



"Computational modeling of decision making in stem cells"

PROJECT REPORT

Submitted in partial fulfillment for
the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Engineering

By

Pratik Varshney

Faculty No.: 11-PEB-002

Enrolment No.: GE0004

Abhay Mittal

Faculty No.: 11-PEB-005

Enrolment No.: GD8309

Under the guidance of

Prof. Nesar Ahmad

Department of Computer Engineering
Zakir Husain College of Engineering and Technology
Aligarh Muslim University
Aligarh (U.P.) – 202002

Z. H. College of Engineering & Technology
Aligarh Muslim University



CERTIFICATE

This is to certify that the project entitled “Computational modeling of decision making in stem cells” being submitted by Abhay Mittal and Pratik Varshney as their major project for the 8th semester in partial fulfilment of the degree of Bachelor of Technology in Computer Engineering has been carried out under my supervision and guidance and, to the best of my knowledge, has not been submitted anywhere else for the award of any other degree.

(Prof. Nesar Ahmad)

Professor and Chairman

Department of Computer Engineering

Zakir Husain College of Engineering and Technology

Aligarh Muslim University

Aligarh - 202002

Z. H. College of Engineering & Technology
Aligarh Muslim University



DECLARATION

We declare that the project entitled “Computational modeling of decision making in stem cells” being submitted by us is a record of our own work. This project report has not been submitted to any other Institute for the award of any other degree. This written submission represents our ideas in our own words and where others' ideas have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or falsified or fabricated any idea/fact/source in our submission.

(Pratik Varshney)
11-PEB-002

(Abhay Mittal)
11-PEB-005

ACKNOWLEDGEMENT

We would sincerely like to thank our supervisor Prof. Nesar Ahmad who guided and encouraged us throughout the project and provided us with valuable suggestions whenever we faced any problems. He regularly checked our progress and provided his valuable time and effort throughout the project.

We would also like to sincerely thank our senior Mr. Sandeep Kumar, PhD Scholar, IIT Bombay who helped us in selecting this project and clearing our doubts. He also helped us in becoming familiar with the topic of stem cells. He has been a mentor to us throughout this project.

We are also sincerely thankful to all our teachers and the Non-Teaching staff of Zakir Husain College of Engineering and Technology for their kind attitude, unhesitant support and their willingness to help us time to time.

Finally, we would also like to express our gratitude towards our family members and friends for supporting and encouraging us throughout our study.

ABSTRACT

Stem cells are the special cells in our body which have the ability to develop ‘mutations’ in them. By mutation they acquire different properties and adapt to different environment. For example, if we put stem cells in hard medium, they become bone like and if we put them in soft medium they acquire the properties of brain cells (i.e. neurons). How stem cells decide upon the suitable mutation and how does that mutation help them in adapting the environment is a fundamental and an intriguing question about stem cells. There are few known properties of stem cells and we know that how do these properties vary with substrate (environment) properties. The main objective of this project is to develop a program which can model the life cycle of the stem, transit amplifying and terminally differentiated cells in 3-dimensional space.

Keywords: Stem cells, cellular automaton, simulation, computational modeling

Table of Contents

1. Introduction	2
1.1. Cell	2
1.2. Extracellular Matrix	3
1.3. Concept of Proteins and their role in biological systems	4
1.4. Cells as signal analysis device	5
1.5. Cellular differentiation	6
1.6. Stem Cells	6
1.7. Transit Amplifying Cells.....	6
1.8. Terminally Differentiated Cells	7
1.9. Difference between TD and Stem cells.....	7
1.10. Role of stem cells in our body.....	7
1.11. Cellular Division in Stem Cells.....	8
1.11.1. Symmetric Division	8
1.11.2. Asymmetric Division.....	8
1.12. Cell Lineage	9
1.13. Cellular Automata	11
1.14. Organization of the remainder of the report.....	13
2. Tools and Languages Used	15
2.1. C++.....	15
2.2. Java.....	16
2.3. Python.....	16
2.4. Doxygen v1.8.8	17
2.5. Code::Blocks v13.12 with GNU GCC v4.8.....	17
2.6. libxml++ v 2.6.....	18
2.7. Paraview v4.3.1	18

2.8.	Bitbucket with Git version control.....	19
2.9.	MATLAB	19
2.10.	NetBeans IDE 8.0.....	20
3.	Implementation	22
3.1.	Classes	22
3.1.1.	AutomatonCell.....	22
3.1.2.	Cell.....	22
3.1.3.	Environment.....	22
3.1.4.	Line	22
3.1.5.	Point	22
3.1.6.	Simulation	22
3.1.7.	SimulationParameters	22
3.1.8.	StemCell.....	22
3.1.9.	TACell.....	22
3.1.10.	Utilities.....	22
3.1.11.	Var.....	22
3.2.	Class Diagram	23
3.3.	Description of Configuration File	24
3.4.	Flowchart of main program.....	26
3.5.	Major Operations.....	31
3.5.1.	Setting up Extra Cellular Matrix.....	31
3.5.2.	Determination of favorable location	31
3.5.3.	Move Cells	32
3.5.4.	Update E-Cadherin / β -Catenin (EB).....	33
3.5.5.	Cell Division	34
3.5.6.	Save the state of the environment	35
3.5.7.	TA Cell differentiation.....	36

3.5.8. Evolve Genetic Code	36
3.5.9. Cell Death	38
4. User Interface Screenshots.....	40
4.1. User Interface	40
4.1.1. CLI	40
4.1.2. GUI.....	40
4.2. Program UI Screenshots.....	41
5. Results	49
5.1. Simulation Plots	49
5.2. Visualization Screenshots	54
6. Conclusion and Future Work	59
6.1. Conclusion.....	59
6.2. Future Work	59
References	60

Introduction

1. Introduction

Stem cell research is being very actively pursued in various biological laboratories around the world. These cells have the unique property that they can develop into other cell types in an organism's body during early stages of life and growth. Scientists are working on developing custom transplants by manipulating stem cells for patients. Also, stem cell research is playing a key role in demonstrating how an organism grows out from a single cell and how damaged cells are replaced by healthy cells in an adult organism.

In this project we are developing a program which can mimic the phenomenon of decision making in stem cells.

1.1. Cell

A cell is considered the most fundamental unit of life. Every living organism is composed of cells. Adult humans can be considered a collection of 100 trillion cells (1). Each cell is composed of various molecules which vary in size and shapes. A cell membrane works as a filter and controls the transfer of substances between the environment and the cell (2). Some cells also contain a cell wall which strengthens the cell and also protects it mechanically and chemically.

There are two main types of cells:

a. Prokaryotic Cells

In these cells the genetic material (DNA) is not enclosed within a nuclear envelope (3). Generally, the term Prokaryotes refers to single celled organisms such as bacteria (1). The structure of a prokaryotic cell is shown in Figure 1(a).

b. Eukaryotic Cells

In these cells the genetic material (DNA) is enclosed within a nuclear envelope. The genomes of these cells are more complex than those of Prokaryotic cells. Also, these cells are generally larger and more complex than Prokaryotic cells (3). The structure of a eukaryotic cell is shown in Figure 1(b).

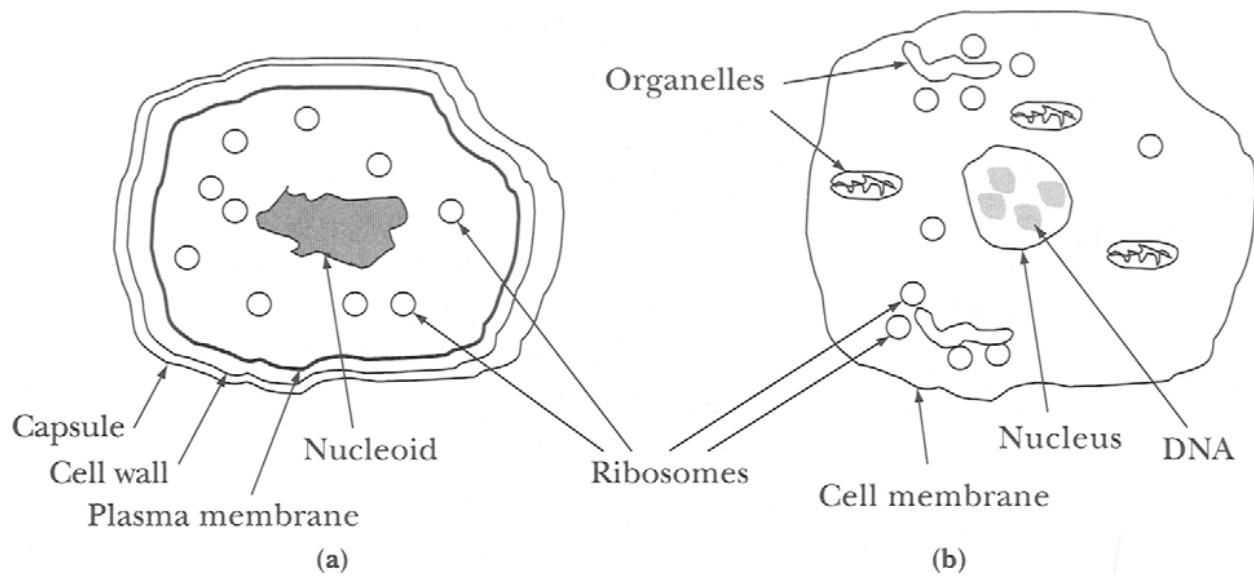


Figure 1: The structure of a (a) Prokaryotic cell and (b) Eukaryotic cell. Image taken from (1)

1.2. Extracellular Matrix

In multi cellular organisms, the spaces between cells are filled with the extracellular matrix. This matrix binds the cells and tissues together. It consists of numerous extracellular molecules like proteins and polysaccharides which are secreted by the cells in the extracellular matrix. Thus, the extracellular matrix is formed mainly by the cells inside it. These cells are also responsible for organizing the matrix (4). There are many types of extracellular matrices. It is most abundant in connective tissues (3) and is essential for functions like growth and wound healing and plays a key role in providing tissues with their mechanical strength and their elastic properties. It also helps in regulating cell proliferation and differentiation (5).

There are two types of cellular organization:

- a. Each cell is surrounded by the extracellular matrix. (e.g. - blood cells)
- b. Groups of cells are surrounded by the extracellular matrix (e.g. - skin cells)

1.3. Concept of Proteins and their role in biological systems

Proteins are linear polymers of amino acids. The amino acids are linked together using peptide bonds. The main task of proteins is to perform the tasks directed by the genetic information of the cell (3). Proteins are "polymer" like molecules that are well structured. Every protein has a unique three dimensional structure. A protein named collagen is the main constituent of the extracellular matrix and is responsible for providing support to various body parts like skin, bones, muscles, etc. (1). Proteins have a special ability to bind with other molecules and this binding determines their activity. The strength of the binding varies from tight to very weak. A protein cannot bind with all the molecules that are in its environment. Each protein has some selected molecules only with which it can bind (4) and such molecules are known as **ligands**. The binding occurs using weak non-covalent bonds like ionic bonds, hydrogen bonds, etc. Proteins may also change their shapes (reversibly) during the binding process. The binding is depicted in the Figure 2.

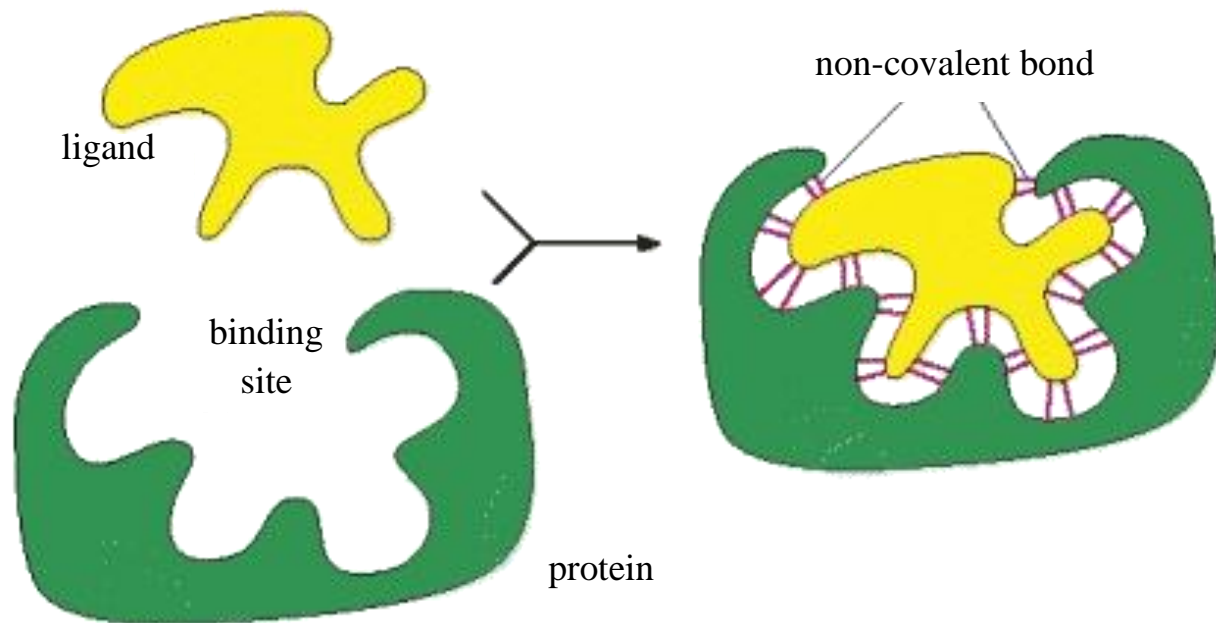


Figure 2: Selective binding of protein with another molecule. Image taken from (4)

The most important role of proteins is that they act as catalysts in almost all the chemical reactions that occur inside a cell. These type of proteins are also known

as **enzymes** and such proteins direct almost all of the activities that occur inside a cell (3). Apart from acting as catalysts, proteins also act as motors, signal receptors, switches or tiny pumps (4).

1.4. Cells as signal analysis device

In order to survive, cells need to receive and process information (which is in the form of signals) from the external environment. The information can be regarding the availability of nutrients, or temperature and light conditions of the environment (6). Unicellular organisms use signaling to communicate with other organisms in their population to carry out tasks in a team.

There are two types of signals that cells respond to

a. Chemical Signals

Majority of the signals that cell respond to are chemical in nature. Examples are growth factors, extracellular matrix components in multi cellular organisms.

b. Mechanical Signals

Examples are the sensory cells that are present on our skin. These cells are sensitive to pressure of touch. Such sensory cells are also present in the ear which respond to the sound waves.

Cells respond to signals with the help of certain proteins known as **receptors**. These proteins bind to the signaling molecules. Each cell has numerous receptors located on the periphery, interior and even inside the nucleus of the cell (6).

Whenever a signaling molecule binds with a receptor present on the surface of the cell, a chain of reactions is triggered. This chain is responsible for amplifying the signal and carrying it inside the cell. It is because of cell signaling that tissues are able to carry out tasks which in no condition can be performed by individual cells. Signaling pathways play a crucial role in maintaining the state of equilibrium in a tissue. This state of equilibrium is known as **homeostasis** (6).

1.5. Cellular differentiation

Cellular differentiation is the process by which a less specialized cell is converted into a more specialized cell. This process is qualitative and thus, a cell can differentiate with respect to another cell only. This is common in stem cells (discussed in the next section). e.g. - A stem cell may differentiate into a brain cell eventually.

Based on differentiation, we can classify cells into three categories:

- a. Stem cells:** These are undifferentiated or very less differentiated cells. These cells are precursor to some other cells.
- b. Specialized cells:** These cells are highly differentiated to perform specific tasks. Examples are liver cells, brain cells, etc.
- c. Transit amplifying cells:** These cells are more differentiated than stem cells but less differentiated than specialized cells.

1.6. Stem Cells

Stem cells are cells which have high capacity of self renewal during the whole life of the organism and that can produce at least one type of highly differentiated cell (7). Stem cells in an adult generally exist in small numbers in locations which are tissue specific. These locations are known as **niches** and they provide precisely regulated micro environment to properly nurture stem cell activity and sustain self renewal (8). Embryonic stem cells have the potential to differentiate into all the cell types of the body. In other words, embryonic stem cells are **pluripotent**.

1.7. Transit Amplifying Cells

Generally, between the stem cell and its highly differentiated descendent (progeny), there is a series of transit amplifying cells which act as progenitors having limited capacity of proliferation and restricted differentiation potential (7). The transit amplifying cells can either behave like stem cells or undergo

differentiation. These cells show intermediate properties between stem cells and their terminally differentiated descendents.

1.8. Terminally Differentiated Cells

Terminally Differentiated Cells (TDCs) are specialized cells that perform specific tasks. Examples of TDCs are brain cells, liver cells, etc.

1.9. Difference between TD and Stem cells

Stem cells differ from Terminally Differentiated cells in the following ways (9) :

- During the lifespan of an organism, stem cells have unlimited self renewal capacity.
- Stem cells can undergo asymmetric cell divisions, i.e., one daughter is itself a stem cell while the other daughter will eventually differentiate terminally.
- The process of asymmetric division is irreversible as daughters committed to undergo terminal differentiation are not stem cells.

Also the property of stemness allows stem cells to modify their internal signal processing mechanism significantly.

1.10. Role of stem cells in our body

Stem cells are found in many regions of our body like brain, liver (7), bone marrow, epidermis, intestinal epithelium (9), etc.. Cell divisions have different contributions during embryonic development and during adult life. During the embryonic phase, the rapid proliferation of embryonic stem cells characterizes early development. These cells then differentiate to form the specialized cells consisting various body parts and organs and thus increase the total number of cells. The rate of proliferation also decreases as the level of differentiation increases. In an adult, the major role of stem cells is to maintain the state of equilibrium in tissues (homeostasis) and also help in healing during an injury by

producing new cells rapidly. In an adult, cell divisions maintain the number of differentiated cells at a constant level. In tissues with permanently renewing populations like blood, testis, the terminally differentiated cells have a very short life span. They are replaced through proliferation of stem cells (9).

1.11. Cellular Division in Stem Cells

Two types of cellular division occurs in stem cells:

1.11.1. Symmetric Division

In this type of division, a parent stem cell divides into two daughter cells which are identical to the parent cell. Thus both the daughters are stem cells having the same properties as the parent. It is depicted in the figure below.

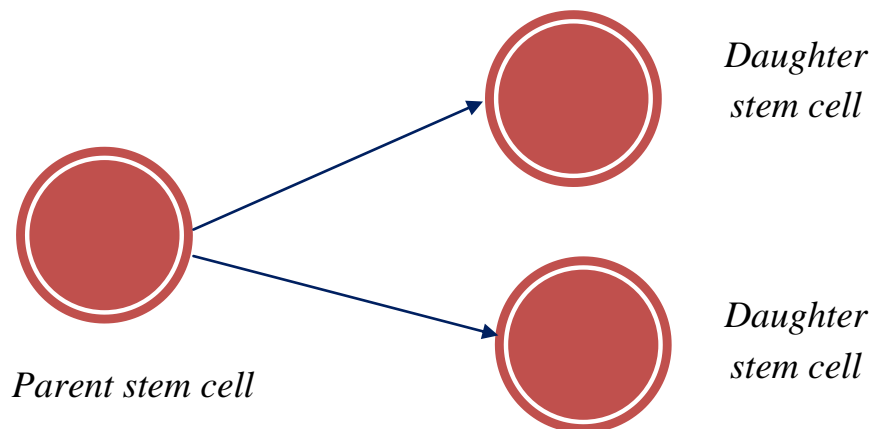


Figure 3: Symmetric Division in stem cells

1.11.2. Asymmetric Division

In this type of division, a parent stem cell divides into an identical daughter cell and one transit amplifying cell. The transit amplifying cell eventually differentiates into a specialized cell. It is depicted in the figure below.

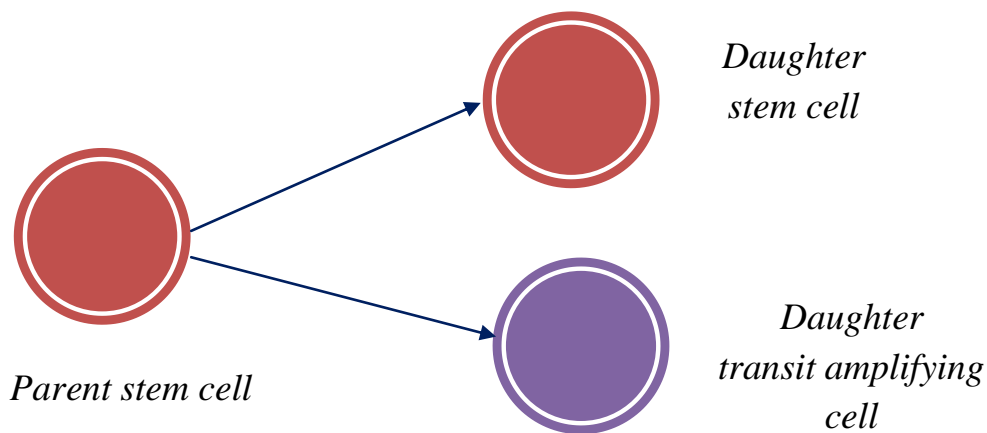


Figure 4: Asymmetric Division in stem cells

1.12.Cell Lineage

Everything begins with stem cells. As mentioned previously, a stem cell can divide both symmetrically as well as asymmetrically. In asymmetric division a stem cell creates a transit amplifying cell. TACs undergo several rounds of symmetric division (a.k.a. transient amplification) and finally differentiate into Terminally Differentiated Cells (TDCs). TDC are the normal human body cells like skin cells, brain cells, etc. TDCs have a certain life span and they die once they reach that age.

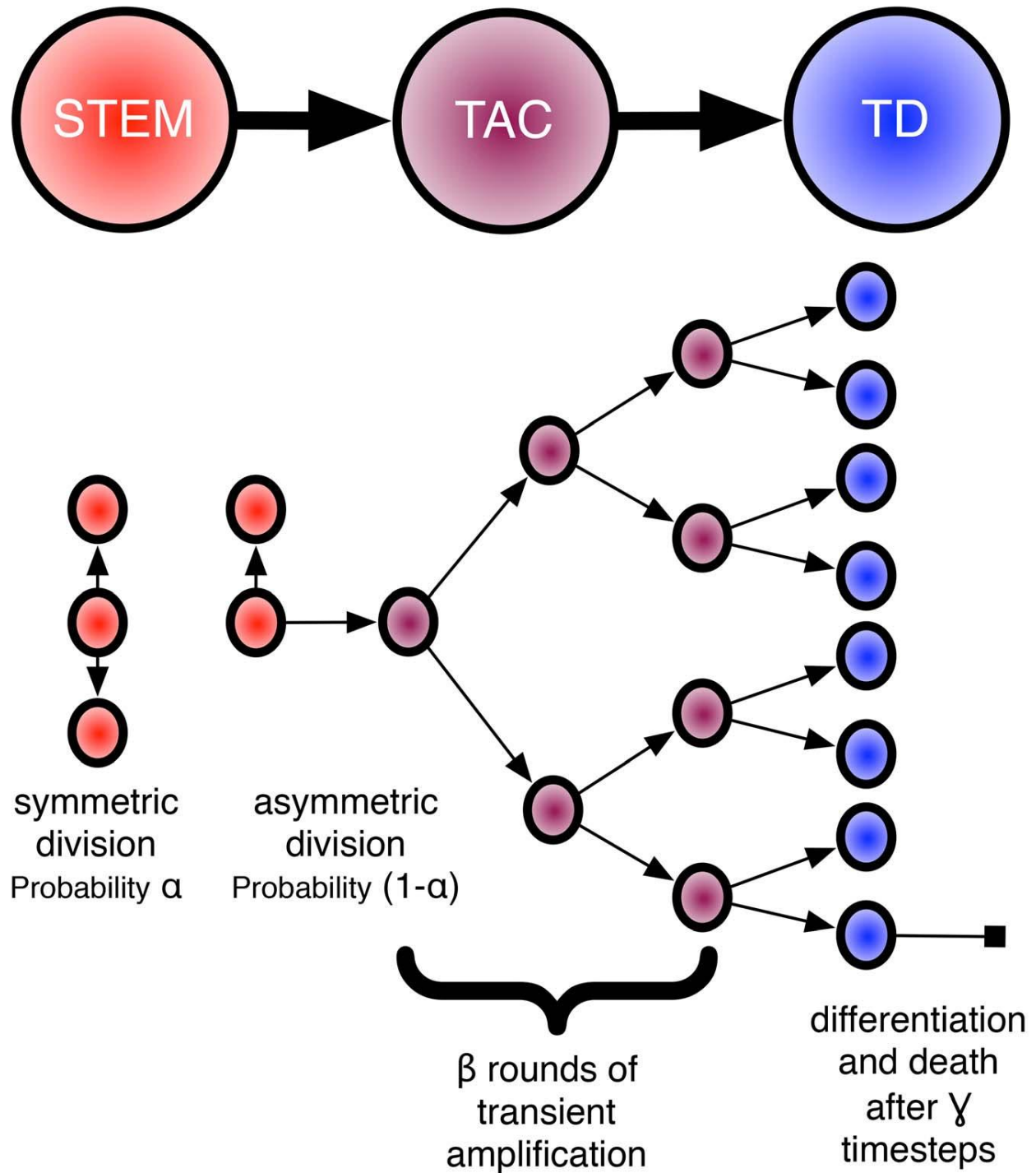


Figure 5: Cell Lineage. Image taken from (10)

1.13. Cellular Automata

A Cellular automaton (11) is a system containing a collection of cells on a grid such that

- Each cell has an associated state.
- There is a neighborhood of each cell.
- There is a set of rules and based on these rules, the grid updates itself (evolves) at discrete time steps.

This concept was developed by John von Neumann and Stanislaw Ulam in the 1940s. The concept of self reproducing systems based on cellular automata was also suggested by Von Neumann in the 1950s. Each cell in the grid of a cellular automaton is like a finite state automaton whose state depends on the states of the cell and its neighbors in the previous generation (12).

The simplest example of a cellular automaton is a 1D binary cellular automaton in which a cell can have only two states - 0 and 1 and only two neighbors - one to the right and other on the left.

Let the set of rules governing the automaton be as follows

INPUT	111	110	101	100	011	010	001	000
OUTPUT	0	1	0	1	1	0	1	0

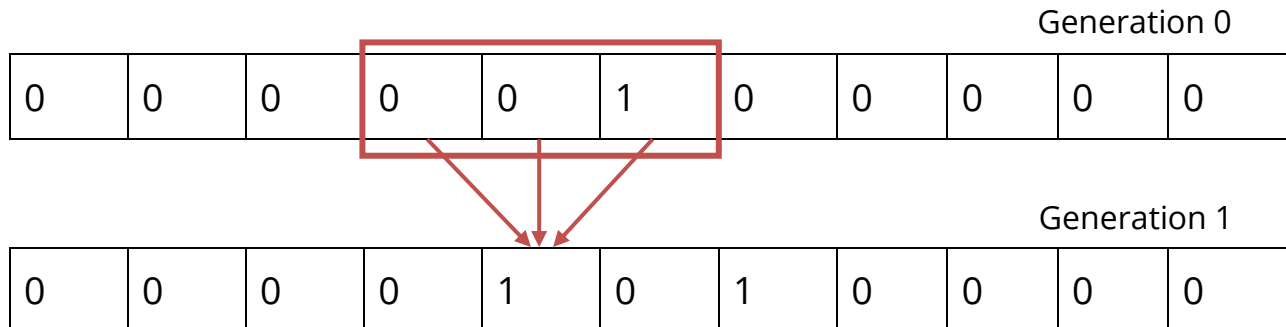
Each input has 3 digits. Starting from left, the first digit is the state of the left neighbor, the second digit is the state of the cell and the third digit is the state of the right neighbor.

If the combine the digits of output in one number (e.g. for the above case - 01011010) we get a 8 bit binary number. We know that an 8 bit binary number can have 256 values. Thus there can be 256 possible rule sets for this case of a 1D cellular automaton where the neighbors are adjacent to the cell. The name of the rule is the value of the 8 bit binary number in decimal. Hence the rule given above is known as Rule 90 as $(01011010)_2 = (90)_{10}$.

If we consider generation 0 to be as

0	0	0	0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---

Then, the transition from generation 0 to generation 1 can be seen as



Here, we have considered it as a circular grid, i.e. the left neighbor of the leftmost cell is the rightmost cell and the right neighbor of the rightmost cell is the leftmost cell. Another approach can be to delete the cells present at left and right extremities in generation 0 on reaching generation 1. A third approach can be to keep the states of leftmost and rightmost cells constant throughout all the generations. If we color the cells (0 for white and 1 for black) and apply the rules for a number of generations and put each generation below its previous generation, we get the pattern shown in figure 4.

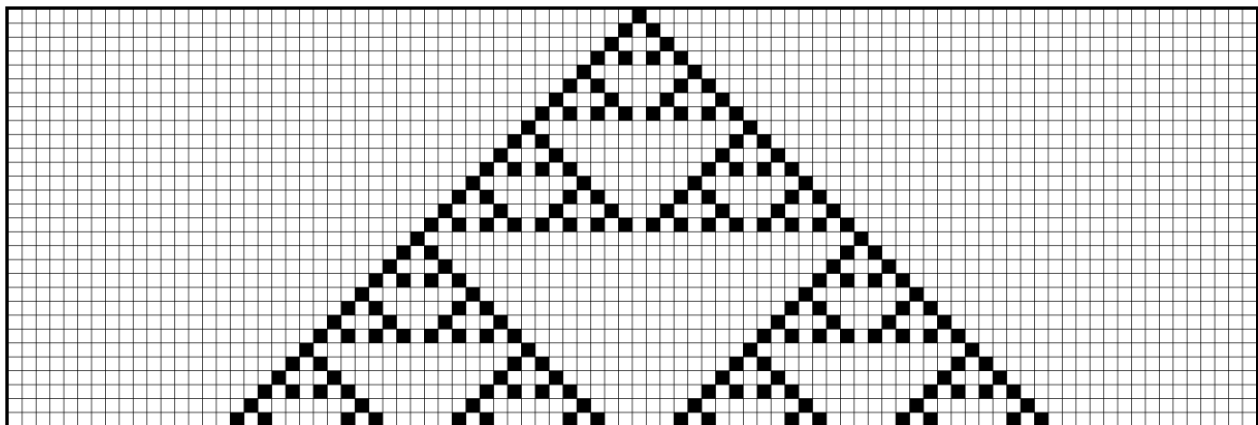


Figure 6: The output of rule 90. Image taken from (11)

The shape formed by Rule 90 (shown in Figure 4) is a fractal pattern and is known as the "Sierpiński triangle".

Though originally generated for simulating artificial life, cellular automata have found their use in a number of fields. They are used in pathology to analyze blood smears (12), random number generators, designing error correction codes, etc.

1.14. Organization of the remainder of the report

Chapter 2 contains description about the tools and languages used in the project.

Chapter 3 contains the details regarding our implementation.

Chapter 4 contains the screenshots of our user interface.

Chapter 5 contains our simulation results.

Chapter 6 states the conclusion and future work.

Tools and Languages Used

2. Tools and Languages Used

2.1. C++

C++ (13) is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing the facilities for low-level memory manipulation.

It is designed with a bias toward system programming (e.g., for use in embedded systems or operating system kernels), with performance, efficiency and flexibility of use as its design requirements. C++ has also been found useful in many other contexts, including desktop applications, servers (e.g. e-commerce, web search or SQL servers), performance-critical applications (e.g. telephone switches or space probes), and entertainment software. C++ is a compiled language, with implementations of it available on many platforms and provided by various organizations, including the FSF, LLVM, Microsoft and Intel.

C++ is standardized by the International Organization for Standardization (ISO), with the latest (and current) standard version ratified and published by ISO in December 2014 as ISO/IEC 14882:2014 (informally known as C++14). The C++ programming language was initially standardized in 1998 as ISO/IEC 14882:1998, which was then amended by the C++03, ISO/IEC 14882:2003, standard. The current C++14 standard supersedes these and C++11, with new features and an enlarged standard library. Before the initial standardization in 1998, C++ was developed by Bjarne Stroustrup at Bell Labs, starting in 1979, who wanted an efficient flexible language (like the C language), which also provided high-level features for program organization.

Many other programming languages have been influenced by C++, including C#, Java, and newer versions of C (after 1998).

We used C++ to carry out all the major operations of our program due to its high speed, efficiency and the availability of the Standard Template Library.

2.2. Java

Java (14) is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2015, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

We have used Java to create the GUI of our program.

2.3. Python

Python (15) is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

Python was used to render the vizualization in Paraview as Paraview only supports Python scripting.

2.4. Doxygen v1.8.8

Doxygen (16) is the de facto standard tool for generating documentation from annotated C++ sources.

Doxygen can help developers in three ways

1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in \LaTeX) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
2. Developers can configure doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. Doxygen can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
3. Developers can also use doxygen for creating normal documentation.

We used Doxygen to generate a documentation for our project.

2.5. Code::Blocks v13.12 with GNU GCC v4.8

Code::Blocks (17) is a *free C, C++ and Fortran IDE* built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable. Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin.

The GNU (18) Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Java, Ada, and Go, as well as libraries for these languages (libstdc++, libgcj,...). GCC was originally written as the compiler for the GNU operating system. The GNU system was developed to be 100% free software, free in the sense that it respects the user's freedom.

2.6. libxml++ v 2.6

libxml++ (19) is a C++ API for the popular libxml XML parser, written in C. libxml is famous for its high performance and compliance to standard specifications, but its C API is quite difficult even for common tasks.

libxml++ presents a simple C++-like API that can achieve common tasks with less code. Unlike some other C++ parsers, it does not try to avoid the advantages of standard C++ features such as namespaces, STL containers or runtime type identification, and it does not try to conform to standard API specifications meant for Java. Therefore libxml++ requires a fairly modern C++ compiler such as g++ 3.

But libxml++ was created mainly to fill the need for an API-stable and ABI-stable C++ XML parser which could be used as a shared library dependency by C++ applications that are distributed widely in binary form. That means that installed applications will not break when new versions of libxml++ are installed on a user's computer. Gradual improvement of the libxml++ API is still possible via non-breaking API additions, and new independent versions of the ABI that can be installed in parallel with older versions. These are the general techniques and principles followed by the GNOME project, of which libxml++ is a part.

We used libxml++ to read the configuration file from a C++ program.

2.7. Paraview v4.3.1

ParaView (20) is an open-source, multi-platform data analysis and visualization application. ParaView users can quickly build visualizations to analyze their data using qualitative and quantitative techniques. The data exploration can be done interactively in 3D or programmatically using ParaView's batch processing capabilities. . ParaView runs on distributed and shared memory parallel as well as single processor systems and has been successfully tested on Windows, Linux, Mac OS X, IBM Blue Gene, Cray XT3 and various Unix workstations and clusters. Under the hood, ParaView uses the Visualization Toolkit as the data processing and rendering engine and has a user interface written using the Qt cross-platform application framework.

ParaView was developed to analyze extremely large datasets using distributed memory computing resources. It can be run on supercomputers to analyze datasets of petascale size as well as on laptops for smaller data, has become an integral tool in many national laboratories, universities and industry, and has won several awards related to high performance computation.

We have used Paraview to render our simulation results.

2.8. Bitbucket with Git version control

Bitbucket (21) is a hosting site for the distributed version control systems (DVCS) Git and Mercurial. The service offering includes an issue tracker and wiki, as well as integration with a number of popular services such as Basecamp, Flowdock, and Twitter.

Git (22) is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

2.9. MATLAB

MATLAB® (23) is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java®. You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing.

Key Features (24)

- High-level language for numerical computation, visualization, and application development
- Interactive environment for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
- Built-in graphics for visualizing data and tools for creating custom plots
- Development tools for improving code quality and maintainability and maximizing performance
- Tools for building applications with custom graphical interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft® Excel®

We have used MATLAB to generate plots of the statistics gathered during the simulation.

2.10.NetBeans IDE 8.0

NetBeans IDE (25) lets you quickly and easily develop Java desktop, mobile, and web applications, as well as HTML5 applications with HTML, JavaScript, and CSS. The IDE also provides a great set of tools for PHP and C/C++ developers. It is free and open source and has a large community of users and developers around the world.

Design GUIs for Java SE, HTML5, Java EE, PHP, C/C++, and Java ME applications quickly and smoothly by using editors and drag-and-drop tools in the IDE.

NetBeans IDE can be installed on all operating systems that support Java, from Windows to Linux to Mac OS X systems.

Implementation

3. Implementation

3.1. Classes

3.1.1. AutomatonCell

This class represents a point in the simulation environment.

3.1.2. Cell

This class represents a terminally differentiated cell.

3.1.3. Environment

This class provides functions for setting up the simulation environment.

3.1.4. Line

This class represents a line (viz. the ECM fiber) in 3-dimensional space (viz. the simulation environment).

3.1.5. Point

This class represents a point in the 3-dimensional space.

3.1.6. Simulation

This class provides necessary functions for performing the simulation.

3.1.7. SimulationParameters

This class provides necessary functions for initializing and retrieving the simulation parameters.

3.1.8. StemCell

This class models the Stem Cells and is derived from the Cell class.

3.1.9. TACell

This class models the Transit Amplifying Cells and is derived from the Cell class.

3.1.10. Utilities

This class provides necessary functions for generating the Output.

3.1.11. Var

This class represents a coordinate variable.

3.2. Class Diagram

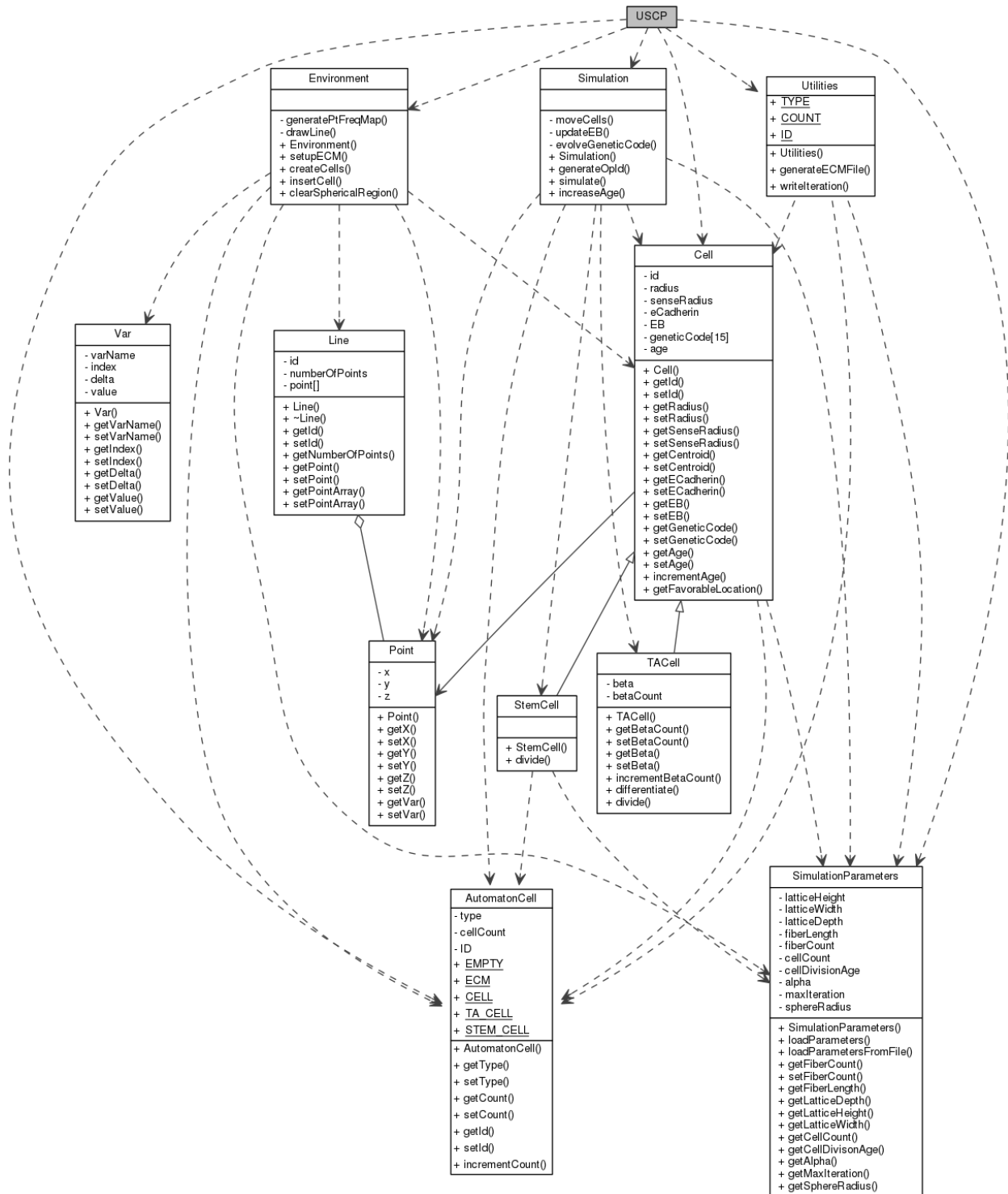


Figure 7: Class Diagram

3.3. Description of Configuration File

Filename : *"uscp.conf"*

This configuration file contains the value of the environment variables for the simulation operation. It is a standard XML file. It allows the user to easily modify the variables as per the requirement before starting the simulation.

Sample:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Parameters>

    <Lattice>

        <Width>40</Width>
        <Height>40</Height>
        <Depth>40</Depth>

    </Lattice>

    <Alpha>0.5</Alpha>

    <Beta>20</Beta>

    <Gamma>30</Gamma>

    <FiberCount>20000</FiberCount>

    <FiberLength>3</FiberLength>

    <CellDivisionAge>3</CellDivisionAge>

    <MaxIteration>500</MaxIteration>

    <SphereRadius>10</SphereRadius>

</Parameters>
```


Description:

Below is the description of the XML attributes:

- **Parameters** : *The root of xml file.*
 - **Lattice** : *Represents the simulation environment.*
 - **Height** : *The Height of the lattice.*
 - **Width** : *The Width of the lattice.*
 - **Depth** : *The Depth of the lattice.*
 - **Alpha** : *The probability of symmetric division of stem cells.*
 - **Beta** : *The number of rounds of amplification a TA cell undergo before differentiation.*
 - **Gamma** : *The age at which TDC (i.e. normal cell) die.*
 - **FiberCount** : *The number of fibers in the lattice.*
 - **FiberLength** : *The Length of each fiber.*
 - **CellDivisionAge** : *The age after which the cell divides.*
 - **MaxIteration** : *The maximum number of simulation iterations.*
 - **SphereRadius** : *The Radius of the empty spherical region at the center.*

3.4. Flowchart of main program

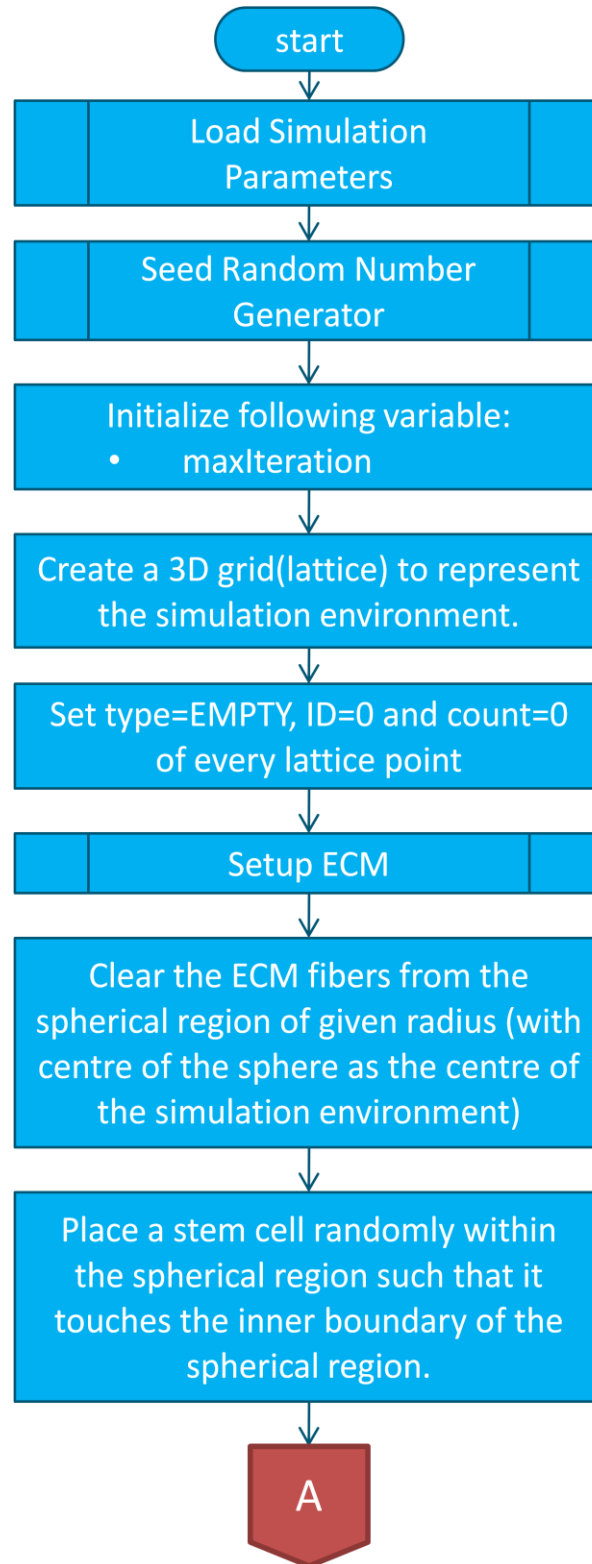


Figure 8 : Flowchart of main program

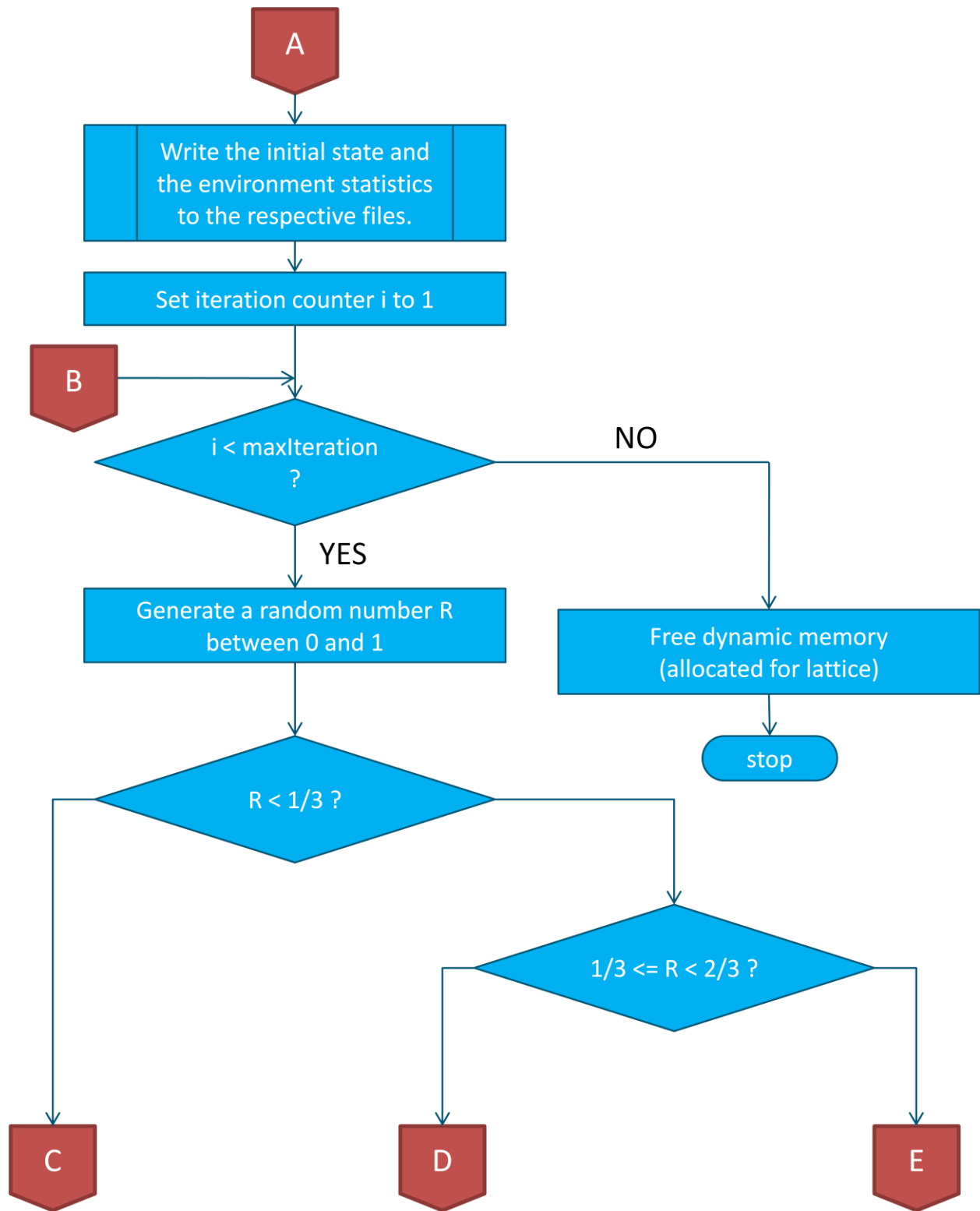


Figure 8: (Continued) Flowchart of main program

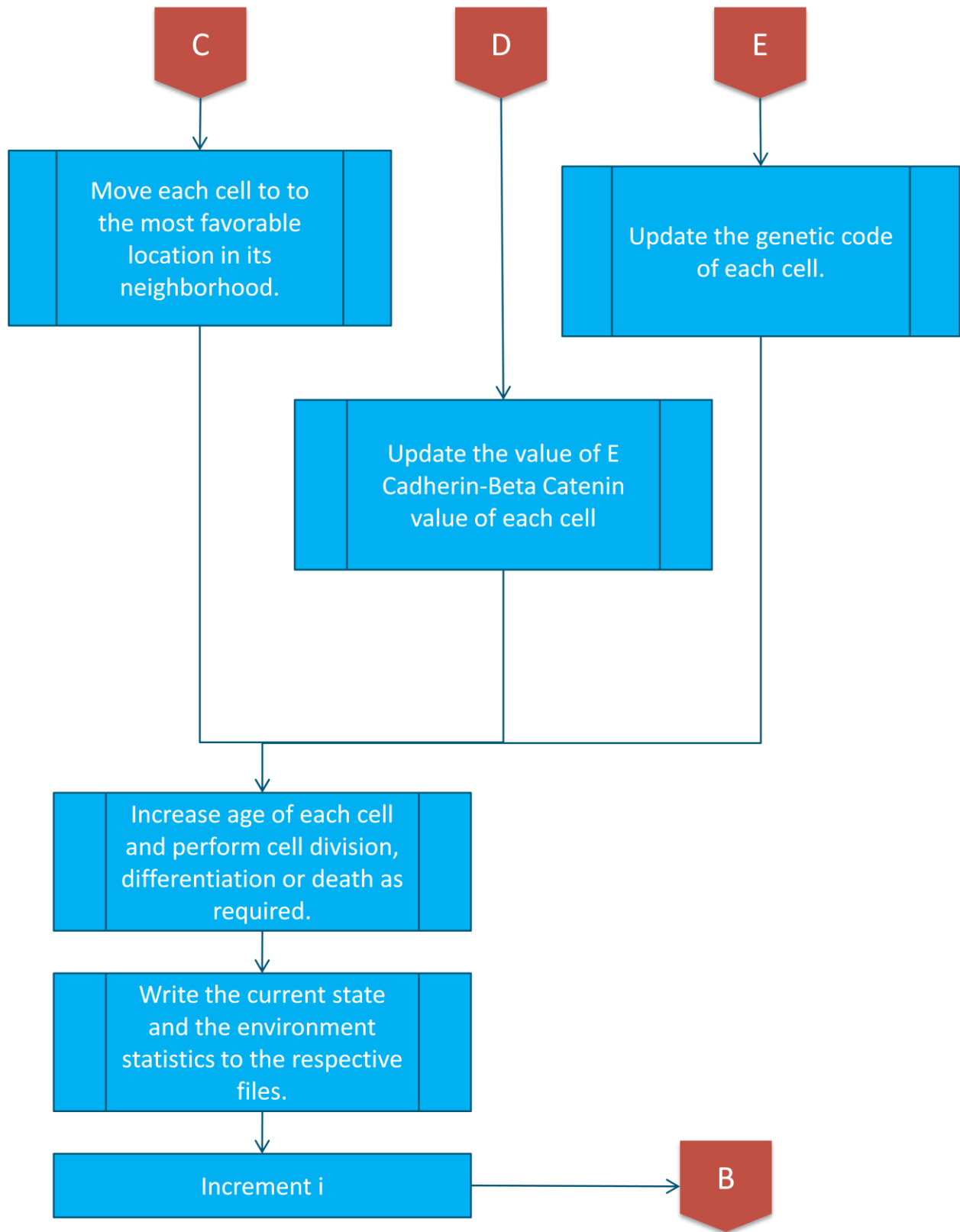


Figure 8: (Continued) Flowchart of main program

Description :

The main program is responsible for loading the simulation parameters, seeding the random number generator with system time (so that we can randomly decide the position of all the cells and ECM fibers and can also select a simulation operation randomly for execution), setting up the simulation environment (creating a virtual cuboidal space to represent the simulation environment) and performing various simulation operations and writing the states (starting, intermediate and final states) of the environment.

Following is the execution sequence of the program :

1. Load the simulation parameters from the configuration file.
2. Seed the random number generator.
3. Load the number of iterations to be performed in a variable (we have taken maxIteration variable).
4. Create a 3-dimentional lattice to represent the simulation environment.
5. Initialize the properties of every lattice point with default values (we have take 0 as default value) and set type to EMPTY (i.e. nothing is present).
6. Setup the simulation environment by creating the required number of ECM fibers (as specified in the configuration file).
7. Clear the ECM fibers from the spherical region of given radius (with centre of the sphere as the centre of the simulation environment).
8. Randomly select a point within the spherical region to place a stem cell such that it touches the inner boundary of the spherical region.

9. Write the initial state of the environment (0th iteration) and the environmental statistics to the respective files.
10. Set iteration counter "i" to 1.
11. Repeat steps 12-15 while counter "i" is less than maxIteration.
 12. Randomly select one simulation operation (moving the modeled cells, updating E-cadherin / β - Catenin or evolving the genetic code of the cells).
 13. Increase the age of each cell and perform the cell division, differentiation or death as required.
 14. Write the current state of the environment (i.e. state after "i" simulation iterations) and the environmental statistics to the respective files.
 15. Increment the value of counter "i".
16. Free any dynamically allocated memory. (e.g. memory allocated for the environment).
17. Exit.

3.5. Major Operations

1. Setting up Extra Cellular Matrix
2. Determination of favorable location
3. Move Cells
4. Update E-Cadherin / β -Catenin (EB)
5. Cell Division
6. Save the state of environment

3.5.1. Setting up Extra Cellular Matrix

This operation is performed to initialize the simulation environment with the given number of ECM (Extra-Cellular Matrix) fibers (as specified in the configuration file), position them at random locations with random orientations in the environment (lattice).

Following is the sequence of steps performed in this operation:

1. Create a lattice of given dimensions.
2. Repeat following steps(3-6) ***fiberCount*** number of times :
 3. Randomly select a point (P) from within the lattice.
 4. Randomly generate a latitude and a longitude value.
 5. Create a fiber of length ***fiberLength*** originating from point P, inclined at angles according to latitude and longitude values.
 6. Assign a unique ID to newly created fiber.
7. Count the number of fibers passing through each point within the lattice and generate frequency map of it.

3.5.2. Determination of favorable location

This operation is performed to identify a location in the neighbourhood of the given cell(i.e. within its sensing radius) where that cell can be moved or a new

daughter cell can be added. The probability of selecting a location is defined by the Gaussian probability distribution.

Thus, this operation is performed before adding a new daughter cell or moving the given cell.

For this operation we have assumed that every cell has a centroid associated with it. So, while initializing the cell, we select a random lattice point (where no cell or ECM fiber is present) as the centroid of that cell.

Following is the sequence of steps performed in this operation:

1. Get the centroid of the parent cell.
2. Identify the neighbour fibers (inside the region of interest) within the sensing radius from the centroid, count them, and calculate the mean with respect to the number of neighbour points.
3. For each neighbouring point in the region of interest perform steps 4-6
 4. Determine the Gaussian probability “ p ” of selecting it as the favorable point.
 5. Generate a random number “ r ” between 0 and 1.
 6. If $r > p$ select this point and go to step 7.
7. If a point was found return it, else return null.

3.5.3. Move Cells

This operation is performed to move the given cell to an appropriate location in its neighbourhood.

Following is the sequence of steps performed in this operation:

1. For each cell present in the environment, repeat steps 2-4
2. Determine a favorable location to move the given cell.
3. Remove the cell from the current position in the lattice (grid).
4. Put the cell on the new position in the lattice.

3.5.4. Update E-Cadherin / β -Catenin (EB)

E-Cadherin/ β -Catenin is a protein that is responsible for cell-cell adhesion. It is also used in signaling pathways. This operation is performed to update the E-Cadherin / β -Catenin value of the cells according to the change in the neighborhood.

Following is the sequence of steps performed in this operation:

1. For each cell present in the environment, repeat steps 2 – 3
2. Calculate the total number of ECM Fibers and total E Cadherin (EC) present in the neighborhood of the cell.
3. Update EB as

$$EB = \frac{\text{sumFiber}}{\text{sumFiber} + k} + \frac{\text{totalNeighbourEC}}{N}$$

Where N is the total number of fibers and k is a constant.

3.5.5. Cell Division

3.5.5.1. *Stem Cells*

Two types of cell division occur in stem cells: Symmetric division and asymmetric division.

The following sequence of steps occurs:

1. Get a favorable location where the daughter cell will be generated (in the neighborhood of the parent)
2. Generate a random number r between 0 and 1.
3. If $r < \alpha$ (The probability of symmetric division)

Create a new daughter stem cell located at the favorable location determined in step 1. The properties of the daughter will be identical to the properties of the parent.

else

Create a new daughter Transit Amplifying cell located at the favorable location determined in step 1. The properties of the daughter will be identical to the properties of the parent.

4. Reset the age of parent and daughter to zero.

3.5.5.2. *Transit Amplifying Cells*

Only symmetric division occurs inside transit amplifying cells. The following is the sequence of steps that occur:

1. Get a favorable location where the daughter cell will be generated (in the neighborhood of the parent).
2. Create a new daughter Transit Amplifying cell located at the favorable location determined in step 1. The properties of the daughter will be identical to the properties of the parent.

3. Increment the value of beta (the number of times division has occurred also known as the rounds of amplification) for both the cells.
4. Reset the age of parent and daughter to 0.

3.5.6. Save the state of the environment

This operation saves the complete state of the simulation environment in a csv file. This operation is executed at the end of each iteration of simulation. The file contains data corresponding to each point in the simulation environment.

There is a switch to select the type of information that needs to be stored. The following information is supported:

1. Type: The information regarding the type of each point in the simulation environment is stored as follows:

Type, x, y, z

where x, y and z are the x, y and z coordinates of the point. The following are the values for types:

- a. EMPTY: Nothing present at that point.
- b. ECM: ECM fiber present at that point.
- c. CELL: A Terminally Differentiated Cell present at that point.
- d. TA_CELL: A Transit Amplifying Cell present at that point.
- e. STEM_CELL: A Stem Cell present at that point.

2. Count: When this switch is enabled, the number of ECM fibers present at each point in the simulation environment are stored as follows:

Count, x, y, z

where x, y and z are the x, y and z coordinates of the point. A value of 0 in count denotes that no ECM fiber is present at that location, hence that location is either empty or contains some cell.

3. Id: When this switch is enabled, the id of the cell present at each point in the simulation environment is stored as follows:

Id, x, y, z

If no cell is present then 0 is stored at that location.

3.5.7. TA Cell differentiation

This operation simulates the differentiation mechanism in transit amplifying cells. The following steps occur:

1. Create a new Terminally Differentiated Cell a.k.a. Normal Cell.
2. Set its centroid, radius, sensing radius, E-Cadherin , E-Cadherin / β -Catenin and genetic code values to be equal to the TA Cell being differentiated.
3. Set the age of TDC to zero.
4. Change the state of the environment such that the TD Cell is now present at the location where the TA Cell was originally present.
5. Remove the TA Cell from the environment.

3.5.8. Evolve Genetic Code

The genetic information inside each cell is stored in a genetic code which evolves with each iteration according to the following diagram (26).

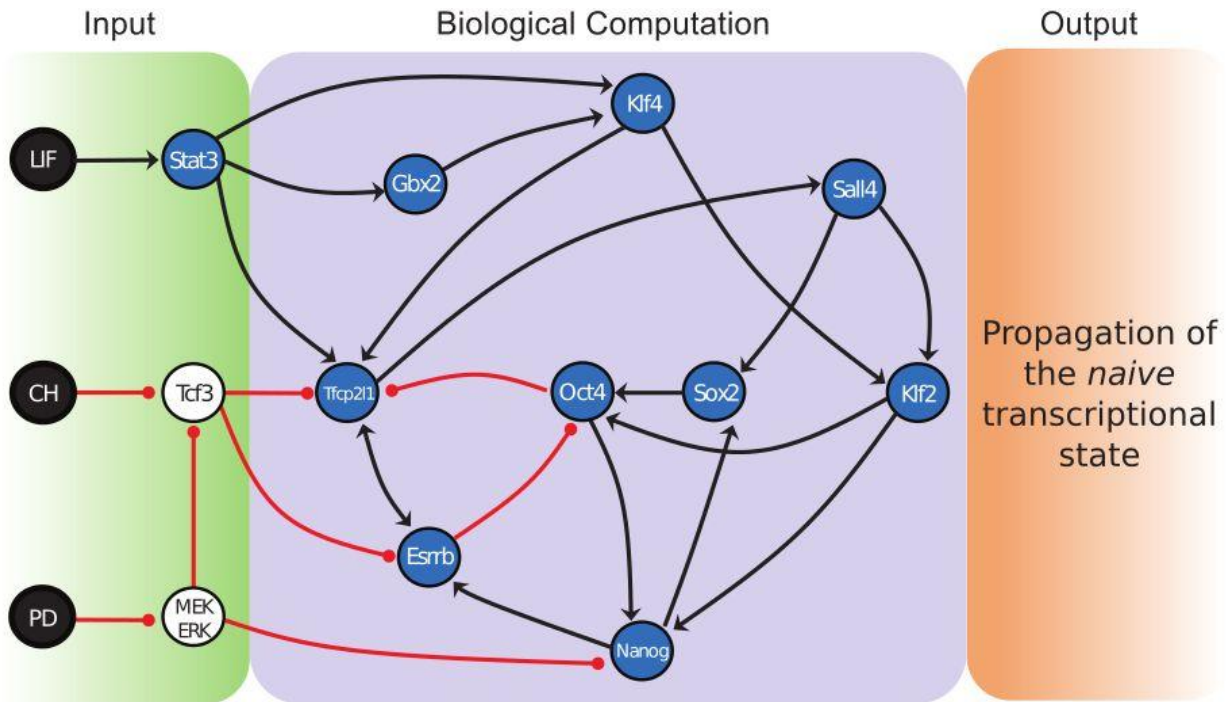


Figure 9: The Boolean model consisting of 16 interactions and 12 components. Image taken from (26).

The model breaks down to the following basic logical computations:

$$\text{STAT3} = \text{LIF}$$

$$\text{TCF3} = \text{CH}' \cdot \text{MEK_ERK}$$

$$\text{MEK_ERK} = \text{PD}'$$

$$\text{GBX2} = \text{STAT3}$$

$$\text{KLF4} = \text{STAT3} \cdot \text{GBX2}$$

$$\text{TFCP2L1} = \text{ESRRB} \cdot \text{TCF3}' \cdot \text{STAT3} \cdot \text{KLF4} \cdot \text{OCT4}'$$

$$\text{ESRRB} = \text{TFCP2L1} \cdot \text{TCF3}' \cdot \text{NANOG}$$

$$\text{OCT4} = \text{SOX2} \cdot \text{KLF2} \cdot \text{ESRRB}'$$

$$\text{SOX2} = \text{SALL4} \cdot \text{NANOG}$$

$$\text{NANOG} = \text{MEK_ERK}' \cdot \text{OCT4} \cdot \text{KLF2}$$

$$\text{KLF2} = \text{KLF4} \cdot \text{SALL4}$$

$SALL4 = TFCP2L1$

where \cdot denotes AND operation and $'$ denotes the NOT operation.

Following is the sequence of steps performed:

1. For each cell present in the environment perform the above mentioned computations using values from the previous iteration.

3.5.9. Cell Death

This operation is performed to remove the Terminally Differentiated cells from the environment when they reach a particular age (i.e. the value of gamma in our case) leaving behind the empty space.

Following is the sequence of steps performed in this operation:

1. For each Terminally Differentiated cell present in the environment, repeat
step 2
2. If the cell age is equal to gamma, remove that cell from the environment
and mark its location as empty.

User Interface Screenshots

4. User Interface Screenshots

4.1. User Interface

We have two types of interfaces: CLI (command-line interface) and GUI (graphical user interface)

4.1.1. CLI

A command-line interface (27) or command language interpreter (CLI), also known as command-line user interface, is a means of interacting with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines). Command-line interfaces to computer operating systems are less widely used by casual computer users, who favor graphical user interfaces. Command-line interfaces are often preferred by more advanced computer users, as they often provide a more concise and powerful means to control a program or operating system.

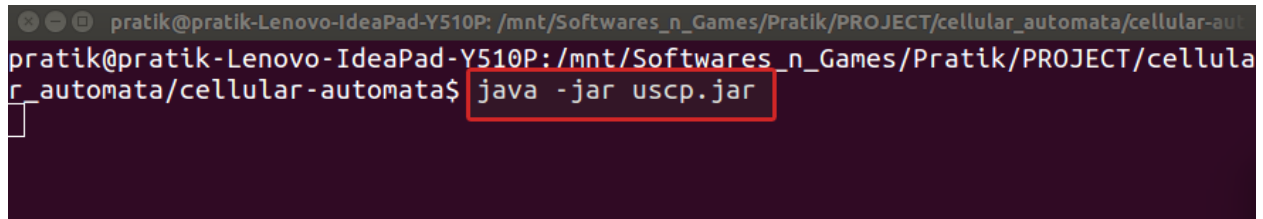
Programs with command-line interfaces are generally easier to automate via scripting.

4.1.2. GUI

In computing, a graphical user interface (GUI) (28) is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), which require commands to be typed on the keyboard.

4.2. Program UI Screenshots

1. Executing the program using terminal (CLI)



```
pratik@pratik-Lenovo-IdeaPad-Y510P: /mnt/Softwares_n_Games/Pratik/PROJECT/cellular_automata/cellular-aut  
r_automata/cellular-automata$ java -jar uscp.jar
```

Figure 10: How to execute the program using command-line interface

2. Startup Screen (GUI)

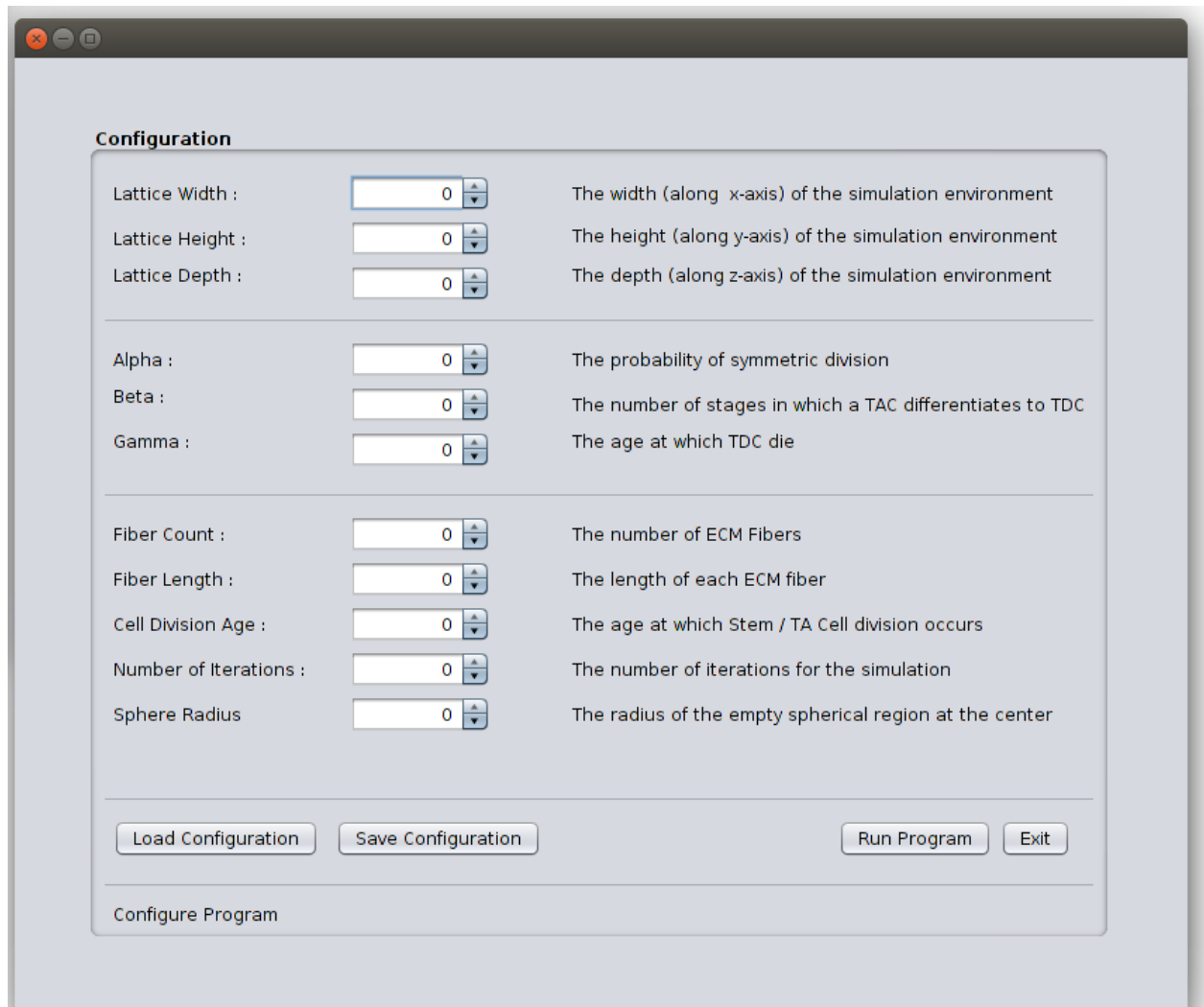


Figure 11: GUI for setting the simulation parameters

3. Loading Configuration File (Dialog Box)

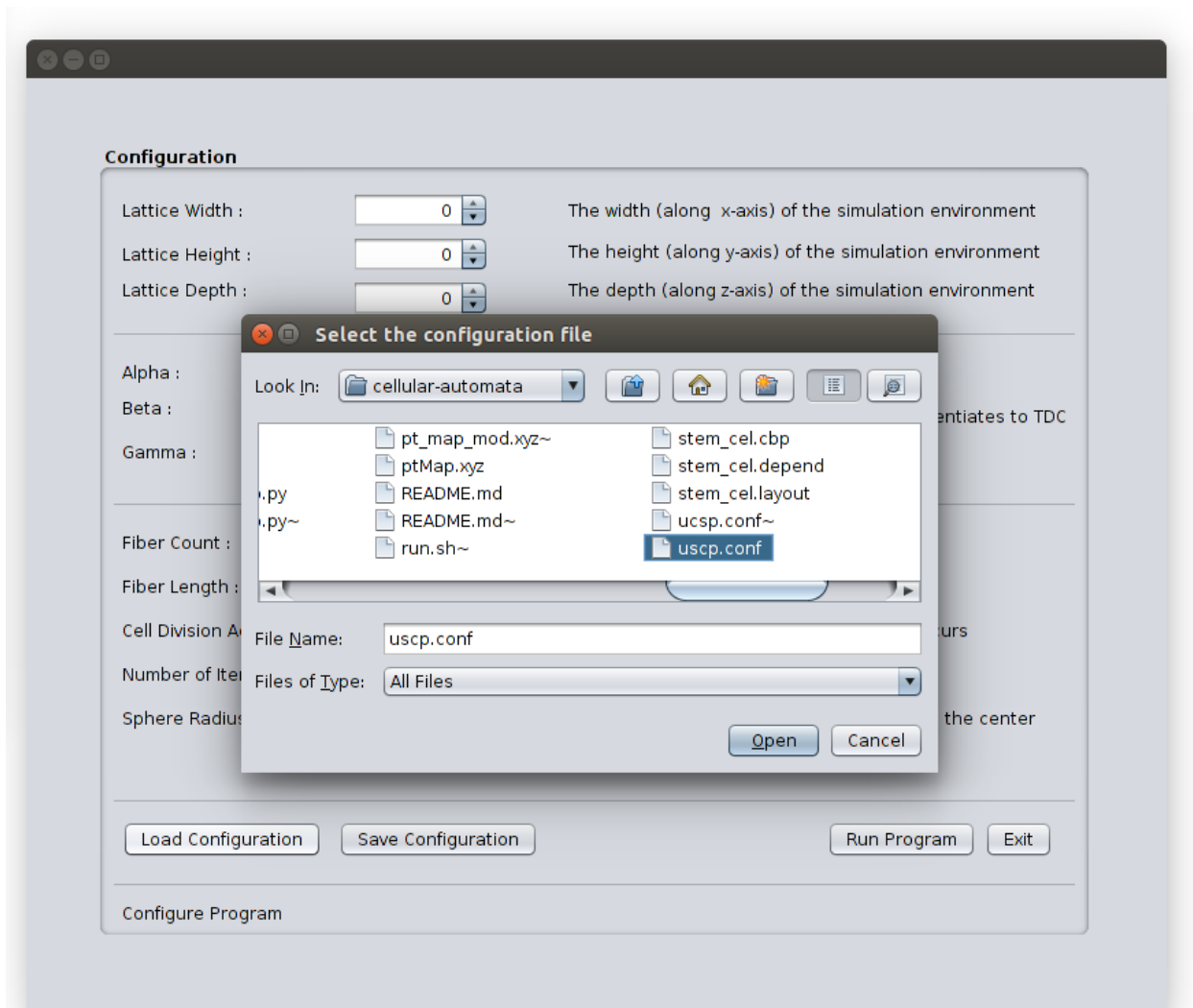


Figure 12: Dialog box for loading configuration file

We can load previously saved configuration.

4. Updated values after loading configuration

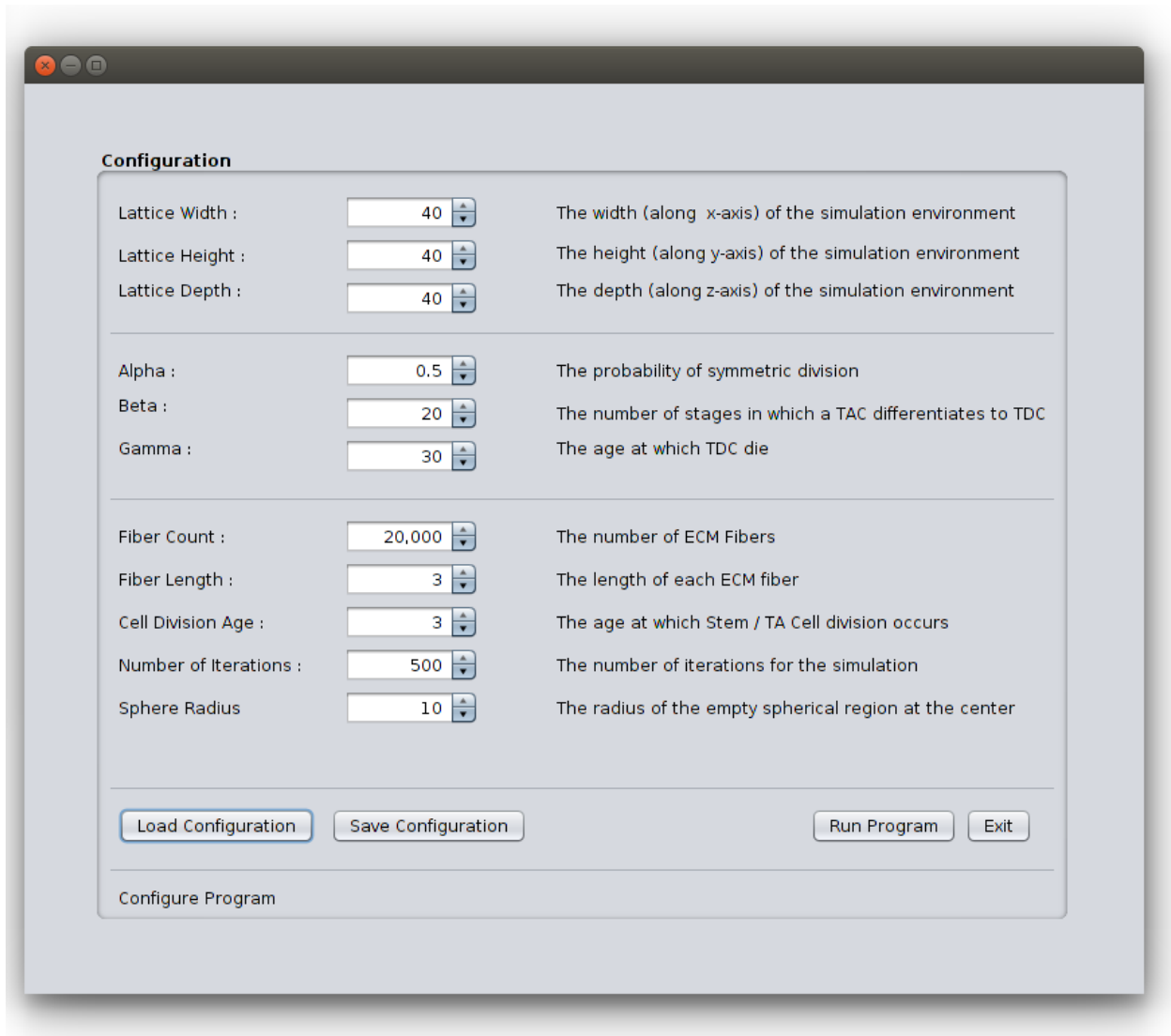


Figure 13: Sample simulation parameters loaded from the configuration file

These values can be easily modified using the spinner buttons or manually entering the desired values.

5. Save Configuration File

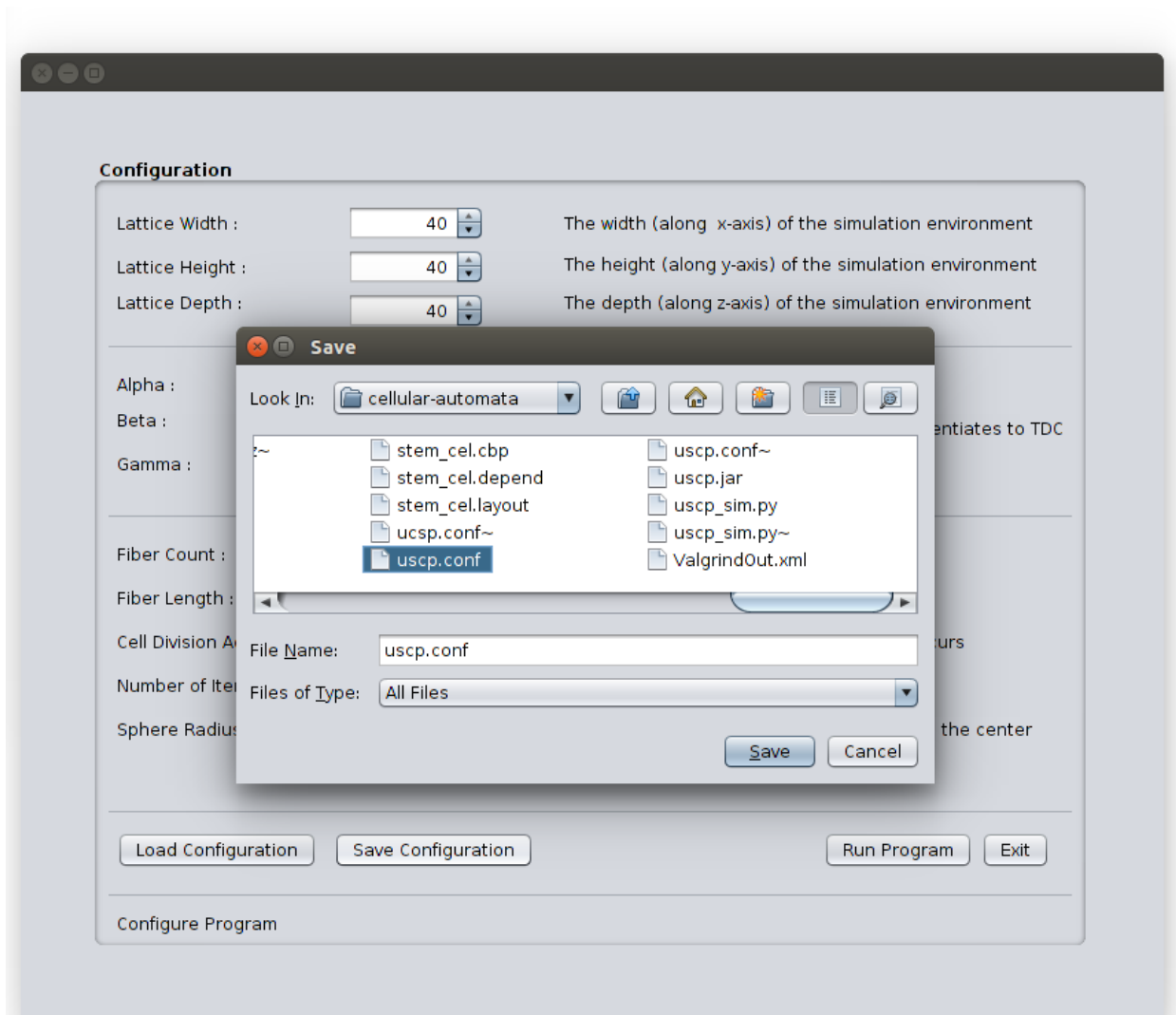
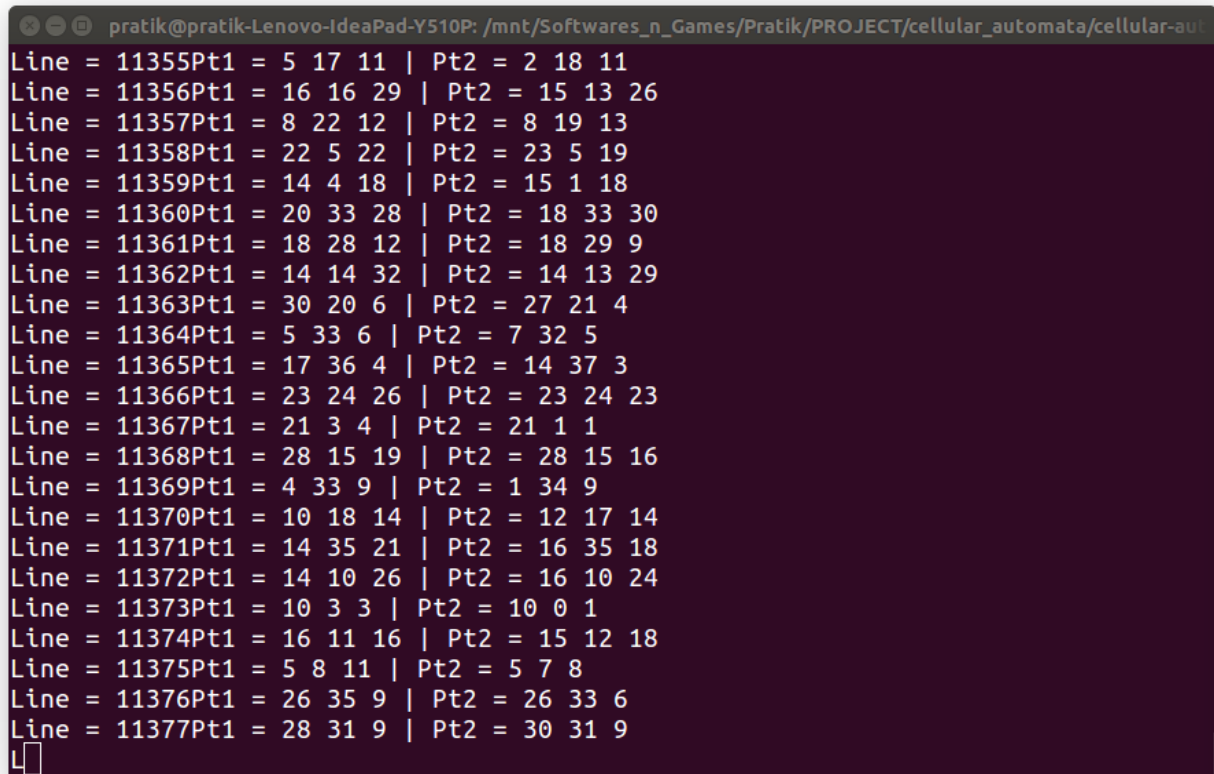


Figure 14: Dialog box for saving configuration file

We can save our configuration for future use.

6. Program Status (while generating ECM fibers)

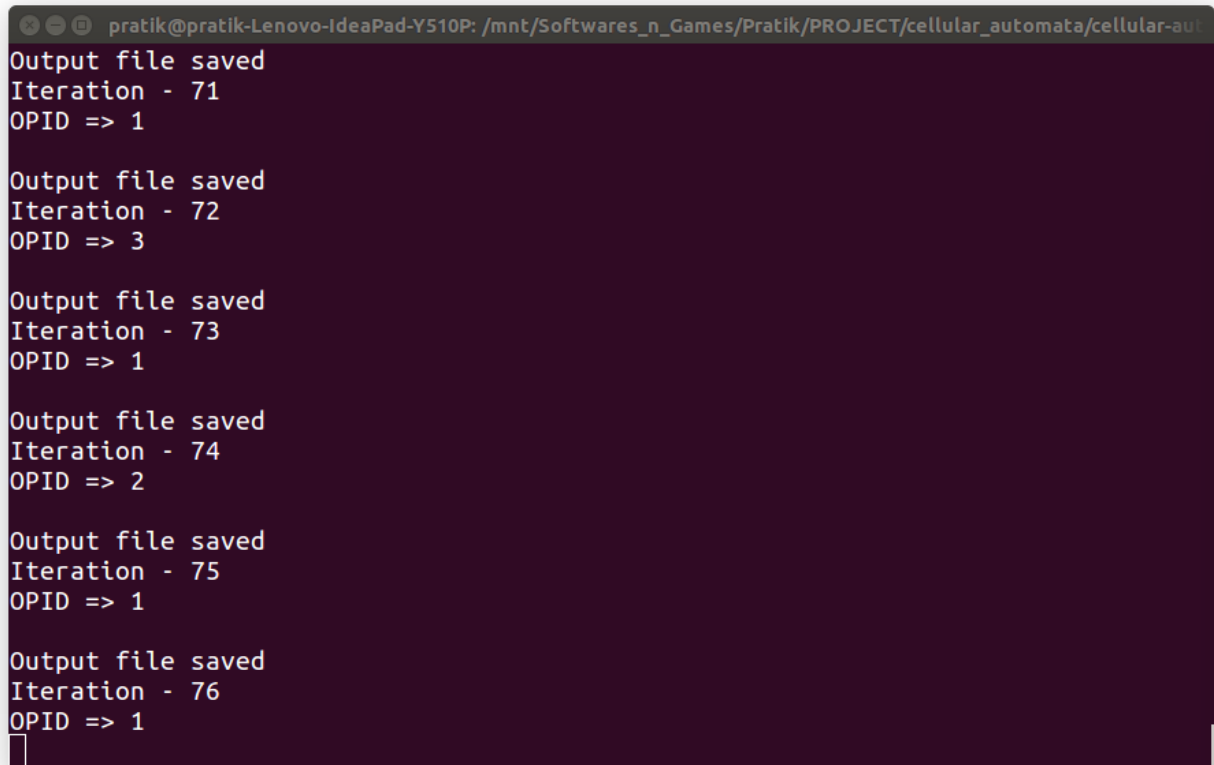


```
pratik@pratik-Lenovo-IdeaPad-Y510P: /mnt/Softwares_n_Games/Pratik/PROJECT/cellular_automata/cellular-aut
Line = 11355Pt1 = 5 17 11 | Pt2 = 2 18 11
Line = 11356Pt1 = 16 16 29 | Pt2 = 15 13 26
Line = 11357Pt1 = 8 22 12 | Pt2 = 8 19 13
Line = 11358Pt1 = 22 5 22 | Pt2 = 23 5 19
Line = 11359Pt1 = 14 4 18 | Pt2 = 15 1 18
Line = 11360Pt1 = 20 33 28 | Pt2 = 18 33 30
Line = 11361Pt1 = 18 28 12 | Pt2 = 18 29 9
Line = 11362Pt1 = 14 14 32 | Pt2 = 14 13 29
Line = 11363Pt1 = 30 20 6 | Pt2 = 27 21 4
Line = 11364Pt1 = 5 33 6 | Pt2 = 7 32 5
Line = 11365Pt1 = 17 36 4 | Pt2 = 14 37 3
Line = 11366Pt1 = 23 24 26 | Pt2 = 23 24 23
Line = 11367Pt1 = 21 3 4 | Pt2 = 21 1 1
Line = 11368Pt1 = 28 15 19 | Pt2 = 28 15 16
Line = 11369Pt1 = 4 33 9 | Pt2 = 1 34 9
Line = 11370Pt1 = 10 18 14 | Pt2 = 12 17 14
Line = 11371Pt1 = 14 35 21 | Pt2 = 16 35 18
Line = 11372Pt1 = 14 10 26 | Pt2 = 16 10 24
Line = 11373Pt1 = 10 3 3 | Pt2 = 10 0 1
Line = 11374Pt1 = 16 11 16 | Pt2 = 15 12 18
Line = 11375Pt1 = 5 8 11 | Pt2 = 5 7 8
Line = 11376Pt1 = 26 35 9 | Pt2 = 26 33 6
Line = 11377Pt1 = 28 31 9 | Pt2 = 30 31 9
L
```

Figure 15: Screenshot taken while setting up the environment

Here, the value of Line is the ID of the ECM Fiber, Pt1 and Pt2 are the (x y z) coordinates of the endpoints of that line. These points are the random locations within our environment.

7. Program Status (while executing 76th iteration)



```
pratik@pratik-Lenovo-IdeaPad-Y510P: /mnt/Softwares_n_Games/Pratik/PROJECT/cellular_automata/cellular-aut
Output file saved
Iteration - 71
OPID => 1

Output file saved
Iteration - 72
OPID => 3

Output file saved
Iteration - 73
OPID => 1

Output file saved
Iteration - 74
OPID => 2

Output file saved
Iteration - 75
OPID => 1

Output file saved
Iteration - 76
OPID => 1
```

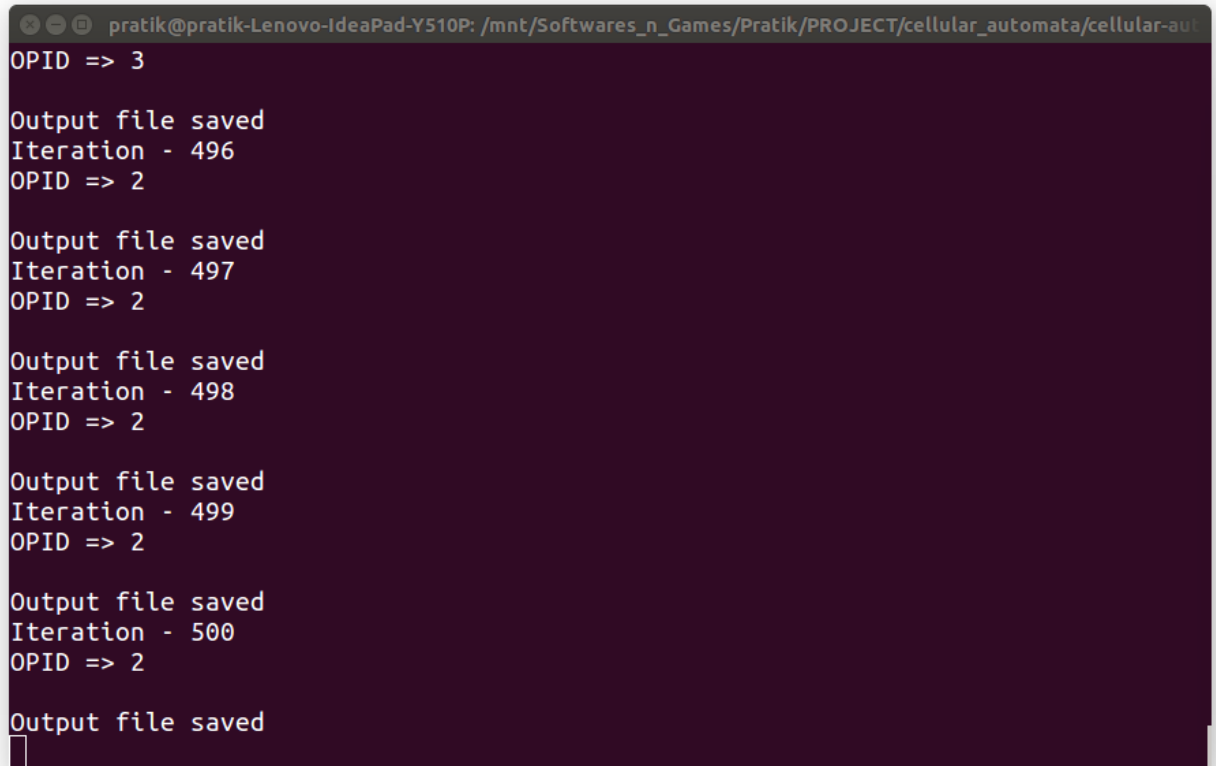
Figure 16: Screenshot taken while executing 76th iteration

Here OPID signifies which operation we are performing.

Following are the operation IDs (OPID):

- 1) Move cells
- 2) Update E-Cadherin / β -Catenin value
- 3) Evolve the genetic code of the cells

8. Program Status (on completing 500 iterations)

A screenshot of a terminal window with a dark purple background and white text. The terminal shows the output of a program after 500 iterations. The text is as follows:

```
pratik@pratik-Lenovo-IdeaPad-Y510P: /mnt/Softwares_n_Games/Pratik/PROJECT/cellular_automata/cellular-aut
OPID => 3
Output file saved
Iteration - 496
OPID => 2
Output file saved
Iteration - 497
OPID => 2
Output file saved
Iteration - 498
OPID => 2
Output file saved
Iteration - 499
OPID => 2
Output file saved
Iteration - 500
OPID => 2
Output file saved
```

The terminal window has a title bar at the top with three window control icons and the text 'pratik@pratik-Lenovo-IdeaPad-Y510P: /mnt/Softwares_n_Games/Pratik/PROJECT/cellular_automata/cellular-aut'. The output shows a repeating pattern of 'Output file saved', 'Iteration - [number]', and 'OPID => [number]' for iterations 496 through 500. The iteration numbers increase by 1 in each cycle, and the OPID values are 3 for the first iteration shown and 2 for the subsequent ones.

Figure 17: Screenshot taken after completing 500 iterations

Results

5. Results

5.1. Simulation Plots

In a sample execution, we generated 5 plots.

The following plot shows the total number of cells as a function of time (iteration):

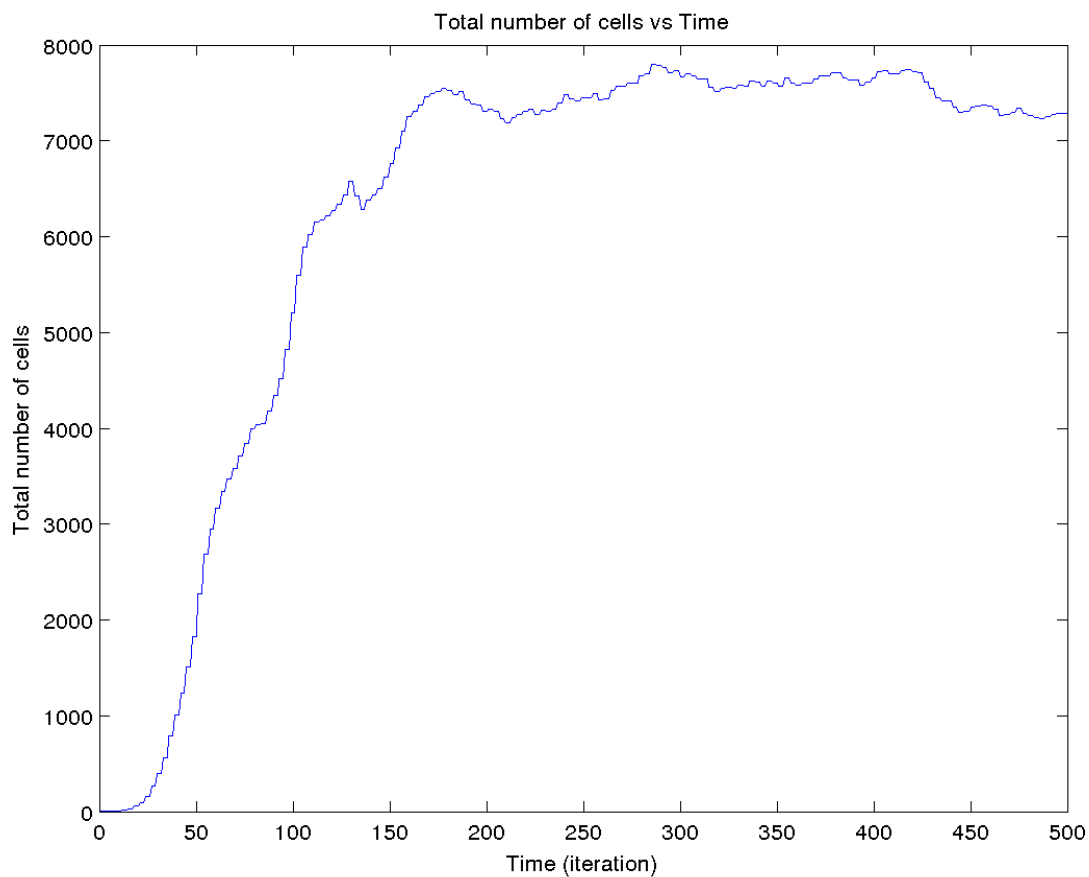


Figure 18 : Total number of cells vs Time

As we can see that the number of cells increases rapidly in the initial stages and then becomes stable.

The following plot shows the number of stem cells as a function of time (iteration):

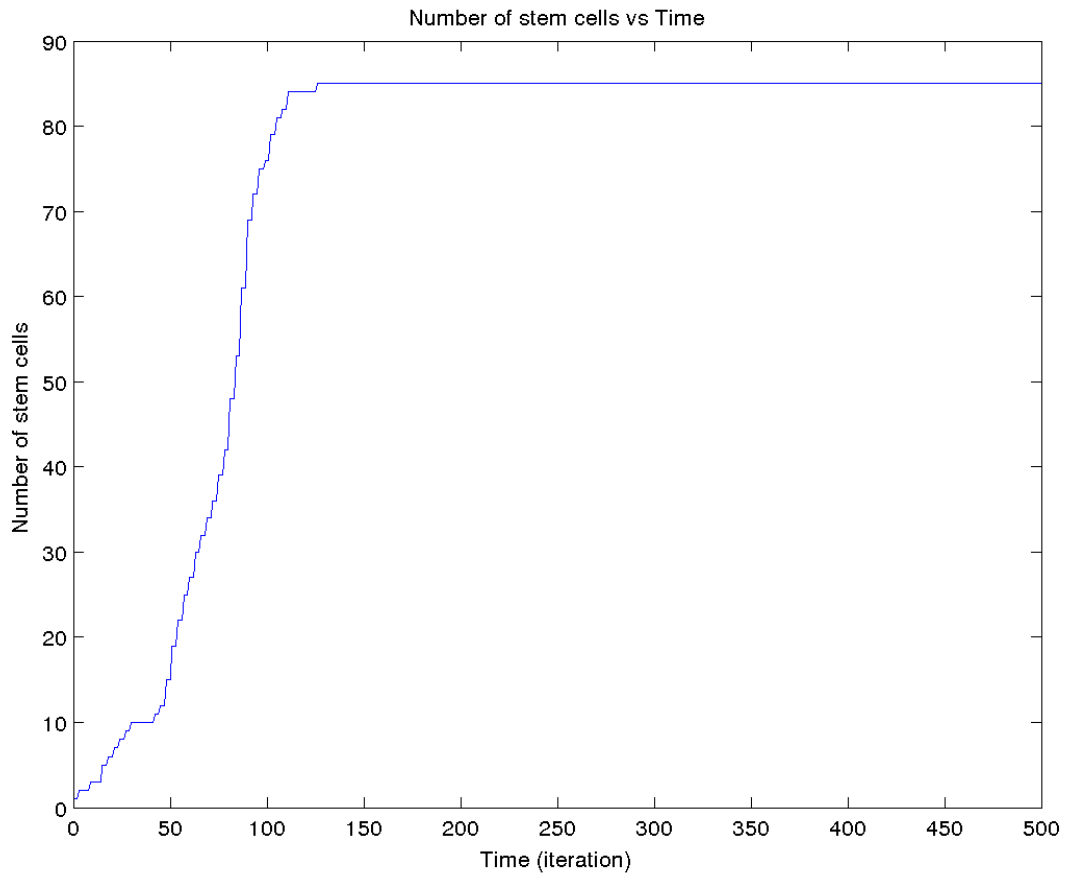


Figure 19 : Number of stem cells vs Time

Here we can see that initially the number of stem cells increases but later their rate of growth becomes very slow and thus they increase very slowly.

The following plot shows the number of transit amplifying cells as a function of time (iteration):

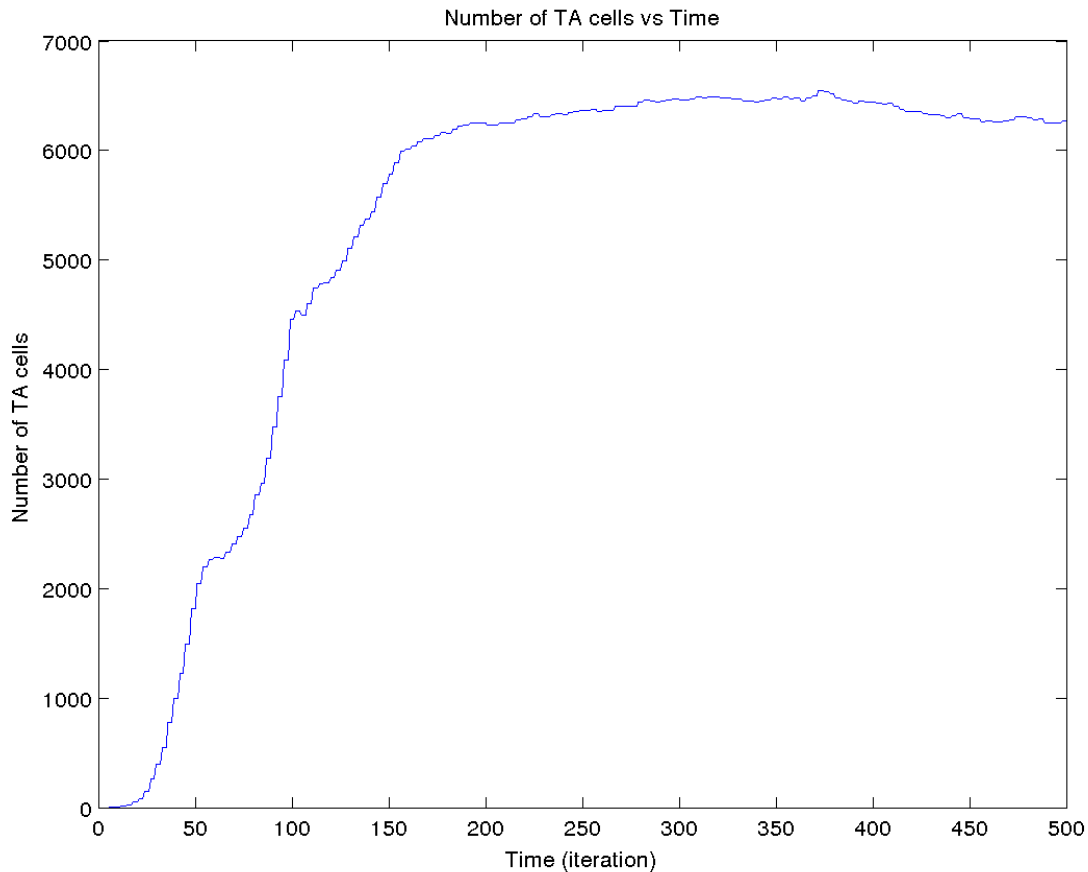


Figure 20 : Number of transit amplifying cells vs Time

Transit Amplifying Cells are the cells that occur maximum in number because they always divide into two transit cells and are also produced as a result of asymmetric division in stem cells. There is also decrease in their numbers due to the fact that they differentiate into Terminally Differentiated Cells.

The following plot shows the number of Terminally Differentiated (a.k.a. Normal) cells as a function of time (iteration):

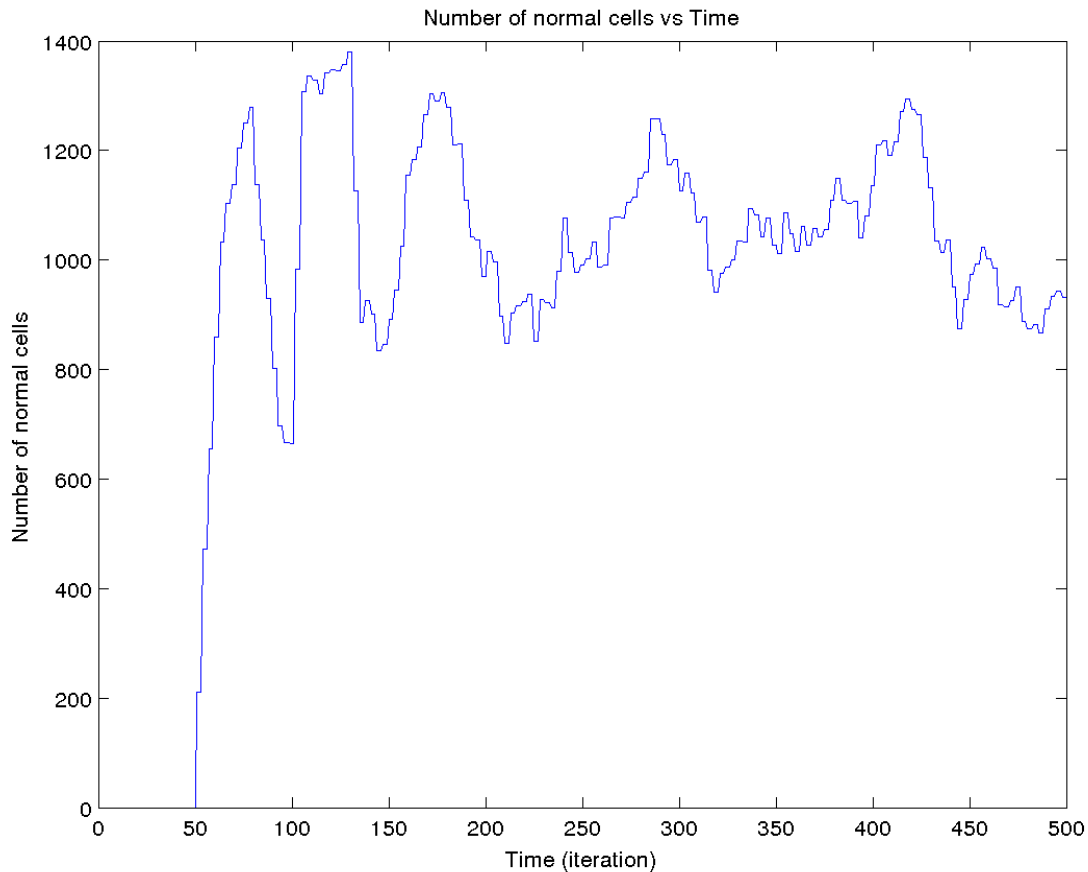


Figure 21 : Number of terminally differentiated (normal) cells vs Time

The number of these cells varies by the highest amount compared to other cells. This is due to the fact that these cells die once they reach a certain age and they are produced in groups when multiple Transit Amplifying Cells differentiate into Terminally Differentiated Cells.

The following plot shows the number of ECM fibers as a function of time (iteration):

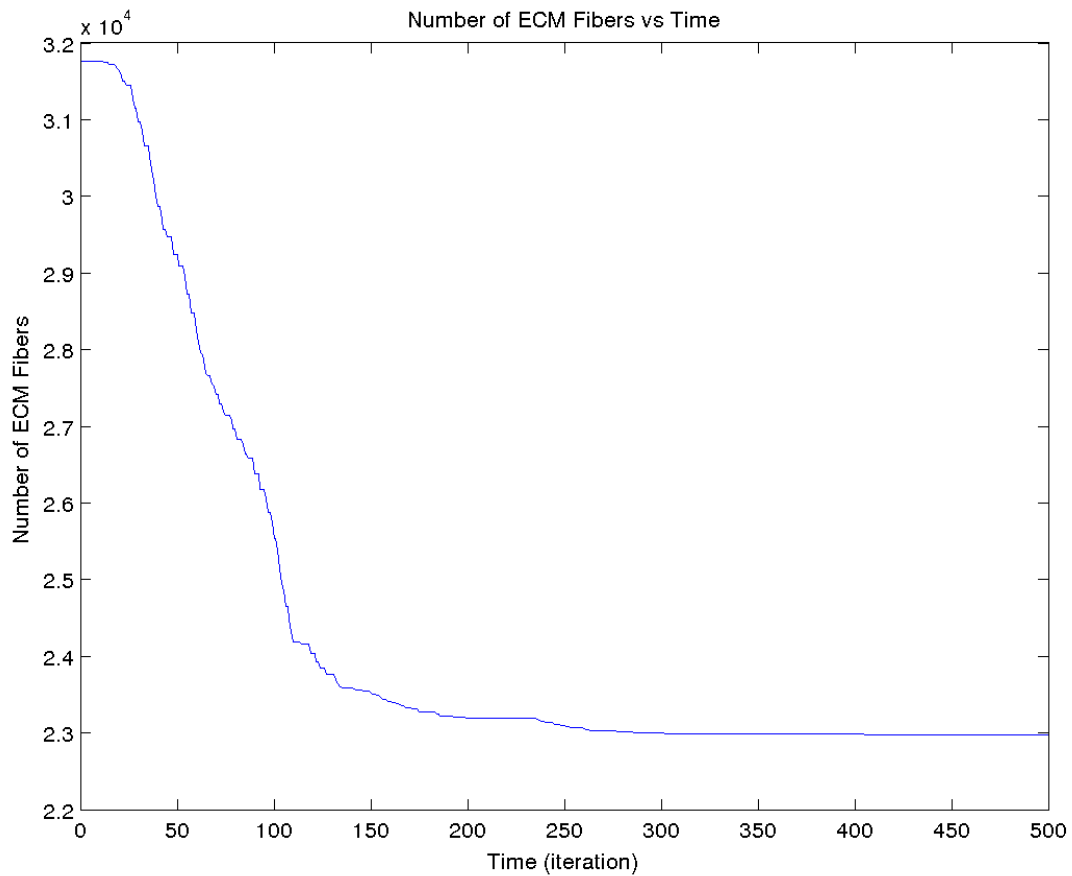


Figure 22 : Number of ECM fibers vs Time

Here we can see that the number of ECM fibers decreases with time. This is because cells consume ECM when they move or when new daughters are created. Since the number of cells grows rapidly in the initial stages, the rate of ECM consumption is also very high.

5.2. Visualization Screenshots

We start our simulation a 3D-grid (environment) given number of ECM fibers placed randomly within the environment. Then we clear the ECM fibers from the spherical region of given radius (with centre of the sphere as the centre of the simulation environment) and place a stem cell randomly within the spherical region such that the stem cell touches it's the inner boundary.

Figure below shows an example of the initial setup.

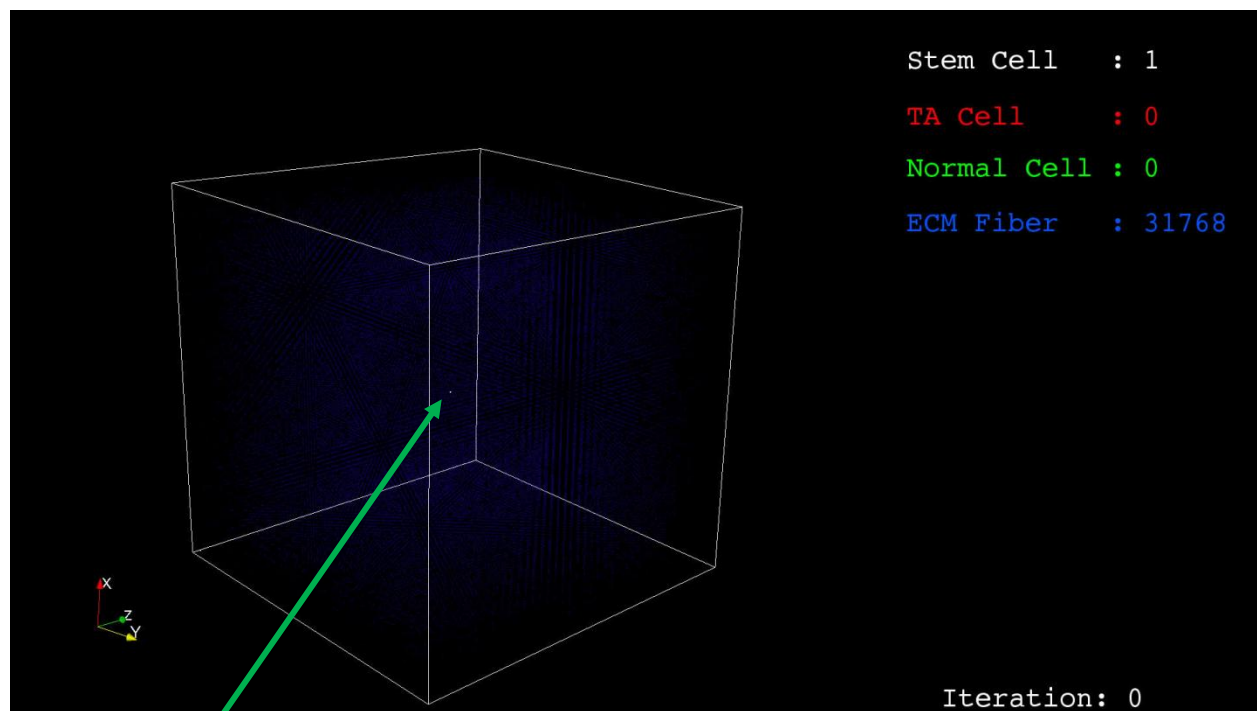


Figure 23 : Initial Setup

Stem Cell

After 100 iterations, the environment looked like:

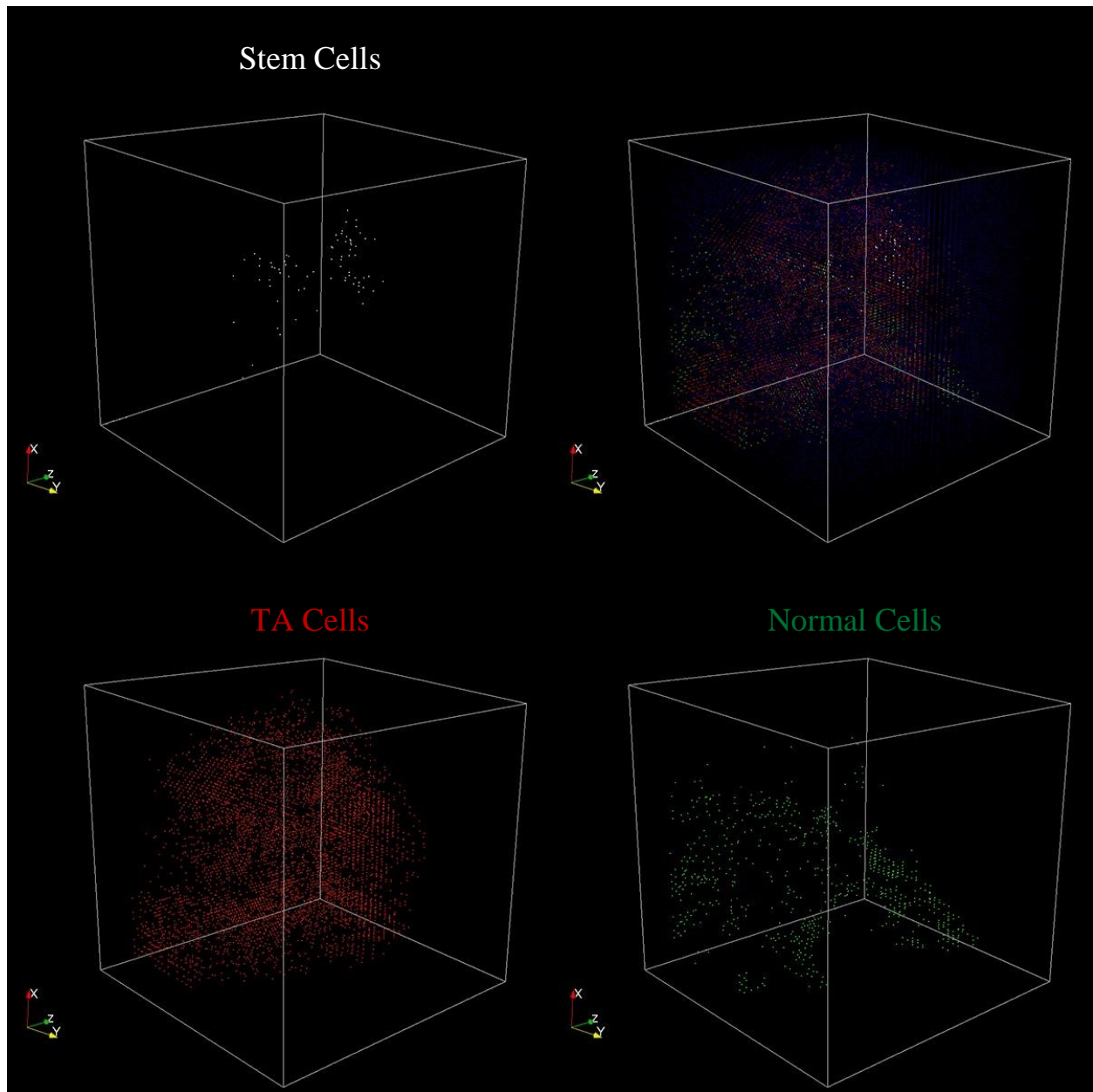


Figure 24 : Environment after 100 iterations

The count of these cells and fibers were:

- | | | | |
|--------------|--------|----------------|---------|
| • Stem Cells | : 76 | • Normal Cells | : 664 |
| • TA Cells | : 4461 | • ECM Fibers | : 25537 |

After 500 iterations, the environment looked like:

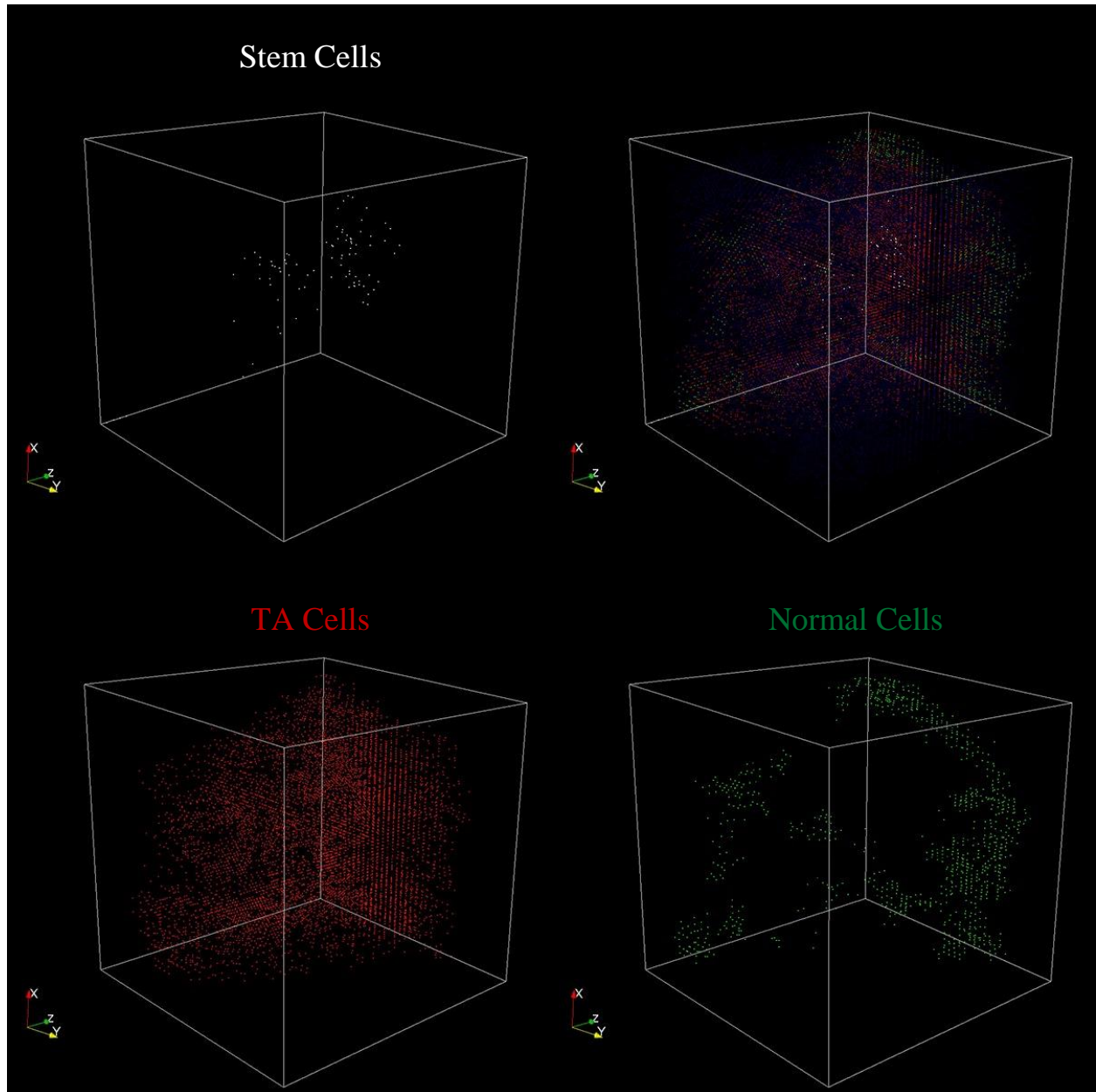


Figure 25 : Environment after 100 iterations

The count of these cells and fibers were:

- | | | | |
|--------------|--------|----------------|---------|
| • Stem Cells | : 85 | • Normal Cells | : 932 |
| • TA Cells | : 6268 | • ECM Fibers | : 22968 |

This example shows the positions of Stem Cells (top left grid), TA Cells (bottom left grid), and Normal Cells (bottom right grid) separately as well as the complete environment containing Cells and ECM fibers (top right grid).

After running the simulation several times we found that initially the number of stem cells increases quickly (starting from a single stem cell) and after few iterations their rate of increment becomes very low. Also, the ECM fibers keeps decreasing (as shown in the example, ECM fibers reduces from 31768 to 25537 in 100 iterations and then to 22968 in 500 iterations) as the cells consumes them.

It is also observed that the TA Cells are maximum in number, followed by the normal cells and then the stem cells.

Conclusion and Future work

6. Conclusion and Future Work

6.1. Conclusion

In this work we have presented a cellular automata based discrete model to understand the life cycle dynamics of stem cells. The presented 3D cellular automata based model is constructed using rule based approach and takes into account the asymmetric vs symmetric division, cell-differentiation etc. In this model, all cells are essentially divided into three type of cells viz. 1. stem cells, 2. transiently amplifying cells and 3. terminally differentiated cells. The model was used to predict the population dynamics of different type of cells. Results of the simulation done for one set of parameters suggest that, while stem cells and transiently amplifying cells have saturating profile, terminally differentiated cells have oscillating profiles.

6.2. Future Work

We aim to use this model to understand the biological developmental processes i.e. how does a single stem cell give rise to variety of differentiated cells. We also aim to augment the present model with other aspects like oxygen dynamics and cell migration. Studying the role of different model parameters on population dynamics is the main objective of the developed model.

References

1. **Fogel, Gary B. and Corne, David W.** An Introduction to BioInformatics for Computer Scientists. [book auth.] Gary B. Fogel, David W. Corne and Morgan Kaufmann. *Evolutionary computation in bioinformatics*. 2003, pp. 3-18.
2. *Bioinformatics---an introduction for computer scientists*. **Cohen, Jacques**. 2004, ACM Computing Surveys, pp. 122-158.
3. **Cooper, Geoffrey M and Hausman, Robert E.** *The Cell: A molecular approach*. 2007.
4. **Alberts, Bruce, et al.** *Molecular Biology of the Cell (4th ed)*. 2002.
5. *Growth factors in the extracellular matrix*. **Taipale, J and Keski-Oja, J.** 1997, The FASEB Journal, pp. 51-59.
6. **O'Connor, Clare M. and Adams, Jill U.** *Essentials of Cell Biology*. Cambridge, MA : NPG Education, 2010.
7. *Out of Eden: Stem Cells and Their Niches*. **Watt, Fiona M.** 5457, s.l. : Science, 2000, Vol. 287, pp. 1427-1430.
8. *Collective dynamics of stem cell populations*. **MacArthur, Ben D.** 10, s.l. : Proceedings of the National Academy of Sciences of the United States of America, 2014, Vol. 111, pp. 3653-3654.
9. *Stem cells: the generation and maintenance of cellular diversity*. **Hall, Peter A. and Watt, Fiona M.** 4, s.l. : Development, 1989, Vol. 106, pp. 619-633.
10. *Microenvironmental Variables Must Influence Intrinsic Phenotypic Parameters of Cancer Stem Cells to Affect Tumourigenicity*. **Scott, Jacob G., et al.** 1, 2014, Plos Computational biology, Vol. 10.
11. **Shiffman, Daniel.** Cellular Automata. *The Nature of Code: Simulating Natural Systems with Processing*. 2012.
12. **Coppin, Ben.** Artificial Life: Learning through Emergent Behavior. *Artificial Intelligence Illuminated*. s.l. : Jones & Bartlett Learning, 2004.

13. C++ - Wikipedia, the free encyclopedia. [Online] [Cited: May 6, 2015.] <http://en.wikipedia.org/wiki/C%2B%2B>.
14. Java (programming language) - Wikipedia, the free encyclopedia. [Online] [Cited: May 6, 2015.] http://en.wikipedia.org/wiki/Java_%28programming_language%29.
15. Python (programming language) - Wikipedia, the free encyclopedia. [Online] [Cited: May 6, 2015.] http://en.wikipedia.org/wiki/Python_%28programming_language%29.
16. Doxygen: Main Page. [Online] www.doxygen.org.
17. Code::Blocks. [Online] <http://www.codeblocks.org/>.
18. GCC, the GNU Compiler Collection. [Online] <https://gcc.gnu.org/>.
19. libxml++ - An XML Parser for C++. [Online] <http://libxmlplusplus.sourceforge.net/docs/manual/html/>.
20. Paraview. [Online] <http://www.paraview.org/Wiki/ParaView>.
21. Bitbucket Documentation Home. [Online] <https://confluence.atlassian.com/display/BITBUCKET/>.
22. Git. [Online] <http://git-scm.com/>.
23. Matlab Documentation - MathWorks India. [Online] [Cited: May 6, 2015.] <http://in.mathworks.com/help/matlab/>.
24. Features - MATLAB - MathWorks India. [Online] [Cited: May 6, 2015.] <http://in.mathworks.com/products/matlab/features.html>.
25. NetBeans IDE - Overview. [Online] <https://netbeans.org/features/index.html>.
26. *Defining an essential transcription factor program for naïve pluripotency.* **Dunn, S-J, et al.** New York, N.Y. : Science , 2014, Science, Vol. 344, pp. 1156-60.
27. Command-line interface - Wikipedia, the free encyclopedia. [Online] May 6, 2015. http://en.wikipedia.org/wiki/Command-line_interface.

28. Graphical user interface - Wikipedia, the free encyclopedia. [Online] May 6, 2015. http://en.wikipedia.org/wiki/Graphical_user_interface.