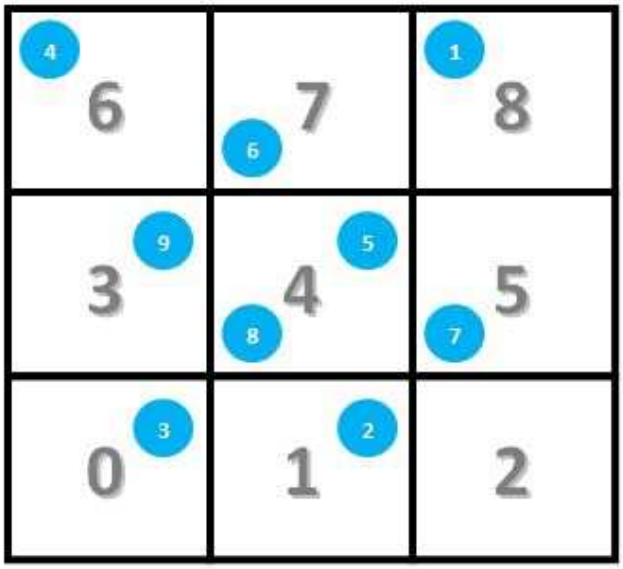



**Figure 6.** An example of 2D spatially partitioned technique. The agent with ID 8 located in grid 4 iterates through all the agent messages present in 9 grids (0-2, 3-5, 6-8). Agent messages located within fixed radius are considered for interaction.

## Agent types

The agents that take part in immune system simulation are generic functional tissue cells (Parenchymal cells), innate immune cells (Macrophages, Natural Killers, Dendritic cells and Granulocytes) and adaptive immune cells (B-cells, T-cells and CTLs). Innate immune cells originate in bone marrow Shortman and Naik (2007). These immune cells are the first ones to respond to distressed signals Matzinger (2002b) produced by infected Parenchymal cells. They enter Zone 1 via portals present in this zone. Natural Killers kill infected Parenchymal cells. Macrophages are responsible for

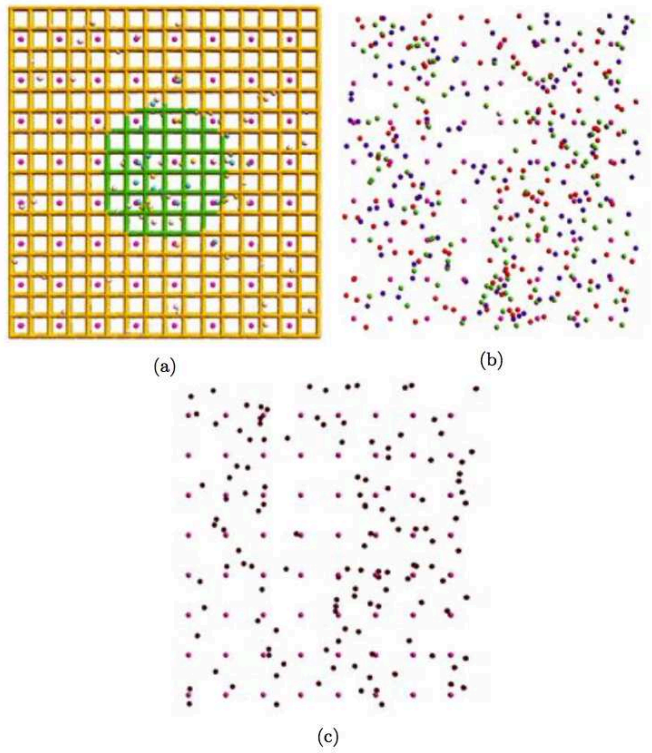



**Figure 7.** Virtual representation of three different zones. (a) Zone 1: Site for infection of Parenchymal Cells by a virus. (b) Zone 2: Representation of lymph node or spleen. (c) Zone 3: Representation of blood circulation system

killing infected Parenchymal cells as well as scavenging of dead Parenchymal cells. The process of scavenging of dead Parenchymal cells sets up a necessary condition for regeneration of healthy Parenchymal cells. Dendritic cells gather antigens from infected Parenchymal cells and travel to Zone 2 i.e lymph node through portals and present antigens to adaptive immune cells that match its specificity. Activation, proliferation, as well as transition of adaptive immune cells (B-cells, T-cells, CTLs) to memory cells take place in Zone 2 when they come in contact with Dendritic cells or with each other. The activated adaptive immune cells first move to Zone 3 where they randomly move for some time and finally travel to Zone 1. T-cells represent helper T-cells which manage activities of macrophages and B cells by releasing cytokines Janeway et al. (2001). CTLs are killer cells responsible for killing infected Parenchymal cells that match their antigen specificity. B cells, on the other hand, produce antibodies against the antigens making them primary targets for killer immune cells. Agent motion is influenced by chemical signal detected in the proximal environment. Immune cells would tend to move randomly if there isn't any chemical signal around them. If they detect any chemical signal in their vicinity, they will follow the signal with highest concentration Beilhack and Rockson (2003).The rules for the state change of different types of cells participating in simulation is described in the sub-section below:

## *Parenchymal cell agents*

**Algorithm 1** State change : Parenchymal Cells

```
1:  procedure STATE_PARENCHYMAL_CELLS()
2:      if (state==HEALTHY & virus>Ab1+Ab2) then
3:          state=CHALLENGED
4:      end if
5:      if (state==CHALLENGED) then
6:          PK1=outputSignal
7:          if      (Ab1+Ab2>virus & (bearAb==FALSE) &
    (Ab2≥0) then
8:              state=TARGETED
9:          end if
10:         if (NK||Ts||MΦs ) then
11:             state=APOPTIC
12:         end if
13:     end if
14:     if (C'>0) & ((Ab1-(virus+Ab2)>Ab_Lysis) then
15:         state=NECROTIC
16:     end if
17:     if (state==STRESSTED) then
18:         PK1=outputSignal
19:         if (life>stressedTime+DURATION_STRESSED)
    then
20:             state=HEALTHY
21:         end if
22:     end if
```

Parenchymal cells are the generic functional tissue cells that remain stationary throughout the course of simulation. These agents are initialized in a HEALTHY state during the start of the simulation.The viral infection starts at the center of Zone 1 which gradually spreads out infecting all neighboring Parenchymal cells. The infected Parenchymal cells release a distressed signal, Parenchymal-kine 1 (PK1) in the form of heat shock proteins Srivastava (2002), uric acid Shi et al. (2003) or chemerin Wittamer et al. (2003). The infected PCs then make a state transition to one of three states: STRESSED, CHALLENGED or TARGETED. The challenged PCs are target for Natural Killers, pro-inflammatory T-cells or CTLs and are killed upon contact. They may undergo lysis in presence of complement products (C') and antibody Ab1 Guo and Ward (2005). The challenged PCs make a transition to state TARGETED when they are bounded by antibody produced by B cells making them susceptible for killing by pro-inflammatory Macrophage agents Casadevall and Pirofski (2003). This causes neighboring parenchymal cells to become stressed as a result of reactive oxygen species released by pro-inflammatory macrophages Ricevuti (1997). The granulocytes also act upon the stressed PCs and kill them by a release of lethal degranulation product 1 (G1) Ricevuti (1997); Segal (2005). The dead Parenchymal cells are scavenged by macrophages facilitating necessary conditions for regeneration of healthy PCs Huynh et al. (2002).

## *Dendritic cell agents*

The DCs begin in an INACTIVE state in zone 1 and are of two types: pro-inflammatory (DC1) and anti-inflammatory (DC2). The DCs in INACTIVE state can transition to

two possible states: ACTIVATED or AG-PRIMED (antigen primed) depending upon the type of signal it detects.

**Algorithm 2** State change : Dendritic Cells - Zone 1

```
1:  procedure STATE_DENDRITIC_CELLS_ZONE1()
2:      if (life < LIFE_DC_ZONE1) then
3:          if (state==INACTIVE) then
4:              if (PK1>0) then
5:                  state=ACTIVATED
6:                  if (MK1≫MK2 & type==DC1) then
7:                      type=DC2
8:                  end if
9:              end if
10:             if (virus>200) then
11:                 state = AG-PRIMED
12:                 if (Ab1>200 & Ab2>200 & type==DC1)
    then
13:                     type=DC2
14:                 end if
15:                 if (CK1>200) & type==DC2 then
16:                     type=DC1
17:                 end if
18:             end if
19:         end if
20:         if (state==ACTIVATED) then
21:             if (virus || CHALLENGED PC contact) then
22:                 state=AG-PRIMED
23:             end if
24:         end if
25:     end if
26:     if (life>LIFE_DC_ZONE1) then
27:         state=APOPTIC
28:     end if
```

The dendritic cells of type DC1 or DC2 move randomly until they detect shock signal (PK1) produced by infected parenchymal cells. The detection of PK1 in its immediate environment causes DCs to change to ACTIVATED state Gallucci et al. (1999). The activated DCs produce a chemical signal MK1 (Mono-kine 1) or MK2 (Mono-kine 2) depending on their type. For example, INACTIVE DC1 changes to ACTIVATED DC1 and INACTIVE DC2 changes to ACTIVATED DC2 upon detection of PK1. However, if signal MK2 (Mono-kine 2) is greater than signal MK1 (Mono-kine 1), it converts to ACTIVATED DC2 Koch et al. (1996). The DCs attain antigen-primed (AG-PRIMED) state under three conditions. First, the detection of virus signal in proximal location of DC1 or DC2 causes them to convert to AG-PRIMED DC1 or AG-PRIMED DC2 Zuniga et al. (2004). Second, the presence of virus along with antibody changes INACTIVE DC1 to AG-PRIMED DC2 Anderson et al. (2004). INACTIVE DC2 is transitioned to AG-PRIMED DC1 in presence of virus and signal CK1 (cyto-kine 1) Vieira et al. (2000). Third, the DCs in ACTIVATED state when make contact with virus infected PCs change to AG-PRIMED state respective of their type Hart (1997). The AG-PRIMED DCs move to Zone 2 via portals present in Zone 1 to present the antigen to adaptive immune cells (B-cells, T-cells and CTLs) Hart (1997).The DCs undergo apoptosis if they fail to detect any infected PCs or stress signal within their lifetime Hou and Van Parijs (2004).

**Algorithm 3** State change : Dendritic Cells - Zone 2

```
1:  procedure STATE_DENDRITIC_CELLS_ZONE2()
2:      if (life < LIFE_DC_ZONE1 & zone==2) then
3:          if (state==AG-PRIMED) then
4:              if (type==DC1 || type==DC2) then
5:                  if (timerMK<DURATION_MK_ZONE2)
    then
6:                      MK1 or MK2 = outputSignal
7:                      timerMK1 or timerMK2 += 1
8:                  end if
9:                  if (Ag-matched B1 or B2) then
10:                     life=0
11:                 end if
12:                 if (Ag-matched T1 or T2) then
13:                     MK1 or MK2 = 0
14:                     NumTsContact += 1
15:                 end if
16:             end if
17:         end if
18:     end if
19:     if (numTsContact≥12) then
20:         state=APOPTIC
21:     end if
22:     if (life>LIFE_DC_ZONE1) then
23:         state=APOPTIC
24:     end if
```

After the DCs move to Zone 2, they move randomly for some time and become stationary. They continue to produce MK1 or MK2 in Zone 2 as well. At this moment, DCs wait for antigen matched adaptive immune cells (B-cells, T-cells and CTLs) to make contact with them. The contact with DCs affects the state of adaptive immune cells as well as the state of themselves. For example, contact with antigen matched B cells extends the life of DCs Miga et al. (2001) and contact with antigen matched T-cells resets the production of MK1 or MK2. The DCs undergo apoptosis if they reach allocated life time in Zone 2 or exceed threshold for T- cells contact Ingulli et al. (1997); Kriehuber et al. (2005).

### *Macrophage agents*

The macrophages (MΦs) enter Zone 1 in naive state (MΦ0) when portals present in Zone 1 sense PK1 emitted by infected PCs. The state change of MΦ is determined by type of signals it senses from its local environment.

The presence of PK1, CK1, C' (complement products) and necrotic debris Guo and Ward (2005) causes MΦs to transition to ACTIVATED MΦ1s (pro-inflammatory) whereas sensing of apoptic signal and antigen-antibody (Ag-Ab) complexes Casadevall and Pirofski (2003) cause MΦs to change to ACTIVATED MΦ2s (anti-inflammatory). Both MΦ1 and MΦ2 in activated state have the ability of scavenging dead PCs which provides necessary condition for regeneration of healthy PCs. MΦ1 also has the ability to kill infected PCs. MΦ1 and MΦ2 in activated state produce the signals MK1 and MK2 respectively. The killing of infected PCs and scavenging of dead PCs causes MΦs to attain

**Algorithm 4** State change : Macrophages

```
1:  procedure STATE_MACROPHAGES()
2:      if (life < LIFE_MΦ_ZONE1) then
3:          if (state==MΦ0) then
4:              if (PK1>0 || CK1>0 || C'>0 || Nec. debris>0
    ) then
5:                  state=ACTIVATED MΦ1
6:              end if
7:              if Apop.Signal>0 || Ab-Ag complexes then
8:                  state=ACTIVATED MΦ2
9:              end if
10:         end if
11:         if (state==ACTIV. MΦ1 || ACTIV. MΦ2) then
12:             if (kill PCs || scavenge PCs) then
13:                 state = AG-PRIMED MΦ1 or MΦ2
14:             end if
15:         end if
16:     end if
17:     if (life>LIFE_MΦ_ZONE1) then
18:         state=APOPTIC
19:     end if
```

antigen-primed (AG-PRIMED) state respective of their type. The MΦs in AG-PRIMED state posses ability from previous state to scavenge dead PCs as well as kill PCs bound by an antibody. If MΦs in this state make a contact with antigen-matched T-cells, they extend the life of MΦs. The MΦs in activated and antigen primed state undergo apoptosis when their time is over.

### *Natural Killer agents*

Natural Killers (NKs) are introduced to Zone 1 when portals in Zone 1 sense PK1 from PCs. They move randomly for some time. When they detects PK1 that is being emitted from infected PCs, they follow PK1 with highest concentration eventually seeking for infected PCs. If signal PK1 is greater than CK1 around the Natural Killer of interest, killing of infected PCs take place upon contact. At this time, release of chemical signal CK1 by NKs also take place Stetson et al. (2003). After the killing of infected PCs and CK1 production, NKs return to the state where they continue to move randomly. They have a limited number of kills and lifetime, after which they undergo apoptosis.

### *B cell agents*

B cells (Bs) reside in Zone 2 in an INACTIVE state waiting for antigen matched DCs to make contact with them Dubois et al. (1997). If they make contact with DC1, B cells of type B1 are produced and type B2 is the result of contact with DCs of type DC2. B cells of type (B1 or B2) get ACTIVATED when they make contact with activated, antigen-matched T1 or T2. Bs in ACTIVATED state may transition into either of two states: GERMINAL or PLASMA.

Germinal cells in activated state produce antibodies and remain in Zone 2 whereas Plasma B cells, on the other hand, travel to Zone 1 and produce antibodies there. Contact with Ts also leads to birth of B cells. ACTIVATED Bs make a transition to MEMORY Bs when they make series of contacts with antigen matched DC1 or DC2, thus, extending

**Algorithm 5** State change : Natural Killers

```
1:  procedure STATE_NATURAL_KILLERS()
2:      if (life < LIFE_NK_ZONE1) then
3:          if (PK1>0) then
4:              followPK1=TRUE
5:          end if
6:          if (PK1>CK2 & followPK1==TRUE) then
7:              killPC = TRUE
8:          end if
9:          if (killPC==TRUE) then
10:             killCount+=1                              ;
    CK1Timer=DURATION_NK_CK1
11:         end if
12:         if (CK1Timer>0) then
13:             CK1=outputSignal ; CK1Timer -= 1
14:         end if
15:         if (CK1Timer==0) then
16:             followPK1 = FALSE ; randomMotion =
    TRUE
17:         end if
18:         if (killCount ≥ NK_KILL_LIMIT) then
19:             state=APOPTIC
20:         end if
21:     end if
22:     if (life > LIFE_NK_ZONE1) then
23:         state=APOPTIC
24:     end if
```

**Algorithm 6** State change : B cells - Zone1

```
1:  procedure STATE_BCELLS_ZONE1()
2:      if (life < LIFE_Bs_ZONE1) then
3:          if (type==B1 or B2 & CK1 or CK2>0) then
4:              if (AbTicker < DURATION_AB_ZONE1)
    then
5:                  Ab1 or Ab2 = outputSignal
6:                  AbTicker += 1
7:              end if
8:          end if
9:      end if
10:     if (life > LIFE_Bs_ZONE1) then
11:         state=APOPTIC
12:     end if
```

life of B cells. Memory Bs are again activated when they make contact with antigen matched Ts in presence of cytokines (MK1, CK1 or MK2, CK2). ACTIVATED Bs undergo apoptosis after their life surpasses allocated lifetime.

Plasma B cells travel to Zone 1 from Zone 3 via portals present in Zone 1. They move randomly in Zone 1 where they produce antibodies in the presence of CK1 or CK2 for predefined period of time. If they don't detect CK1 or CK1 in their proximal location, they stop producing antibodies. They undergo apoptosis after their life is over.

### *T-cell agents*

T-cells begin in Zone 2 i.e. lymph nodes in an INACTIVE state where they move randomly until they find DCs matching their antigen specificity. The contact with DC1 or DC2 results in activation and proliferation of T1 or T2

**Algorithm 7** State change : B cells - Zone2

```
1:  procedure STATE_BCELLS_ZONE2()
2:      if (Ag-matched DC1 or DC2) then
3:          if (type==B1 or B2 & state==ACTIV.) then
4:              DCcontacts + =1 ; life += ADD_LIFE
5:          end if
6:          if (type==B1 or B2 & state==MEM_Bs) then
7:              DCcontacts + =1
8:          end if
9:          if (type==B0) then ; type = B1 or B2
10:         end if
11:     end if
12:     if (Ag-matched T1 or T2) then
13:         if (state==INACTIVE) then
14:             if (type==B0) then
15:                 type = B1 or B2
16:             end if
17:             if (type==B1 or B2) then
18:                 state=ACTIVATED ; flag=rand()
19:                 Birth_Bs = ADD_Bs
20:                 mode=(flag>0.5)?GERMINAL:PLASMA
21:             end if
22:         end if
23:         if (state==MEMORY_Bs) then
24:             state=ACTIVATED ; mode=PLASMA
25:             BIRTH_Bs=ADD_Bs
26:         end if
27:     end if
28:     if (state==ACTIVATED & DCcontacts≥1 then
29:         if (mode==GERMINAL) then
30:             state = MEMORY_Bs ; life += ADD_LIFE
31:         end if
32:     end if
33:     if (state==MEMORY_Bs & DCcontacts≥1) then
34:         life += ADD_LIFE
35:     end if
36:     if (state==ACTIVATED & type==B1 or B2) then
37:         if (AbTimer≤DURATION_AB_ZONE2) then
38:             Ab1 or Ab2=outputSignal ; AbTimer += 1
39:         end if
40:     end if
41:     if (life > LIFE_B_ZONE1) then
42:         state=APOPTIC
43:     end if
```

**Algorithm 8** State change : T-cells - Zone2

```
1:  procedure STATE_TCELLS_ZONE2()
2:      if (state==INACTIVE T0) then
3:          if (Ag-matched DC1 or DC2) then
4:              state=ACTIVATED T1 or T2
5:          end if
6:      end if
7:      if (state==ACTIVATED T1 or T2) then
8:          if timerCK < DURATION_CK_ZONE2 then
9:              CK1 or CK2 = outputSignal
10:             timerCK1 or timerCK2 += 1
11:         end if
12:         if (Ag-matched DC1 or DC2) then
13:             life += ADD_LIFE
14:             Birth_Ts = ADD_Ts
15:         end if
16:         if MK1==0 || CK1 == 0 then
17:             state = MEMORY T1 or T2
18:         end if
19:     end if
20:     if (state==ACTIVATED & life > LIFE_Ts_ZONE2) then
21:         state=APOPTIC
22:     end if
```

**Algorithm 9** State change : T-cells - Zone1

```
1:  procedure STATE_TCELLS_ZONE1()
2:      if (life < LIFE_Ts_ZONE1) then
3:          if (state==ACTIVE_Ts||state==MEMORY_Ts) then
4:              if (Ag-matched MΦ contact) then
5:                  CK1 or CK2 = outputSignal
6:              end if
7:              if (CHALLENGED PC) then
8:                  killPC = TRUE ; killCount += 1
9:              end if
10:             if (killCount≥MAX_T_KILLS) then
11:                 state=APOPTIC
12:             end if
13:         end if
14:     end if
15:     if (life>LIFE_Ts_ZONE1 then
16:         state=APOPTIC
17:     end if
```

respectively Amsen et al. (2004); Hart (1997); Ingulli et al. (1997); Tanaka et al. (2000) and release of signal CK1 or CK2.

Additionally, contact with antigen-matched B cells as well as series of contacts with DCs extend the life of Ts. ACTIVATED Ts make a transition to MEMORY Ts if there is an absence of CK1 or CK2 in the environment. However, contact with antigen-matched DC makes MEMORY T-cells to be ACTIVATED again. T-cells move to Zone 3 where they randomly for some time before traveling to Zone 1. If ACTIVATED Ts fail to make contact with DC within certain period of time, they undergo apoptosis.

T-cells enter Zone 1 in response to detection of PK1 signal by portals present in Zone 1. T-cells of type T1 seek for infected PCs by following PK1 with strongest attraction and kill them upon contact. If T1 or T2 cells make contact with antigen-matched macrophages (MΦs), they start to produce CK1 or CK2 depending upon type of macrophages Anderson and Mosser (2002). T-cells of type T1 undergo apoptosis if they reach beyond the threshold for number of kills Green et al. (2003) or exceed allocated lifetime in Zone 1.

## *CTL agents*

Like other adaptive immune cells, CTLs begin in an INACTIVE state in Zone 2 where they move randomly

waiting to make contact with an antigen-matched DC1s. The contact with DC1s makes them ACTIVATED.

**Algorithm 10** State change : CTLs - Zone2

```
1:  procedure STATE_CTL_ZONE2()
2:      if (Ag-matched DC1) then
3:          if (state==INACTIVE) then
4:              state = ACTIVATED
5:          end if
6:          if (state==ACTIVATED & CK1==0) then
7:              state = MEMORY_CTLs
8:          end if
9:          if (state==MEMORY_CTLs & CK1>0) then
10:             state = ACTIVATED
11:         end if
12:     end if
13:     if (life < LIFE_CTLs_ZONE2) then
14:         if (state==ACTIVATED) then
15:             CK1 = outputSignal
16:             BIRTH_CTLs = ADD_CTLs
17:         end if
18:     end if
19:     if (life > LIFE_CTLs_ZONE2) then
20:         state=APOPTIC
21:     end if
```

In an ACTIVATED state they proliferate and release CK1 signal. ACTIVATED CTLs transition to MEMORY CTLs if they make contact with a DC1 in absence of CK1 signal Badovinac et al. (2005). But, presence of CK1 signal and contact with DC1 extends the life of CTLs. ACTIVATED CTLs then migrate to Zone 3 and eventually to Zone 1 where they kill infected PCs. CTLs enter Zone 1 in an activated state and continue to produce CK1 signal for a finite period of time. Like T-cells (Ts), they seek virus-infected PCs by following PK1 emitted by such PCs and kill them upon contact. The contact with infected PCs also extends the duration of emission of CK1 signal by CTLs. ACTIVATED CTLs are transitioned to MEMORY CTLs in absence of PK1 or CK1 in their immediate environment. The contact with infected PCs causes MEMORY CTLs to be ACTIVATED again. CTLs undergo apoptosis after their life is over.

**Algorithm 11** State change : CTLs - Zone1

```
1:  procedure STATE_CTL_ZONE1()
2:      if (CHALLENGED or TARGETED PC contact) then
3:          if (state==ACTIVATED) then
4:              killPC = TRUE
5:          end if
6:          if (state==MEMORY_CTLs) then
7:              state = ACTIVATED
8:          end if
9:      end if
10:     if (life<LIFE_CTL_ZONE1 & state==ACTIV.) then
11:         if (No CK1 or PK1) then
12:             state = MEMORY_CTLs
13:         end if
14:         if (ticker<DURATION_CK1_ZONE1) then
15:             CK1 = outputSignal ; ticker += 1 end if
16:     end if
17:     if (life > LIFE_CTL_ZONE1) then
18:         state=APOPTIC
19:     end if
```

**Algorithm 12** State change : Granulocytes - Zone 1

```
1:  procedure STATE_GRANULOCYTES_ZONE1()
2:      if (life < LIFE_GRAN_ZONE1) then
3:          G1=outputSignal
4:      end if
5:      if (life > LIFE_GRAN_ZONE1) then
6:          state=APOPTIC
7:      end if
```

### *Granulocyte agents*

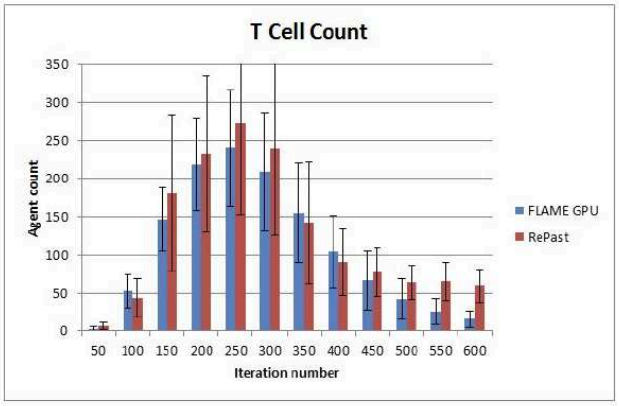
Granulocytes begin in Zone 3 where they are moving randomly. They travel to Zone 1 when portals present in Zone 3 sense presence of complement products Guo and Ward (2005) or MK1. After entering Zone 1, they move randomly until they detect complement products around them and eventually follow signal of highest concentration. They release degranulation product which is capable of killing any stressed PCs Segal (2005). They undergo apoptosis after their time is up.

## Results

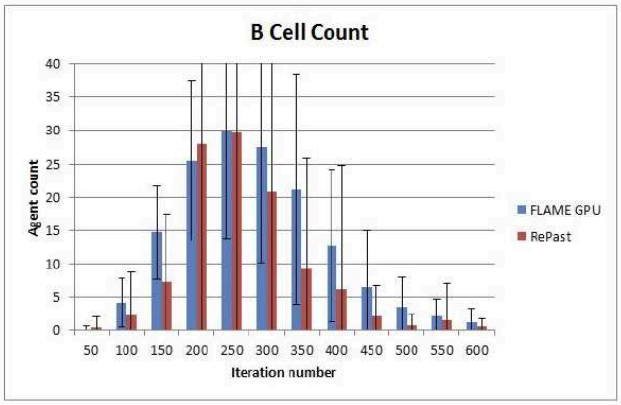
In this section we present results of our comparison with the previous implementation done in the RePast agent-modeling system. The first comparison tests the statistical accuracy of our implementation. There is a possibility that the results obtained from immune simulator implemented in FLAME GPU may deviate from results obtained from their RePast implementation due to the difference in implementation of randomized motion of immune cells. Furthermore, discrepancies in the order at which the state change of immune cells are executed could also lead to difference in results. For this reason, statistical comparison is required to ensure that results from immune simulator in FLAME GPU lie within statistical limits of results obtained from its RePast implementation. The second comparison tests the performance advantage. For statistical accuracy, we chose to set initial conditions, similar to as initialized in RePast, for an immune win situation as shown in Table 1 Folcik et al. (2007). Immune win is a condition in which all the infected parenchymal cells are eliminated by immune cells and sets up necessary conditions for regeneration of healthy parenchymal cells Folcik et al. (2007).

| Parameter | Value |
|---|---|
| Number of PCs | 5457 |
| Viral_Infection_Threshold | 50 |
| Ab_Lysis_Threshold | 100 |
| Duration_Stressed | 25 |
| Number of DCs (Zone 1) | 50 |
| NumDC_ToSend | 3 |
| Life_DC_Zone1 | 50 |
| Life_DC_Zone2 | 100 |
| Duration_MK_Zone1/Zone2 | 25 |
| NumBs_ToSend | 1 |
| Life_B_Zone1 | 25 |
| Life_B_Zone2 | 10 |
| Life_B_Zone3 | 25 |
| Duration_Ab_Zone1 | 150 |
| NumTs_ToSend | 2 |
| Life_T_Zone1 | 20 |
| Life_T_Zone2 | 13 |
| Life_T_Zone3 | 50 |
| Duration_CK_Zone1/Zone2 | 25 |
| T_Max_Kills | 10 |
| NumMΦ _ToSend | 5 |
| Life_MΦ _Zone1 | 50 |
| NumNK_ToSend | 4 |
| NK_Kill_Limit | 15 |
| NumCTL_ToSend | 4 |
| Life_CTL_Zone1/Zone2 | 25 |

**Table 1.** Initial values for immune win condition which are derived from RePast implementation of BIS




(a)




(b)

**Figure 8.** Immune cells count with standard deviation bars for immune win condition. a) T-cell count for Zone 2. b) B cell count for Zone 2.

The initial number of parenchymal cells (PCs) that take part in simulation was set to 5457. Out of 5457 PCs, 192 cells are infected with virus. We ran 50 trials each for both the FLAME GPU and the Repast Implementation. As the simulation progresses, the infection spreads out, thereby infecting all the neighboring PCs. Figure 8 and Figure 9 show the counts of T-cells, B-cells, CTLs and Dendritic cells at Zone 2. The immune cells (innate and adaptive) counts gradually increase as number of PCs infected by virus rises and eventually reaches a peak value when all PCs are infected. The count of immune cells (innate and adaptive) gradually decreases as infected PCs are killed and scavenged by them. As the infected PCs are killed and scavenged, new healthy PCs are regenerated replacing the infected ones.

The results obtained from the simulation were used to measure accuracy of Basic Immune Simulator implemented in FLAME GPU with the previous RePast implementation utilizing available statistical tools such as mean value and standard deviation. We compared T-cell, B-cell, Dendritic cell, and CTL counts in Zone 2. The standard deviation bars in Figures 8,9 indicate that immune cells counts for our FLAME GPU implementation lie within the range of Basic Immune Simulator implemented in RePast. This verifies the statistical accuracy of our implementation in FLAME GPU.

| Initial Agent count | Initial DC count | Speed Up |
|---|---|---|
| 8000 | 500 | 3.43 |
| 10000 | 1000 | 4.03 |
| 12000 | 1500 | 4.55 |
| 14000 | 2000 | 5.04 |
| 16000 | 2500 | 6.35 |
| 18000 | 3000 | 8.1 |
| 20000 | 3500 | 13.002 |

**Table 2.** Initial count of DCs for different agent populations and simulation speed-up with FLAME GPU

Besides this statistical comparison, we benchmarked the performance of BIS implemented in FLAME GPU against the RePast implementation. The benchmark was done by varying agent population from 8000 to 20000. Since the number of PCs in the simulation is fixed, the initial agent population was achieved by varying initial count of DCs from 500 to 3500 as shown in Table 2 and keeping count of other immune cells (B-cells, T-cells, NKs, MΦ, CTLs) unchanged.

The simulation was run on an Intel Core i7 2.67 GHz CPU with 6.00 GB RAM equipped with NVIDIA Tesla C2050 GPU on Windows 7 OS. The result of the benchmark was obtained by running 15 trials as the counts of agents varied significantly in each trial due to stochastic nature of simulation. Figure 10 illustrates plot for speed-up obtained with FLAME GPU against agent population. It was observed that computational performance increased by 13 times when

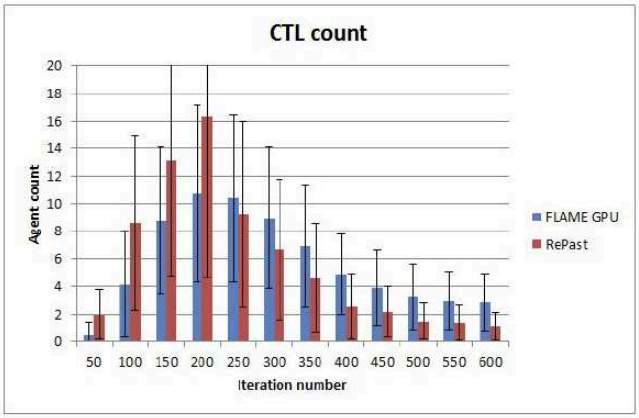



**Figure 9.** Immune cells count with standard deviation bars for immune win condition. a) CTL count for Zone 2. d) Dendritic cell count for Zone 2.

simulation was run for initial agent count was set to 20000 agents. However, performance analysis was not carried out for agent population greater than 20000 as BIS implemented in RePast cannot handle such large model sizes (it took over two hours to run a single realization of the model with 20000 agents). Hence, it can be noted that there is a significant improvement in computational performance.

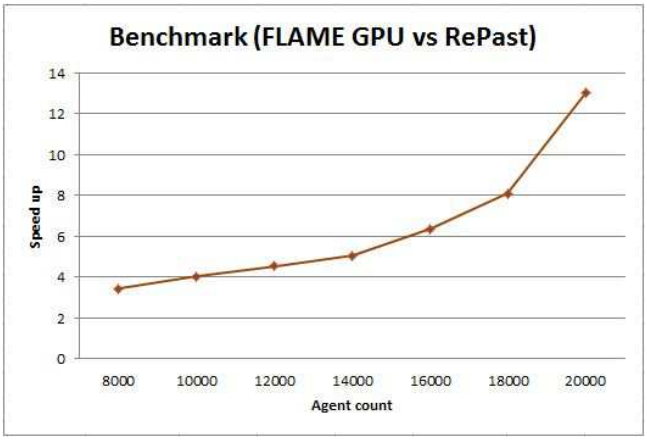



**Figure 10.** Benchmark: plot for speed-up obtained with FLAME GPU against agent count

## Conclusion

We have successfully implemented the Basic Immune Simulator (BIS) using FLAME GPU framework. Since FLAME GPU utilized computational power of GPUs using optimized CUDA code, we observed a significant improvement in performance as opposed to serial version of BIS implemented in RePast. It was possible to visualize the progression of the simulation in real time without having to write additional OpenGL code for visualization.

Apart from performance analysis of FLAME GPU, we also made statistical comparison for immune win condition with BIS modeled in RePast to validate accuracy of our results. We meticulously followed the state diagrams of different immune cell types implemented in the original BIS implementation to conserve the quality of simulation. We ran multiple simulations to find the mean and variance of the time trajectory of various variables. Since the simulation has a large stochastic component and we did not have access to the random seed as well as the algorithm used for generation of the random numbers in the original implementation in RePast, we can only check the accuracy of our implementation in a statistical sense. In this case, if the mean of the results of the two simulation were within the bounds of the variances, we conclude that parallelization does not induce any artifacts. As can be seen from the results, this is indeed the case.

We found that there is an initial learning curve to understanding the general structure of FLAME-GPU. An additional requirement is the knowledge of programming in C to program the state transition functions. The advantage of the FLAME-GPU function is the fact that it is agnostic as far as the underlying parallel GPU hardware as well as the specific API is concerned. Therefore, once the ABM is programmed, it will be possible to move on to newer hardware and API by simply running the pre-compiler. Currently, the pre-compiler only supports NVIDIA hardware and CUDA API.

Our future work will involve incorporating more details into the immune simulator. For example, the FLAME-GPU framework does not support solution of coupled PDEs for simulation of bio-chemical. In our current implementation, as in the previous RePast implementation, we have used a simple 2-D convolution with a stencil for modeling diffusion. A major enhancement will be to solve actual diffusion, reaction, advection PDEs in conjunction with the agent models. Furthermore, for more accurate representation of the immune agent interaction, we could possibly replace the ad-hoc rules that are currently present in the finite state models with rules based on actual chemical kinetics. This will increase the fidelity of the model. Of course, increase in fidelity will lead to larger computational load which we hope can be handled by the GPU.

## References

Alberts, B., A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter (2002). *Molecular Biology of the Cell*.

Amsen, D., J. M. Blander, G. R. Lee, K. Tanigaki, T. Honjo, and R. A. Flavell (2004). Instruction of distinct cd4 t helper cell fates by different notch ligands on antigen-presenting cells. *Cell 117*, 515–526.

Anderson, C. F., M. Lucas, L. Gutiérrez-Kobeh, A. E. Field, and D. M. Mosser (2004). T cell biasing by activated dendritic cells. *Journal of immunology (Baltimore, Md. : 1950) 173*, 955–961.

Anderson, C. F. and D. M. Mosser (2002). Cutting edge: biasing immune responses by directing antigen to macrophage fc gamma receptors. *Journal of immunology (Baltimore, Md. : 1950) 168*, 3697–3701.

Antia, R., C. T. Bergstrom, S. S. Pilyugin, S. M. Kaech, and R. Ahmed (2003). Models of cd8+ responses: 1. what is the antigen-independent proliferation program. *Journal of theoretical biology 221*(4), 585–598.

Badovinac, V. P., K. A. N. Messingham, A. Jabbari, J. S. Haring, and J. T. Harty (2005). Accelerated cd8+ t-cell memory and prime-boost response after dendritic-cell vaccination. *Nature medicine 11*, 748–756.

Beilhack, A. and S. G. Rockson (2003). Immune traffic: a functional overview. *Lymphatic research and biology 1*, 219–234.

Bernaschi, M., F. Castiglione, and S. Succi (1998). A parallel simulator of the immune response. In *High-Performance Computing and Networking*, pp. 161–172. Springer.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences 99*(suppl 3), 7280–7287.

Casadevall, A. and L.-a. Pirofski (2003, 9). Antibody-mediated regulation of cellular immunity and the inflammatory response. *Trends in immunology 24*(9), 474–8.

Castiglione, F. and M. Bernaschi (2004). C-immsim: playing with the immune response. In *Proceedings of the Sixteenth International Symposium on Mathematical Theory of Networks and Systems (MTNS2004)*.

Chiacchio, F., M. Pennisi, G. Russo, S. Motta, and F. Pappalardo (2014). Agent-based modeling of the immune system: Netlogo, a promising framework. *BioMed research international 2014*.

Clarke, K. C. (2014). Cellular automata and agent-based models. In *Handbook of Regional Science*, pp. 1217–1233. Springer.

Collier, N. and M. North (2013). Parallel agent-based simulation with repast for high performance computing. *Simulation*, 1215–1235.

DSouza, R. M., M. Lysenko, S. Marino, and D. Kirschner (2009). Data-parallel algorithms for agent-based model simulation of tuberculosis on graphics processing units. In *Proceedings of the 2009 Spring Simulation Multiconference*, pp. 21. Society for Computer Simulation International.

DSouza, R. M., M. Lysenko, and K. Rahmani (2007). Sugarscape on steroids: simulating over a million agents at interactive rates. In *Proceedings of the Agent 2007 Conference*, Volume 20.

Dubois, B., B. Vanbervliet, J. Fayette, C. Massacrier, C. Van Kooten, F. Briere, J. Banchereau, and C. Caux (1997). Dendritic cells enhance growth and differentiation of cd40-activated b lymphocytes. *J Exp Med 185*, 941–951.

Eilenberg, S. and B. Tilson (1974). *Automata, languages, and machines*, Volume 76. Academic press New York.

Figge, M. T. (2009). Optimization of immunoglobulin substitution therapy by a stochastic immune response model. *PloS one 4*(5), e5685.

Figueredo, G. P., P.-O. Siebers, M. R. Owen, J. Reps, and U. Aickelin (2014). Comparing stochastic differential equations and agent-based modelling and simulation for early-stage cancer. *PloS one 9*(4), e95150.

Folcik, V. A., G. C. An, and C. G. Orosz (2007). The basic immune simulator: an agent-based model to study the interactions between innate and adaptive immunity. *Theoretical biology & medical modelling 4*, 39.

Fouchet, D. and R. Regoes (2008). A population dynamics analysis of the interaction between adaptive regulatory t cells and antigen presenting cells. *PloS one 3*(5), e2306.

Gallucci, S., M. Lolkema, and P. Matzinger (1999, 11). Natural adjuvants: endogenous activators of dendritic cells. *Nature medicine 5*(11), 1249–55.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry 81*(25), 2340–2361.

Green, D. R., N. Droin, and M. Pinkoski (2003). Activation-induced cell death in t cells. *Immunological reviews 193*, 70–81.

Green, S. (2008). Cuda particles. *nVidia Whitepaper 2*(3.2), 1.

Guo, R.-F. and P. A. Ward (2005). Role of c5a in inflammatory responses. *Annual review of immunology 23*, 821–852.

Hart, D. N. (1997). Dendritic cells: unique leukocyte populations which control the primary immune response. *Blood 90*, 3245–3287.

Holcombe, M. (1988, 3). X-machines as a basis for dynamic system specification. *Software Engineering Journal 3*(2), 69.

Hou, W.-S. and L. Van Parijs (2004). A bcl-2-dependent molecular timer regulates the lifespan and immunogenicity of dendritic cells. *Nature immunology 5*, 583–589.

Huynh, M.-L. N., V. A. Fadok, and P. M. Henson (2002, 1). Phosphatidylserine-dependent ingestion of apoptotic cells promotes tgf-beta1 secretion and the resolution of inflammation. *The Journal of clinical investigation 109*(1), 41–50.

Ingulli, E., A. Mondino, A. Khoruts, and M. K. Jenkins (1997). In vivo detection of dendritic cell antigen presentation to cd4(+) t cells. *The Journal of experimental medicine 185*, 2133–2141.

Janeway, C., P. Travers, M. Walport, and M. Shlomchik (2001). *Immunobiology* (fifth ed.). Garland Science.

Koch, F., U. Stanzl, P. Jennewein, K. Janke, C. Heufler, E. Kämpgen, N. Romani, and G. Schuler (1996). High level il-12 production by murine dendritic cells: upregulation via mhc class ii and cd40 molecules and downregulation by il-4 and il-10. *The Journal of experimental medicine 184*, 741–746.

Kriehuber, E., W. Bauer, A. S. Charbonnier, D. Winter, S. Amatschek, D. Tamandl, N. Schweifer, G. Stingl, and D. Maurer (2005). Balance between nf-??b and jnk/ap-1 activity controls dendritic cell life and death. *Blood 106*, 175–183.

Mallet, D. G. and L. G. De Pillis (2006). A cellular automata model of tumor–immune system interactions. *Journal of Theoretical Biology 239*(3), 334–350.

Massaioli, F., F. Castiglione, and M. Bernaschi (2005). Openmp parallelization of agent-based models. *Parallel Computing 31*(10), 1066–1081.

Matzinger, P. (2002a). The danger model: a renewed sense of self. *Science (New York, N.Y.) 296*, 301–305.

Matzinger, P. (2002b). The danger model: a renewed sense of self. *Science (New York, N.Y.) 296*, 301–305.

Merrill, S. J. (1981). A model of the role of natural killer cells in immune surveillanceâĂŤi. *Journal of mathematical biology 12*(3), 363–373.

Miga, A. J., S. R. Masters, B. G. Durell, M. Gonzalez, M. K. Jenkins, C. Maliszewski, H. Kikutani, W. F. Wade, and R. J. Noelle (2001). Dendritic cell longevity and t cell persistence is controlled by cd154-cd40 interactions. *European Journal of Immunology 31*, 959–965.

Noble, D. (2002). The rise of computational biology. *Nature reviews. Molecular cell biology 3*, 459–463.

North, M. J., N. T. Collier, and J. R. Vos (2006). Experiences creating three implementations of the repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 16*(1), 1–25.

Onsum, M. and C. V. Rao (2007). A mathematical model for neutrophil gradient sensing and polarization. *PLoS Comput Biol 3*(3), e36.

Owens, J. D., D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell (2007). A survey of general-purpose computation on graphics hardware. In *Computer graphics forum*, Volume 26, pp. 80–113. Wiley Online Library.

Pancer, Z. and M. D. Cooper (2006). The evolution of adaptive immunity. *Annual review of immunology 24*, 497–518.

Puzone, R., B. Kohler, P. Seiden, and F. Celada (2002). Immsim, a flexible model for in machina experiments on immune system responses. *Future Generation Computer Systems 18*(7), 961–972.

Ricevuti, G. (1997, 12). Host tissue damage by phagocytes. *Annals of the New York Academy of Sciences 832*, 426–48.

Richmond, P., S. Coakley, and D. Romano (2009). Cellular level agent based modelling on the graphics processing unit. In *High Performance Computational Systems Biology, 2009. HIBI'09. International Workshop on*, pp. 43–50. IEEE.

Richmond, P. and D. Romano (2011). Template-driven agent-based modeling and simulation with cuda. In W.-m. Hwu (Ed.), *GPU Computing Gems Emerald Edition*, Chapter 21, pp. 313–324. Morgan Kaufmann.

Richmond, P., D. Walker, S. Coakley, and D. Romano (2010). High performance cellular level agent-based simulation with flame for the gpu. *Briefings in bioinformatics 11*(3), 334–347.

Satish, N., M. Harris, and M. Garland (2009). Designing efficient sorting algorithms for manycore gpus. In *IPDPS 2009 - Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium*.

Segal, A. W. (2005, 1). How neutrophils kill microbes. *Annual review of immunology 23*, 197–223.

Seiden, P. E. and F. Celada (1992). A model for simulating cognate recognition and response in the immune system. *Journal of theoretical biology 158*(3), 329–357.

Shi, Y., J. E. Evans, and K. L. Rock (2003). Molecular identification of a danger signal that alerts the immune system to dying cells. *Nature 425*, 516–521.

Shortman, K. and S. H. Naik (2007). Steady-state and inflammatory dendritic-cell development. *Nature reviews. Immunology 7*, 19–30.

Srivastava, P. (2002). Roles of heat-shock proteins in innate and adaptive immunity. *Nature reviews. Immunology 2*, 185–194.

Stetson, D. B., M. Mohrs, R. L. Reinhardt, J. L. Baron, Z.-E. Wang, L. Gapin, M. Kronenberg, and R. M. Locksley (2003). Constitutive cytokine mrnas mark natural killer (nk) and nk t cells poised for rapid effector function. *The Journal of experimental medicine 198*, 1069–1076.

Stundzia, A. B. and C. J. Lumsden (1996). Stochastic simulation of coupled reaction–diffusion processes. *Journal of computational physics 127*(1), 196–207.

Tanaka, H., C. E. Demeure, M. Rubio, G. Delespesse, and M. Sarfati (2000). Human monocyte-derived dendritic cells induce naive t cell differentiation into t helper cell type 2 (th2) or th1/th2 effectors. role of stimulator/responder ratio. *The Journal of experimental medicine 192*, 405–412.

Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences 237*(641), 37–72.

Van Dyke Parunak, H., R. Savit, and R. L. Riolo (1998). Agent-based modeling vs. equation-based modeling: A case study and users' guide. In *Multi-Agent Systems and Agent-Based Simulation*, pp. 10–25.

Varela, F. J. and J. Stewart (1990). Dynamics of a class of immune networks i. global stability of idiotype interactions. *Journal of Theoretical Biology 144*(1), 93–101.

Vieira, P. L., E. C. de Jong, E. A. Wierenga, M. L. Kapsenberg, and P. KaliÅĎski (2000). Development of th1-inducing capacity in myeloid dendritic cells requires environmental instruction. *Journal of immunology (Baltimore, Md. : 1950) 164*, 4507–4512.

Wittamer, V., J.-D. Franssen, M. Vulcano, J.-F. Mirjolet, E. Le Poul, I. Migeotte, S. Brézillon, R. Tyldesley, C. Blanpain, M. Detheux, A. Mantovani, S. Sozzani, G. Vassart, M. Parmentier, and D. Communi (2003). Specific recruitment of antigen-presenting cells by chemerin, a novel processed ligand from human inflammatory fluids. *The Journal of experimental medicine 198*, 977–985.

Yuan, Y. and L. J. Allen (2011). Stochastic models for virus and immune system dynamics. *Mathematical biosciences 234*(2), 84–94.

Zuniga, E. I., D. B. McGavern, J. L. Pruneda-Paz, C. Teng, and M. B. A. Oldstone (2004). Bone marrow plasmacytoid dendritic cells can differentiate into myeloid dendritic cells upon virus infection. *Nature immunology 5*, 1227–1234.

## Author Biographies

Mr. Shailesh Tamrakar received his MS in Mechanical Engineering from the University of Wisconsin Milwaukee in 2015. He was a Mechanical Engineering graduate student in the Complex Systems Simulation Laboratory under the guidance of Dr. Roshan D'Souza. He has interests in agent-based modeling and high-performance computing. He is currently working as a Research Programmer at Boston University - School of Public Health to develop ABMs for various public health applications.

Dr. Richmond recieved his PhD in 2010 from the Department of Computer Science at the University of Sheffield (TUoS). He is a EPSRC Research Software Engineering Fellow and Vice Chancellors Research Fellow at the University of Sheffield. His work focuses on facilitating the use of accelerated architectures such as Graphics Processing Units (GPUs) to accelerate scientific

discovery. He develops software techniques such as FLAME GPU, a multi agent GPU based simulator, to help embed the use of accelerators into mainstream science and engineering.

Dr. Roshan D'Souza recieved his PhD from the University of California, Berkeley in 2003 with a focus on manufacturing planning. He is an associate professor of Mechanical Enigneering at the University of Wisconsin-Milwaukee. His current research interests include high performance computing, agent-based modeling, and image processing. He is the reciepient of the National Science Foundation CAREER award, the 2008 ASME Computers and Information in Engineering Young Engineer Award, and the 2003 ASME Computer and Information in Engineering Best Paper Award.

### Funding

This work was partially funded by a grant 2R01GM077138-04A1 from the National Institutes of Heatlh (NIH). Any opinions and conclusions reached in this manuscript are of the authors' alone and do not necessarily reflect those of the NIH.

## Copyright statement

Please be aware that the use of this LaTeX $2_\varepsilon$ class file is governed by the following conditions.

### *Copyright*