The University of Sheffield

Development of an Agent-based Model Capturing Cellular Interactions Associated with Heart Attack

Harrison Paul Cooper

Supervised By:
Dr. Dawn Walker

COM3610

<Date>

This report is submitted in partial fulfilment of the requirement for the degree of MComp Computer Science with a Year in Industry by Harrison P. Cooper

Signed Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Harrison Paul Cooper

Signature:

Date:

Abstract

Ageing is believed to be the largest contributor to the deterioration of the wall lining the inside of our blood vessels and is dictated by a series of rules which produce emergent behaviours between cells. Agent based models have been used in the past to great success in accurately modelling interactions between cells and one has been adapted for this project.
Migration of endothelial cells plays a crucial role in the healing of damaged endothelium walls, and as we get older this migration is thought to be hindered by the increased presence of larger senescent cells. The model aims to provide insight into the rate of wound closure with age and the results support the theory that senescent cells slow down surrounding cells.

Acknowledgements

I would like to take the time to give my thanks to Dr. Dawn Walker for her continued encouragement and expert advice throughout this challenging project.
I would also like to give thanks to Prof. Paul Evans, my parents, and my nan.

Glossary

Eukaryotic Cell: A biological cell with a membrane-bound nucleus

Endothelial Cell: Cells that line blood vessels inner surfaces

In Vitro: Experimentation outside a living organism (in glass)

Quiescence: A state of cellular inactivity

Senescence: Deterioration of functional cellular characteristics

Telomere: A segment of DNA at the end of chromosomes

Pro-atherosclerotic: Pertaining to atherosclerosis, which is when arteries thicken from fatty deposits.

Abbreviations

ABM: Agent Based Model

CA: Cellular Automata

EBM: Equation Based Model

EC: Endothelial Cell

PC: Proliferating Cell

QC: Quiescent Cell

SC: Senescent Cell

Table of Contents

1 Introduction

1.1 Background Information

The cells which line our blood vessels are called Endothelial cells (EC) which form a layer known as the Endothelium. This layer of cells can repair itself after injury, which is essential to good health, however, the repair process becomes slower with age due to an increased number of larger cells which actively hinder the healing.

These cells are generally in a confluent layer, therefore a larger number of cells are no longer dividing, however, when they're wounded, such as an atheroma, the confluence is broken and the cells leave this phase to continue dividing, repairing the damaged tissue. This process is slower in elderly patients due to the increased number of larger cells, or if the same area is damaged a second time after repair. This is due to scar tissue being less capable of mitosis and repair.

1.2 Aims and Objectives

The main aim of this project is to estimate the affect ageing has on the ability for blood vessels to heal after being scratched. The implications of this project will help professionals further understand the process of wound healing and to provide further insights into the conditions affecting the deadly disease atherosclerosis, which can lead to strokes and heart attacks.
The way the main aim will be implemented requires the development of an agent based model (ABM) to encapsulate the key behaviours associated with ECs, including: cell proliferation, apoptosis, and senescence. This model will record the time taken for the wound to repair itself, and observe any emergent behaviour that takes place through the mitosis and movement of the cells, at varying ages.
For the basis of producing a software solution, I will be looking at the benefits different types of modelling possess, such as Cellular Automata (CA) and Agent Based Modelling (ABM). Then, I'll be building on top of current software frameworks, which already provide basic logic, by giving the agents and environment differing behaviours.
I'll be observing the difference between elderly and younger cells to see how much, if any, age affects repair time.
This project has ample room for expansion; some of these aims include: modelling the problems associated when the endothelium layer doesn't sufficiently repair in time, and the effect on endothelium repair after successive tears (allowing significant scar tissue to build up), showing the differences in speed and process of the repair.

1.3 Summary of Report

This report starts by going through the background information required to understand the differing states and behaviours of the cells that will be modelled and what parameters they should have, it then justifies the use of modelling technique used and looks at current state of the art models to see how they can be adapted to the project. Next, the general flow of the program is defined including the order of functions required to produce accurate results.
The results of the program are laid out in Chapter 6 and are compared to in vitro experiments found from the literature.

2 Literature Review

Our blood vessels inner most wall is called the endothelium and is comprised of endothelial cells (ECs). These cells have certain behaviours which lead them, over time, to decrease their rate of healing. This can cause problems as the damaged artery wall allows for fatty material to build up over time. If this builds up too much or ruptures, a blood clot can form blocking the artery and if this artery supplies blood to the heart it will causes a heart attack. There are several ways software can be used to model this behaviour to better understand and predict undesirable affects, such as atheroma formation. The way this project tackles modelling is an agent based approach, where each EC is simulated and can move around the model independently.

2.1 The Endothelial Cell Cycle

Firstly, it's important to fully understand the mechanisms by which our ECs divide and any biological factors that can change its behaviour. ECs are a specific type of Eukaryotic Cell that line our blood vessels. When these cells are healthy, they secrete molecules, such as hormones, into the blood stream to maintain homeostasis [1]. This is vital as it helps fend off disease progression, keeping the individual healthy.
EC's, like other Eukaryotic Cells undergo several distinct phases during replication as shown in the diagram below, however have another stage they can enter before S Phase.

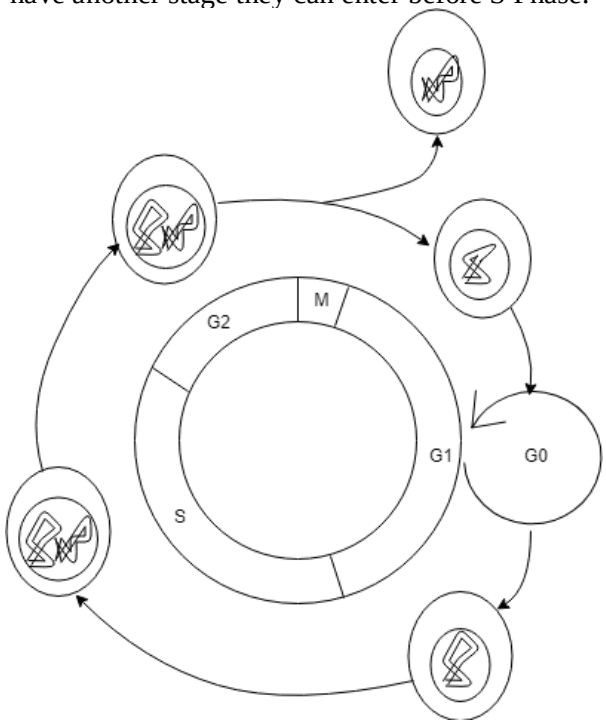


Figure 2.1: Phases of eukaryotic cell cycle adapted from [2]

Stages G_1, S and G_2 are called Interphase; this is the time when the cell is increasing in size, and the lengths of time in each stage are proportional to their relative lengths. As shown in Figure 2.1, during S phase, the DNA is replicated forming a copy of itself which moves onto M phase (mitosis), where the enlarged cell splits into two identical daughter cells [3]. The length of time for a normal Eukaryotic Cell to undergo proliferation is around 24 hours, with 1 hour of that being the M phase, therefore 23 hours (96%) of the time is during cell growth and DNA replication, during which time the cell grows to be about twice its size [3].
However, for eukaryotic cells there is another cycle between the G_1 and S phase. This is called the G_0 phase and generally known as the quiescence state. This is a state of inactivity, usually induced when

EC proliferation is no longer required. If there is a stressor, such as a decrease in external pressure due to the ECs spreading out or moving, the quiescent cell can move out of G_0 back into the normal eukaryotic cell cycle [4]. However, if the EC stays in the quiescent state for too long, it's possible for it to develop into a senescent cell over time where it will never return to the normal cycle [5]. Quiescent cells are stable and can live up to 10 years [6].
In general, ECs are long, flat cells around 5-10μm in radius and 1-2μm wide [7].

2.2 Ageing

An important factor that contributes to pro-atherosclerotic changes to the endothelium is ageing [8]. The number of times an EC can divide is limited, and once reached the cell goes into growth arrest, known as senescence [9]. This is due to the shortening of the ECs telomeres (the end parts of DNA) by 50-200 base pairs each time the cell proliferates. Once these telomeres are shorter than a critical length, the cell becomes senescent. The number of times a cell can proliferate is known as the Hayflick Limit, and for normal ECs is around 50 [10].
Studies have shown that senescent cells accumulate in tissues with age [11, 12], and Cellular Senescence in Aging Primates [13] has shown that the number of senescent cells increases exponentially with age, with total cell count reaching >15% senescent in elderly cases. The limitations of this paper are that the results are from baboons not humans, and so the lifespan is only from 5-30, and the cells were taken from the medial aspect of the arm rather than the endothelium layer. However, this paper is useful in the fact that baboon's telomeres, like humans shorten with proliferation, and the baboon's cells also undergo senescence, and so can be used as the basis for the experiments.

2.3 Senescent Cells

It has been noted the senescent ECs have several characteristics which differ them from normal ECs. Firstly, they are unable to undergo mitosis and have a turnover rate of around 3 years [8], they become enlarged after entering this state [14] and slow down surrounding ECs. Warboys [14] states that senescent ECs could be the main contributor and initiator of atherosclerosis, they go onto suggest that due to the size of the senescent ECs, there is a detrimental effect to the speeds of the neighbouring cells, acting as a barrier and slowing them down. This can hinder wound healing as it will take longer for healthy mitotic ECs to fill the gap and can lead to health problems such as thrombosis. As mentioned above, there's is also an increase in the number of senescent cells over time due to the Hayflick limit, therefore I expect my model to show that as age increases, it takes longer for the wounds to heal.

2.4 Atheroprone Sites

Not all ECs within our blood vessel have the same physiological behaviours; this is due to the differing environmental factors within the vessels, discussed above. This leads to parts of our blood vessels under going higher levels of injury than others. In fact, one of the diseases this project is aimed at further understanding, atherosclerosis, is rather specific, and can be most commonly be found at the bends or branches of arterial trees [15]. These bends and branches are known as atherosusceptible sites, which have enhanced proinflamitory activation, increasing rate of proliferation [15]. These atherosusceptible sites therefore have a higher rate of injury and cell turnover compared to EC at atherprotected sites [16, 17, 18]. Analysis by Chaudhury et al showed that the ECs at Atheroprone sites express proteins that respond to lipopolysaccharides by priming for apoptosis and proliferation [15]. They also state that wherever JNK1 is active is where apoptosis and EC turnover occur in arteries.
I will therefore be looking at branches and bends within my model as they are the areas where there is the highest level of turbulence and concentration of JNK; leading to the greatest injury of the endothelium wall. Which, in turn has the greatest concentration of EC apoptosis and proliferation.

2.5 Methods of Modelling

There three options for modelling the interactions between endothelial cells. Cellular Automata (CA) uses an orthogonal grid of homogeneous cells that interact with their neighbouring cells. Its advantages are that runtime is extremely quick and it can produce complex macro-scale emergent behaviour of the interacting cells [19]. However, the disadvantages are that due to the orthogonal grid, cells are fixed in place, unable to move; this is very much a simplification of the project as ECs move around on the endothelium to fill gaps and is an important factor for wound healing. Another disadvantage of CA is that it can only model local interaction between neighbouring cells, therefore any change further away from the cell won't be noticed until it cascades down the subsequent neighbouring cells over several iterations

Another modelling method would be to use Equation Based Modelling (EBM), otherwise known as continuum modelling. Here, differential equations are used to model population densities. These differential equations could be used to show the rates of healing when a wound has occurred and can provide steady states when confluences have formed. Being equation based, the program could also be written in any language and many libraries already exist for their implementation. However, this approach is limited as the equations do not model each cell individually and so individual interactions between cells is lost. EBMs are also deterministic and so cannot model the stochastic behaviours exhibited by cells.

Finally, an Agent Based Model (ABM) is a dynamic system of interacting agents that builds upon cellular automata. This dynamic property is crucial in producing realistic emergent behaviours as it more closely resembles what occurs in nature. The downside is, that due to the free movement of the cells, expensive calculations must be implemented to resolve overlapping and collisions in more accurate systems, introducing scalability issues. However, there are several methods out there for reducing the time taken; Epitheliome, an ABM created by Dr. Dawn Walker [20], embedded their overlap logic as C within their MATLAB code. This is also possible within python [21]. ABMs also produce graphical outputs of each iteration and can be used to further understand the behaviour of the cells. For these reasons, I believe it's best to complete this project using an Agent Based Model.

2.6 Review of Agent Based Software

There are existing ABMs that have been developed to monitor cellular interactions. The first, Epitheliome, by Dr. Dawn Walker [20] is the most applicable to this project. It uses an agent based approach to visualise the time taken and movement of endothelial cells into a wound with different levels of calcium ions in the environment. The underlying logic of Epitheliome is laid out more in [22] It accurately models the contact inhibition of cells and differentiation of endothelial cells to quiescent cells in the G0 phase.

The implementation of the cell cycle is similar to what was discussed in 2.1 with each cell progressing one tick through the cell cycle each iteration. The duration of S-G2-M phase and G1 phase being slightly different for each cell, imitating the stochastic nature of cells.

The limitations of this approach to my project is the lack of senescent cells being modelled in the simulation which are thought to act as barriers to the endothelial and quiescent cells during migration [14]. Therefore, Epitheliome is unable to monitor the rate of wound healing with age.

Two programs that use agent based modelling to allow for the type of emergent biological behaviours I'm looking for have been investigated. The first program is SPARK which is a lightweight and efficient tool for CA. Being so lightweight, Spark is very capable of modelling the number of cells I would require for this project; in fact, it can simulate a grid of 101x101 with 10201 cells in real time. Its programs are written in SPARK-PL which is translated into Java source code, meaning a significant amount of time will be required to learn the new language. Another downside is that being a CA, the ECs are embedded into the endothelial matrix (the layer the cells sit on top) and therefore are unable to move around the system, and as explained above, this is a simplification of reality as ECs are constantly moving or shifting on top of the endothelium layer.

The other program is a python based ABM by Marziha Tehrani, a PhD student, called CellABM. It uses two agents to model interactions between cancer cells and stem cells and has several classes which allows the user to easily change the rules of each phase of the cell cycle along with the initial cell parameters, such as size, direction and speed. However at large cell numbers, it is rather slow and there are no capabilities of interacting with the agents during the simulation.

There are three other software frameworks I've looked at, but not as in-depth as the two described above; they are: Net Logo, Mason, and Repast.

Below, I have quantitatively summarised the strengths of each software in relation to each other. A scoring system between 1 (low) and 5 (high) is multiplied by the weight of each category. This gives a total showing the overall usefulness of the software.

		Comparison of Software				
		Spark	**CellABM**	**Net Logo**	**Mason**	**Repast**
Method (CA or ABM)	0.1	1	5	5	5	5
Contact Resolution	0.2	1	4	2	2	3
Language	0.2	2	5	3	3	3
Interaction during simulation (GUI)	0.1	3	1	5	4	5
Speed	0.3	4	2	4	4	4
Familiarisation	0.1	3	5	1	1	1
Total	1	2.8	3.5	3.3	3.2	3.5

Table 2.1: Quantifying the differences between possible software

From Table 2.1 CellABM and Repast both score the highest at 3.5 meaning they're equally suited to this project. However, the defining factors between the two are the graphical user interface (GUI) where Repast scored 5 and CellABM 1, and familiarisation where Repast scored 1 and CellABM 5. As this project doesn't require a GUI as there is no interaction with the simulation whilst running, familiarisation is the more important metric and so CellABM is the software of choice.

2.7 Cell Migration

A key element of ECs is their ability to migrate. Endothelial cell migration is a fundamental process to our life, allowing the formation of embryos, organs and tissues. For developed humans, migration allows for immunosuppression and more importantly to the project, the migration of ECs into the wound of a damaged blood vessel to restore the vessels integrity [23].
ECs will migrate in a random manner if there are no external stimuli and will diffuse into the available space [24] until a confluence is formed. Once the cells have formed a monolayer, they bond to each other and the endothelial surface preventing further migration.

2.8 Contact Inhibition and Confluence Detection

Over time, ECs will migrate into any open space and if possible proliferate to form new ECs. This will continue to occur until the area is filled with ECs and there is no more space for proliferation. When cells come into contact with each other, cell growth is arrested by a process known as contact inhibition [25], meaning that when a monolayer is formed ECs are no longer able to proliferate. If the ECs are unable to proliferate they eventually differentiate into quiescent cells where they no longer undergo mitosis.

Confluence Detection occurs when migration and proliferation is no longer possible due to the contact inhibition on the monolayer. At this point, several of the ECs will have differentiated into Quiescent Cells.

3 Requirements and Analysis

3.1 Methodology

For the development of the program to discover the effect age has on heart attacks, an Agent Based Model will provide the best results for the user. As discussed in Chapter 2.5 ABMs model each cell individually with their own parameters, allowing for a more distributed representation of the cells, such as each cell can vary in radius slightly from each other. An ABM also provides a graphical output of how the cells move, allowing us to better understand what's happening with the emergent behaviour in a visual way. The ABM approach is better than an equation based approach as there is no individual agent representation in EBMs and so approximations may be too significant to produce reliable results. Cellular automata was not chosen as it would incorrectly model the endothelial cells on the environment, not allowing them to migrate into the wound and therefore not answering the research question.

3.2 Aims and Requirements

The main aim of this project is to demonstrate and help professional further understand the affect an increase in senescence cells from ageing has on the ability for a wounded area of ECs to repair itself. The main observation will be time taken for the ECs to divide and move into the gap of the wound, once more forming a confluent layer.
To observer the migration of cells moving into a wound with time an agent based model will be produced as described above in Chapter 2.5 however, the current models shown in Chapter 2.6 lack the correct logic or behaviours that occurs within blood vessels. Below, I outline the functional and non-functional requirements, parameters, and rules that need to be met to produce an accurate and correct model.

3.2.1 Functional Requirements

It is critical that the system:
1) Uses an appropriate time scale for each iteration
2) Creates a wound when a confluence is made
3) Includes senescent cells as entities
4) Can vary the level of senescent cells with age
5) Forms a confluence before being wounded

Table 3.1: Critical functional requirements

It is important that the system:
6) Tells the user how long it took for wound healing to occur
7) Produces graphs of cell locations each iteration

Table 3.2: Important functional requirements

It is desirable that the system:
8) Stops the simulation when second confluence is formed

Table 3.3: Desirable functional requirements

It is optional that the system:
9) Models senescent cell death

Table 3.4: Optional functional requirements

3.2.2 Non-functional Requirements

It is desirable that the system:
10) Is simple to run from the command line
11) Is commented well for future development

3.5: Non-functional requirements

3.2.3 Parameters and Rules

The desired emergent behaviour will be produced through the interaction of several agents over several iterations. The way these agents move and interact will be dictated by the implementation of several rules with associated parameters. The values for parameters will be based on the literature found in Chapter 2, however in some cases assumptions must be made due to lacking experimental data.
These rules will be actioned each iteration, and, over time will produce novel behaviours that can be visualised on the output graph.

3.2.4 Emergent Behaviours

Emergent behaviours arise through the interaction of the above rules and are not hard-coded, but observed. Some of these behaviours in action include the formation of tissues and organs and the expansion of tumours. For this project, I expect to see an emergent behaviour of wound healing when the blood vessel is damaged, by having the Quiescent cells differentiate back to Proliferating cells (PCs) due to the increased space, and these PCs migrating and proliferating to fill the space; once more forming a monolayer of cells which will differentiate back to Quiescent Cells. Another expected emergent behaviour is the obstruction of migration of PCs from the Senescent cells leading to delayed healing, increasing the chances of forming an atheroma and blood clot, leading to a heart attack.

3.3 An overview of Python and its Class System

Since the implementation will be driven using CellABM, Python is the language of choice for this project. Python is similar to other widely used languages such as Java and JavaScript [29] in that it is an interpreted and an Object Orientated Programming (OOP) language. However, Python has some significant differences that lead it to be syntactically easier to read than Java and it has better code reuse than JavaScript. A Python program is generally 3-5 times smaller than the same program written in Java, thus decreasing development time and reducing the chance of bugs.
In Python, data is encapsulated inside objects. These objects can change their own data or interact with other objects. This method of object orientation can be used to represent the different types of agents required in the program.
Python also uses inheritance. This means that instead of writing the same function for several classes, there can be one parent class with the function and other classes can inherited that function from them, reducing repeated code. In the case of CellABM, this means each cell type: Proliferative, Senescent, and Quiescent can all inherit the same apoptosis (cell death) function from an overall general_cell class.

3.4 Limitations of Model

Either due to time or computational constraints there are a few areas that this project will not be covering. Firstly, due to the lack of understanding the advanced Biology of the inner workings of ECs, I will be unable to implement all the of rules biologists have found that cause cellular senescence.

Another area I will not be covering are the multiple ways the endothelial monolayer gap can be filled during healing. I am only modelling the spreading of adjacent ECs into the gap due to the decrease in pressure caused by the lack of cells pushing back. The other ways the gap can be filled include: hyperplasia of existing endothelial cells and engraftment of circulating endothelial progenitor cells [8].
I am also assuming, that I am modelling ECs from a healthy person with a Hayflick limit (maximum proliferation) of 50, ignoring deficiencies such as Werner syndrome which causes individuals to have a population growth of 53% and total replicative life span of 27% compared to normal cells [26].
I will not be creating a graphical user interface (GUI) for the user to change parameters on the fly in the simulation. All parameters will be set at the beginning of the simulation and shall remain unchanged. To observe the effect of the changing parameters, several simulations must be run with varying initial conditions.

3.5 Risk Analysis

I've included all the risks I believe are associated with my project below. I outline the nature of the risk, then give it a likelihood and impact score from 1 – 4, 1 being unlikely / negligible and 4 being very likely / project threatening then provide a mitigation plan to decrease severity.

		Likelihood			
		Very unlikely 1	Unlikely 2	Likely 3	Very Likely 4
Impact	Negligible 1	1	2	3	4
	Low 2	2	4	6	8
	Significant 3	3	6	9	12
	Catastrophic 4	4	8	12	16

Table 3.6: Risk Rating Matrix where Risk Rating = Likelihood x Impact

Risk Event	Likelihood	Impact	Risk Rating	Mitigation
Loss of developers' code	1	4	4	Backups of the developers' machine are take daily to an external hard-drive. The code will also be tracked on GitHub.
External event prevents progression	2	3	6	Careful project planning implementation of contingency plans if developer starts to fall behind. Some weeks are designed to have less work in case developer needs to catch up.
Optimistic project plan	3	3	9	Enough time must be given to the development of the software and is something that shouldn't be rushed. Adjustment to project plan may be required if developer start to lag.
Completion of code hinders completion of dissertation	2	4	8	Enough time will be given to produce several drafts of the final dissertation in the project plan.
New functions not working with current software	2	3	6	Ensuring there are no compatibility issues and correct design practices are followed,

				such as the creation of UML diagrams showing function interaction.
Contact resolution scalability not fixed	3	4	12	Review of different software for contact resolution. Decreasing experiment area is a last resort to ensuring a confluence can be modelled.
Lack of accurate data	4	3	12	Continual reviewing of papers surrounding the topic for any extra hints. Otherwise a heuristic approach and sensitivity analysis with several simulations should provide accurate results.
System too slow for use under standard conditions	3	4	12	Avoid implementation of nested loops, and constantly assess performance. Possibility of running simulation on Iceberg.
Requirements change during development	1	3	3	The code will be implemented in an Object Orientated manner, providing modularity of functions with little refactoring.

Table 3.7: Risk identification, analysis and planned mitigations.

3.6 Evaluation and Testing

As it is not possible to prove an ABM is correct, it is important to test the program. Several tests of varying nature will be run to check that the implemented software is working as expected. Unit tests will be produced for each rule associated with the agents, and ensures that functions return the correct results for different inputs. Next, face validation, also known as plausibility checking [A Validation Methodology for Agent-Based Simulations], can be used to check expected behaviour in simple scenarios and full simulations. Quantitate validation will then be used to see if the predicted wound closure rate matches that found in [27] at 8.35μm per hour, and results will also be contrasted with [28] which shows cells migrate fastest in the first 0-24 hours at 84μm per day, rapidly slowing down to 20μm per day after 40 hours.

Statistical validation will be done on the number of cells in the wound and time taken for the wound to heal, due to their inherently stochastic nature. This is achieved by running multiple simulations to generate a distribution of predicted values. However, due to the lack of current experimental data on time taken for endothelial wounds to heal with varying levels of senescence, more rigorous validation such as Students T test will not be able to be carried out.

Finally, locally sensitivity analysis will be carried out in addition to the above validation to determine the usefulness of the program. Here the parameters surrounding cell migration speed and proliferation rate will be independently varied and simulations run to visualise the change on the output of the system.

4 Design

As seen above, there are several ways of developing an ABM to implement the requirements and it has been decided to continue work on CellABM, a PhD project by Marzieh Tehrani. In this chapter, we will explore the underling language of the program and how it can be used to model an ABM, then discuss the class diagram and flow charts of how information will flow through the system, finally discussing what simulations will be run to answer the research question.

4.1 Theorised Program Flow

Below are the guides that will be followed during the development of the program. They provide the road map of how each class and function interacts with each other, leading to emergent behaviour of the cells. A quick overview of the cellular state changes is given in figure 4.1, showing how, generally, endothelial cells start out being normal Proliferating cells, then they can either move onto being Quiescent or Senescent. Quiescent cells can revert to Proliferating cells or turn Senescent if they persist long enough. As shown, Senescent cells act as a sink, trapping the cell in that state until the end of the simulation.

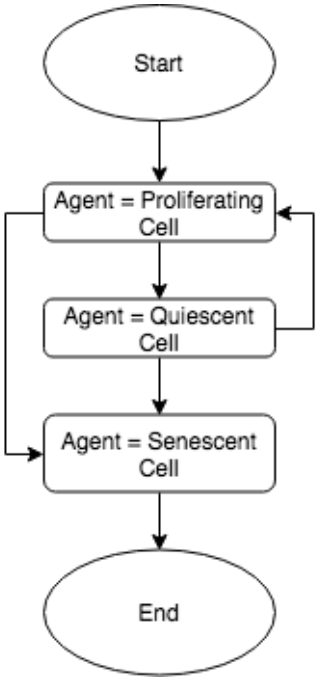


Figure 4.1: Cellular differentiation

4.1.1 CellABM

This flow chart shows how the overall main class will run. It will start by taking the parameters from the user, initialising the environment with these parameters and ensuring the initial agents aren't overlapping. When this is set up, the program will move into an iterative process of solving the agents (allowing to perform their programmed rules), ensuring they aren't overlapping, then checking the number of quiescent cells in the environment. If the number of quiescent cells is larger than a heuristically found threshold a confluence has been formed and the environment simulates the wound then continues the loop. At the end of each iteration, a graph will be plotted showing the location of each agent on the environment.

When the number of quiescent cells passes the user set threshold for a second time, the simulation is stopped as a confluence will have re-formed, this will also produce a growth curve of the agents over the iterations.

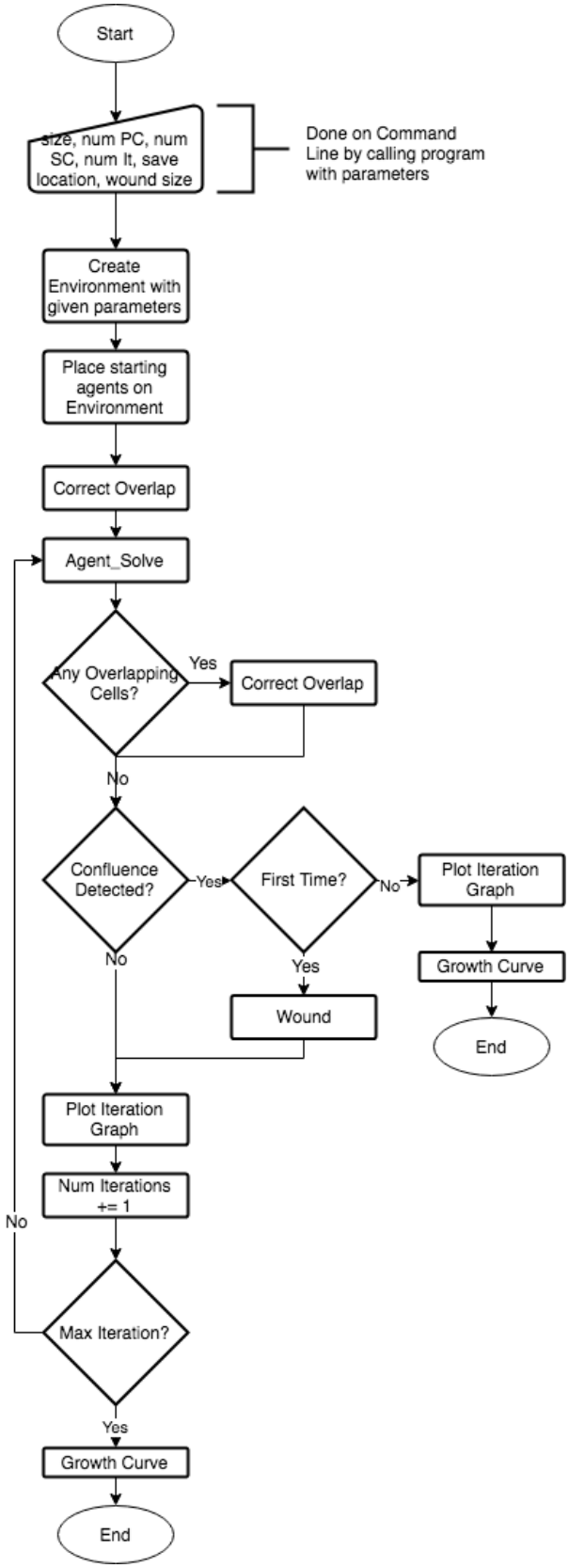


Figure 4.2: CellABM class overview

4.1.2 Cell Transitions

A more thorough plan of cell evolution is given below in figure 4.3. This shows the intended logic behind each of the cell stages, and how the cells will differentiate with the simulation.
The Proliferative cells have both a turnover value and stage value (not shown here). The turnover is the Hayflick Limit [10] mentioned in the Chapter 2.2, and once reached, the proliferative cell will differentiate into a senescent cell. Cell stage however, will be used to track what stage in the cell cycle the cell is at and to decide whether the proliferative cell should undergo mitosis that iteration.
The quiescent and senescent cells only have an age value associated with them. As these cells do not undergo mitosis, there is no need to track what stage of the cell cycle these cells are in and is therefore used as the Hayflick representation.

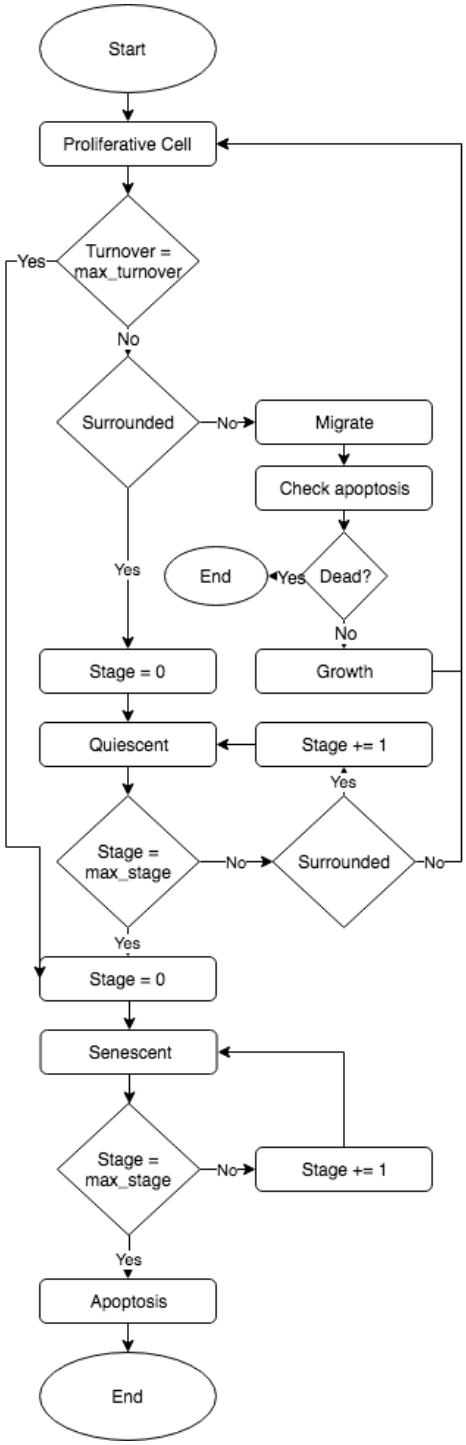


Figure 4.3: Cell Transition Steps

4.1.3 Agent Solve

This flow chart has been created by looking at the current underlying logic for the agent_solve class in CellABM and including the extra steps required to allow for the new rules and cells the project requires. Each iteration, these steps will be run on every cell in the model.
For Proliferative and Quiescent Cells, it is important to test whether they will become Senescent first as if this is true it shows the cells have passed the Hayflick limit [10] as seen in chapter 2.2, and in reality, their telomere ends would have passed their critical length turning the cell senescent. Senescent cells are unable to differentiate back to a PC or QC, thus ever iteration they only test to see whether they will under apoptosis.

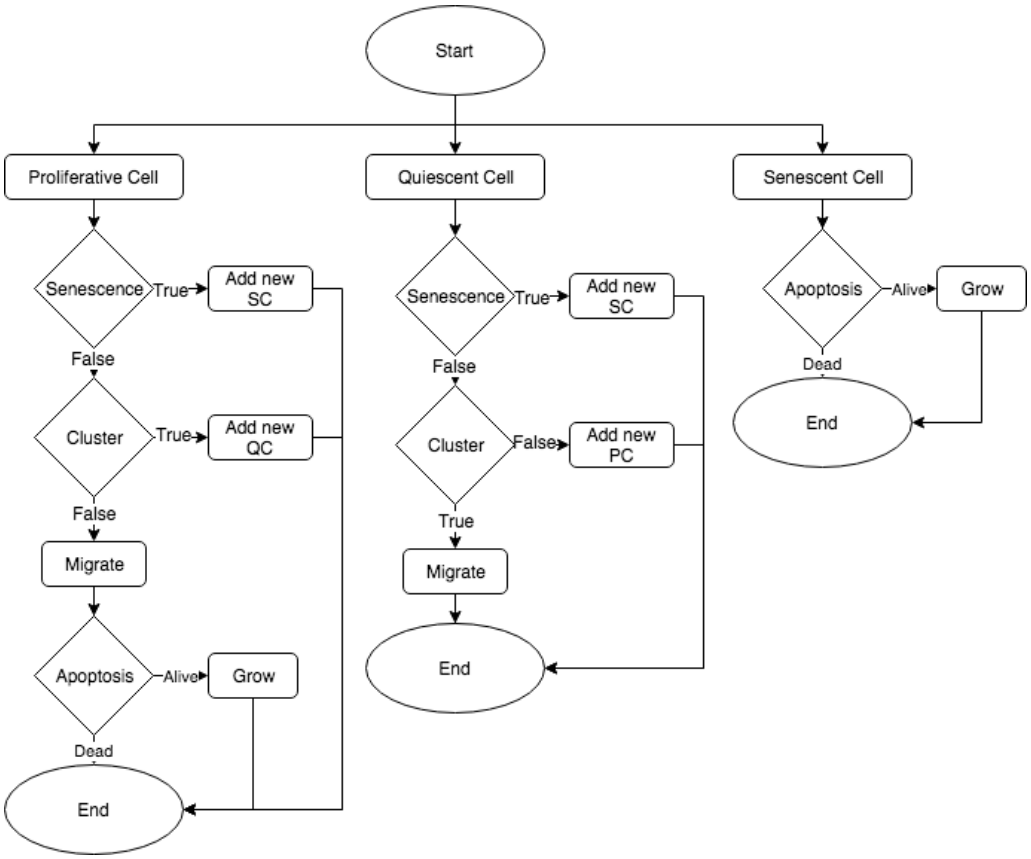


Figure 4.4: Overview of agent_solve class flow

4.1.4 Proliferative Growth

Every iteration each proliferative cell increments 1 stage through the cell cycle. As there are 4 stages in the cell cycle, and the cell needs to double in size by stage 4 to undergo mitosis [3] the following algorithm shown in Figure 4.5 was devised. Here no matter what stage each cell is at it will be double its original size before undergoing mitosis.

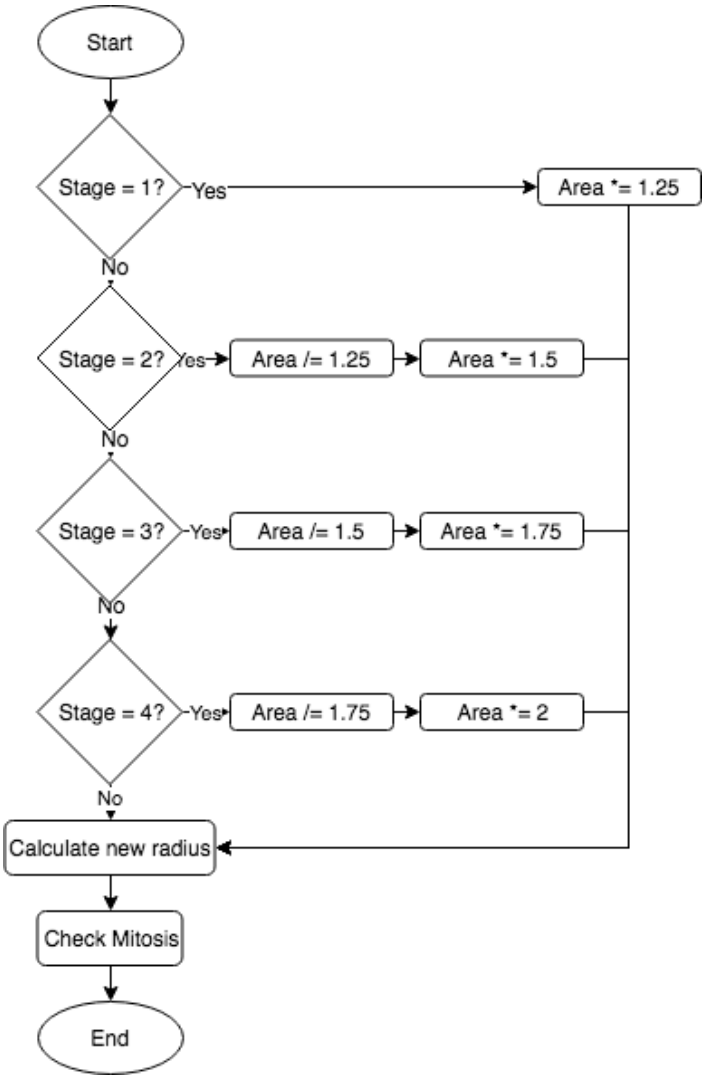


Figure 4.5: How growth is calculated each iteration

4.1.5 Mitosis

After the proliferative cell has undergone growth, the program checks to see whether it can perform mitosis. To qualify, the cell must be in M phase (stage 4) and will be double its starting size. Here the parent cell halves its area, turning into one of the two daughter cells and a new cell is created from the proliferating cell class with the same area as the first daughter cell. If the cell is not in M phase, the program increments the stage of the cell cycle by one and returns it.

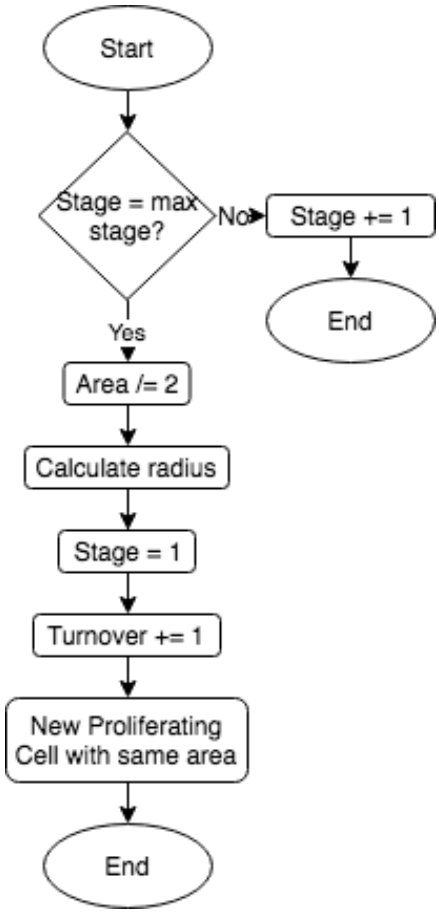


Figure 4.6: Mitosis algorithm

4.2 Class Diagrams

This class diagram is intended to show the information flow throughout the program and how the classes communicate with each other. An important feature to note is the general_cell class acting as a parent class for the three cell types.

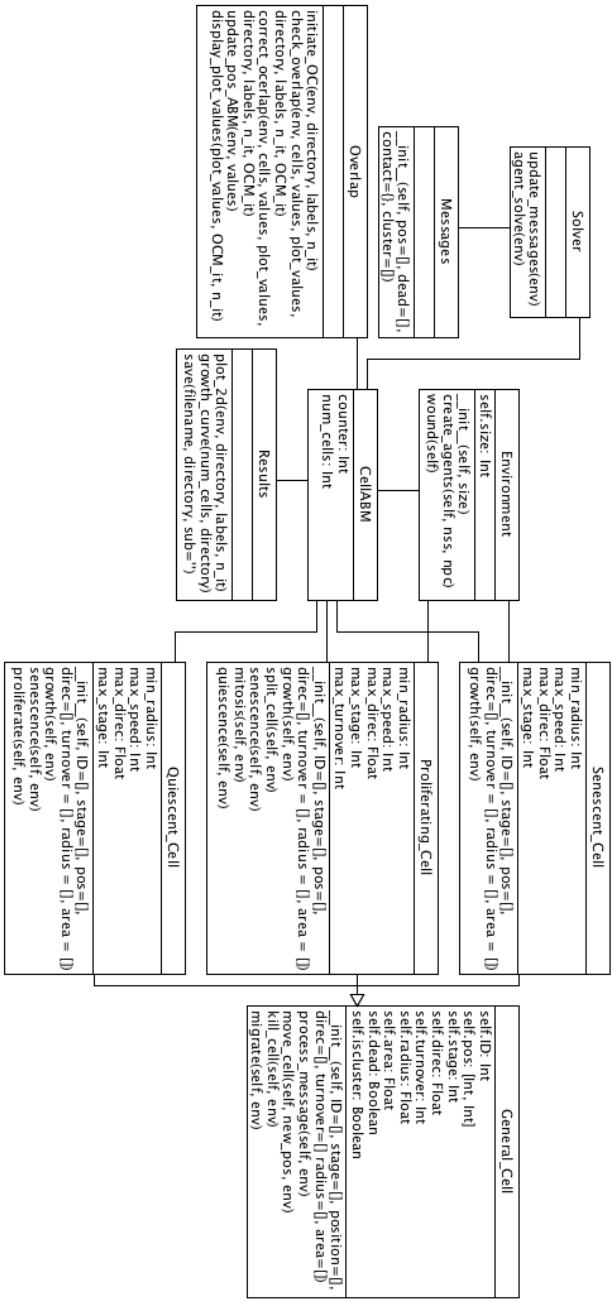


Figure 4.7: Class diagram of CellABM

4.3 Environment

At the beginning of the program, the user will define several key parameters used to initialise the environment. Notably, the size (in micrometres), the number of starting Proliferating Cells and the number of starting Senescent Cells. This allows the user to define cell ratios for differing patient ages in accordance with the research question.

The Environment class creates the starting agents with a random set of parameters taken from a distributed range given, and appends them to a list of starting agents.

The environment will be modelled as a discrete space where agents cannot leave, to preserve computational runtime, and will provide the space for the agents to interact with each other. Cell positions can be mapped into this 2D space using a 2D array of equal size to the user's definition and giving each cell an [x, y] coordinate.

Although in biology endothelial cells live in a 3D space, they tend not to over-lap one another, thus creating a 2D plane. For this reason, it is believed that little information is lost by modelling in 2D.

4.4 Simulations to Run

As the main objective of this project is to determine the different times taken for a wound to heal whilst varying the person's age, several simulations will be run with varying percentages of senescent cells in accordance with the primate paper in chapter 2.2 [13], with the time taken being plotted. ABMs are generally stochastic, and CellABM is no different. The initial placement of cells onto the environment is random, so too is their starting size and stage in the cell cycle. Due to these random variables, several simulations with the same starting parameters must be run to achieve adequate analysis of the model.
Results of the simulations will be compared to an in vitro study of human umbilical vein endothelial cells which have been wounded with p20 pipette (around 400μm) on an area of 1mm by 1mm [27].

5 Implementation and Testing

This chapter is concerned with the final process involved with implementing the background logic to produce the desired emergent behaviours. It will go through the rules outlined in 3.2.3 in detail, then move onto unit and face testing of these rules.

5.1 Implementation

CellABM already had several sections of the program and logic developed, including overlap correction, basic cell agents, environment initialisation and basic cell interactions; therefore, this chapter will focus on the areas of the program that have been changed or developed to produce the required emergent behaviour and observations.
CellABM was originally written in Python 2.7 which was released in 2010 but is seen as the legacy version of the language, with Python 3.6 being the supported language of choice for present and future programs. Thankfully many of the modules from Python 2.7 have been ported over to Python 3.6, such as NumPy which CellABM uses for matrix creation and mathematical functions. This leaves only basic refactoring of the code and changing print statements to functions to make CellABM Python 3.6 compatible. The changes brought in by Python 3.6 are to adjust certain aspects of the old Python program language to be simpler for new programmers to develop, and make it easier to read. These rules have been created using the logic shown in the design flow charts.

5.1.1 Changes to CellABM

A significant amount of refactoring has taken place to convert the original code into PEP8 [30] and a number of unused parameters have been removed. In addition to these adaptions, a new agent has been introduced to increase the total number to 3. Docstrings have been created for each class and method, allowing future development of the program to be achieved easily. The time step has been set to every 6 hours, this is a balance between the simulations taking too long to run to achieve adequate

results to compensate for the stochastic nature, and not being too long to prevent behaviours being expressed.

5.1.2 Senescent Agent

5.1.2.1 Class overview

The senescent agent is a subclass of the general cell class allowing for varying parameters to be specific to the senescent cell. As proliferating and quiescent cells can differentiate into senescent cells and they are capable of being 5μm radius this is the minimum radius the senescent cells can be. It has been programmed as 4.9 so cells at 5μm aren't removed from the simulation.
These cells are intended to act as barriers to the surrounding cell, slowing down the wound healing, therefore, a speed of 0 has been assigned to them ensuring they don't migrate around the simulation.
As seen in chapter 2.3 senescent cells can live upwards of three years [8], therefore as each iteration is six hours, the cells can be in the simulation for a maximum of 4380 iterations. However, it is extremely unlikely for a simulation to run for this long and is intended to be used alongside the initial creation of senescent cells where they are given a random stage between 1 and 4380.

```
class sc(general_cell):
    """
    This is a subclass of general cell for the senescent agent.

    Public methods:
    :growth: Increases area of cell
    Instance variables:
    :min_radius: The smallest the cell can be before dying
    :max_speed: How fast the cell moves per iteration
    :max_direc:
    :max_stage: How many iterations the cell can last for
    :num_sc: The total number of senescent cells
    """
    min_radius = 4.9
    max_speed = 0  # Senescent cells don't move
    max_direc = 0
    max_stage = 4380  # represents 3 years
    num_sc = 0
```

Figure 5.1: Parameters for Senescent Cells

5.1.2.2 Growth

From meeting with my domain expert, Prof. Paul Evans, it was found that senescent cells can, in some cases, grow up to 10 times their original size in the first two weeks, then staying relatively the same size for the rest of their life. This means they can potentially grow up to 100μm in diameter. As the senescent cells grow within their first two weeks and each iteration equates to six hours of simulated time, they should reach 100μm diameter within 56 iterations. To achieve this, the growth function increases the cells diameter by 1.6μm each iteration. However, this on its own has no prevention for the cell to increase over 100μm. To control this a condition is used to ensure only cells that are smaller than 100μm diameter have their radius increased.
This function also increases the age of the cell by 1 each iteration to account for older cells dying out.

```
def growth(self):
    """
    Increases area of cell.

    As long as cell is smaller than threshold, it will increase in size
    each iteration. Also each iteration its stage (age) is increased by 1
    :return: The cell with incremented stage and either the same size or larger
    """
    if self.radius < 50:
        self.radius += 0.8
        self.area = math.pi*(self.radius*self.radius)
    self.stage += 1
    return self
```

Figure 5.2: Senescent Cell Growth

5.1.2.3 Apoptosis

When senescent cells have lived for three years, stage = 4380, they are removed from the simulation. As simulations will generally only run for days to weeks this is rarely called and generally the only cells that will undergo this apoptosis will be the ones created at the start of the simulation as they will have a random stage between 1 and 4380.

```
for agent in env.senescent_cells:
    if agent.stage == agent.max_stage:
        agent.kill_cell()
```

Figure 5.3: Senescent removal

5.1.3 Quiescent Agent

5.1.3.1 Class Overview

The quiescent agent is a subclass of the general cell allowing different parameters to the senescent and proliferating agents. As proliferating cells change state to quiescent cells and the smallest a proliferating cell can be is 4.9μm in radius, the same is true for the QC.
Quiescent cells occur when proliferation is no longer required, generally when a monolayer has been formed, for this reason the agents have been assumed to have a speed of 0 and so they don't actively migrate in the simulation.
It has been theorised here that QCs live for around two months before turning senescent. However, the simulation usually isn't run for this long and new QCs are created with a stage of 1, therefore quiescent cells turning into senescent cells will rarely be seen.

```
class qc(general_cell):
    """
    This is a subclass of general cell for the quiescent agent.

    Public methods:
    :senescence: When cell is old enough differentiates to senescent
    :proliferating: When cell can proliferate differentiates to proliferating
    Instance variables:
    :min_radius: The smallest the cell can be before dying
    :max_speed: How fast the cell moves per iteration
    :max_direc:
    :max_stage: How many iterations until the cell differentiates to senescent
    :num_qc: The total number of quiescent cells
    """
    min_radius = 4.9
    max_speed = 0  # Quiescent cells cannot move
    max_direc = round((2.0/3)*math.pi, 3)
    max_stage = 240  # each level = 6hrs of real time
    num_qc = 0
```

Figure 5.4: Parameters for Quiescent Cells

5.1.3.2 Senescence

Quiescent cells (QCs) can differentiate into Senescent Cells (SCs) when they have been in the simulation for long enough. Each iteration the QC is tested to see whether it can change state, if so the current QC is removed from the simulation by killing it and a new SC is created with the original QCs position, radius, and area. If stage change is not possible, the age (stage) of the cell is increased by one.

```
def senescence(self):
    """
    Differentiate current (quiescent) cell into senescent cell.

    If the cell has passed its Hayflick limit, it will differentiate,
    else the cell remains unchanged and continues
    :return: Either aged cell or a new senescent cell in the same position
    """
    if self.stage == self.max_stage:  # Required minimum of 239 iterations
        self.kill_cell()
        senescent_pos = [self.pos[0], self.pos[1]]
        senescent_cell = sc(ID=sc.num_sc, stage=1, pos=senescent_pos, direc=random.random() * 2 * math.pi,
                            turnover=1, radius=self.radius, area=self.area)
        senescence = senescent_cell
    else:
        senescence = None
        self.stage += 1
    return senescence
```

Figure 5.5: Quiescence state change to Senescent

5.1.3.3 Quiescent Cell Cycle Re-entry

When there is adequate space around the Quiescent Cell (QC) it can change back to a Proliferating Cell (PC) as seen in Figure 4.1. Each iteration the number of cells surrounding the QC is added up and if it is fewer than the assumed value of 4, it is believed that space has freed up around the QC, allowing it to proliferate. The state change is made by killing the QC and creating a new PC with the same: position, turnover, radius, and area of the QC.

```
def proliferating(self):
    """
    Differentiate current (quiescent) cell back to proliferating cell.

    :return: A new proliferating cell in the same position
    """
    from proliferating_cells import pc
    self.kill_cell()
    proliferating_pos = [self.pos[0], self.pos[1]]
    proliferating_cell = pc(ID=pc.num_pc, stage=1, pos=proliferating_pos, direc=random.random() * 2 * math.pi,
                            turnover=self.turnover, radius=self.radius, area=self.area)
    proliferating = proliferating_cell
    return proliferating
```

Figure 5.6: Quiescent state change to Proliferating

5.1.4 Proliferating Agent

5.1.4.1 Class Overview

The Proliferating Cell (PC) will be the most prevalent agent as it the source agent as seen in Figure 4.1. The PC class is a subclass of the general cell class and extends it by giving the PC specific behaviours. As seen in chapter 2.1 endothelial cells have a radius between 5 and 10μm [7] and so the minimum radius for PCs is set to 4.9. If it was set to 5, there would be a case where newly formed PCs that start out with a radius of 5 will be removed during the apoptosis function.
I have assumed that PCs move at 1 μm per minute, giving them a speed of 360μm for the iteration. As seen in chapter 2.1, endothelial cells have distinct stages in the cell cycle [2]. This is tracked by assigning a stage to each PC as shown in Table 5.1.

State	Cell Cycle Stage
1	G1
2	S
3	G2
4	M

Table 5.1: Cell cycle parameters.

From chapter 2.2 is it seen that each time a cell undergoes mitosis and divided its telomeres shorten, thus after several divisions the telomeres are too short to continue the dividing and the cell turns Senescent, this limit is known as the Hayflick limit and has been shown to be around 50 divisions [10]. Thus, the maximum turnover for each PC is set to 50.

```
class pc(general_cell):
    """
    This is a subclass of general cell for the proliferating agent.

    Public methods:
    :split_cell: Creation of daughter cell
    :senescence: When max_turnover is reached cell differentiates to senescent
    :mitosis: M phase of the cell cycle where the cell splits
    :growth: The cell doubles in size during one cycle
    :quiescence: When the cell can no longer proliferate it differentiates to quiescent
    Instance variables:
    :min_radius: The smallest the cell can be before dying
    :max_speed: How fast the cell moves per iteration
    :max_direc:
    :max_stage: How many iterations are in one cell cycle
    :max_turnover: How many times the cell can proliferate before becoming senescent
    :num_pc: The total number of proliferative cells
    """
    min_radius = 4.9
    max_speed = 360  # move at 1micrometer a min
    max_direc = round((2.0/3)*math.pi, 3)
    max_stage = 4
    max_turnover = 50  # Hayflick limit of 50
    num_pc = 0
```

Figure 5.6: Parameters for Proliferating Cells

5.1.4.2 Senescence

As mentioned in and 5.1.4.1 Proliferating Cells (PC) will turn Senescent (SC) when they have hit the maximum turnover of 50. This state change is executed by removing the current PC from the simulation and creating a new SC at the same position with same radius and area. The SC agent uses the turnover parameter to track the age of the cell, and is therefore set to 1.

```
def senescence(self):
    """
    Differentiate current (proliferating) cell into senescent cell.

    If the cell has split enough times and passed its Hayflick limit,
    it will differentiate, else the cell remains unchanged and continues
    :return: Either unchanged cell or new senescent cell
    """
    if self.turnover == self.max_turnover:
        self.kill_cell()
        senescent_pos = [self.pos[0], self.pos[1]]
        senescent_cell = sc(ID=sc.num_sc, stage=1, pos=senescent_pos, direc=random.random() * 2 * math.pi,
                            turnover=1, radius=self.radius, area=self.area)
        senescence = senescent_cell
    else:
        senescence = None
    return senescence
```

Figure 5.7: Proliferative state change to Senescent

5.1.4.3 Quiescence

As seen in chapter 2.1 Proliferating Cells can enter a special state within the cell cycle known as G0 or the quiescent state [4]. This occurs when the cell no longer needs to proliferate due to being surrounded by other cells. The detection of number of neighbours is programmed in the correct overlap function within the overlap class. This is because the correct overlap function was already calculating the number of neighbours each cell had and would therefore be computationally wasteful to recalculate this. As shown in figure 5.8 the number of neighbours required for a proliferating cell to turn quiescent is 4. This was determined by running several simulations at varying values to visually

see how well a confluence formed. Too low a threshold and cells would turn quiescent even with space to proliferate and a higher value caused certain cells to be surrounded but not turn quiescent. The proliferative agent turns quiescent by removing the current PC from the simulation and creating a new QC agent in its place with the same: turnover, radius, and area as seen in figure 5.9.

```
if len(neighbour) > 3:  # parameter: changeable for confluence
    if not cells[i].iscluster:
        cells[i].iscluster = True
else:
    cells[i].iscluster = False
```

Figure 5.8: correct overlap function detecting if cell is surrounded.

```
def quiescence(self):
    """
    Differentiate current (proliferating) cell into quiescent cell.

    When the cell is surrounded and is unable to proliferate anymore,
    it will turn quiescent. Here the original cell is removed from the
    simulation and a new cell of type quiescent it created.
    :return: A new quiescent cell
    """
    self.kill_cell()
    quiescent_pos = [self.pos[0], self.pos[1]]
    quiescent_cell = qc(ID=qc.num_qc, stage=1, pos=quiescent_pos, direc=random.random()*2*math.pi,
                        turnover=self.turnover, radius=self.radius, area=self.area)
    quiescent_cell.iscluster = True
    quiescence = quiescent_cell
    return quiescence
```

Figure 5.9: Proliferating Cell state change to Quiescent.

5.1.4.4 Growth

Through one cycle of the cell cycle a proliferative cell doubles in area so it can divide into two equally sized daughter cells during mitosis. Therefore, what could be done is to increase the size of the cell by two times only when it is in stage 4, however this will assume that all growth occurs just before cell division, will make the growth look sporadic, and is an incorrect model of the biology [3]. Therefore, this function has been created to increase the size of the cell by ¼ each stage so that when mitosis comes around it is double the size. This implementation is limited however as cell growth occurs due to protein synthesis which only takes place in G1 stage.

```
def growth(self):
    """
    At each stage of the cell cycle, the cell grows.

    Once a cell has been through each of the four stages, it will have
    doubled in size.
    :return: The grown cell
    """
    if self.stage == 1:                 # Increase original size by 1/4
        self.area *= 1.25
    elif self.stage == 2:               # Decrease by 1/4 to achieve original, then increase by 2/4
        sa = self.area / 1.25
        self.area = sa * 1.5
    elif self.stage == 3:               # Decrease by 2/4 to achieve original, the increase by 3/4
        sa = self.area / 1.5
        self.area = sa * 1.75
    else:                               # Decrease by 3/4 to achieve original, then double
        sa = self.area / 1.75
        self.area = sa * 2
    self.radius = math.sqrt(self.area/math.pi)
    return self.mitosis()
```

Figure 5.10: Proliferating cell growth

5.1.4.5 Mitosis

When Proliferative Cells enter M phase of the cell cycle they undergo mitosis. This is where the parent cell replicates and divides into two equally sized daughter cells [3]. This function checks to see if the cell has entered M phase. If true it sends the cell to be split. If false, and the cell must be in another stage of the cell cycle and the function will increment stage of the cycle by 1. Returning either the two new daughter cells or the original cell further along in the cell cycle.

```
def mitosis(self):
    """
    Cell enters M phase.

    When the cell has passed through G0, G1, and G2, it will enter M phase
    where it splits into two identical daughter cells of half size.
    :return: Either unchanged cell or a new daughter cell
    """
    if self.stage == self.max_stage:
        new = self.split_cell()
    else:
        self.stage += 1
        new = None
    return new
```

Figure 5.11: Check to see if proliferating cell can undergo mitosis

5.1.4.6 Split Cell

When the cell is undergoing mitosis, it splits into two equally sized daughter cells [3]. This is achieved by reducing the area of the current (parent) cell by half and creating a new proliferative cell next to the current cell with the same area and radius but with a turnover of 1. As the parent cell has proliferated, its telomeres have shortened and to reflect this the turnover is increased by 1.
After the parent cell has divided it enters G1 phase and this is reflected by setting its stage back to 1.

```
def split_cell(self):
    """
    During mitosis the cell divides into two daughter cells.

    The two cells are now half the size of the orignial cell
    :return: The new daughter cell
    """
    self.area /= 2
    self.radius = math.sqrt(self.area/math.pi)
    new_cell_pos = [self.pos[0]+random.uniform(-1, 1)*self.radius, self.pos[1]+random.uniform(-1, 1)*self.radius]
    new_cell = pc(ID=self.num_pc, stage=1, pos=new_cell_pos, direc=random.random() * 2 * math.pi,
                  turnover=1, radius=self.radius, area=self.area)
    self.stage = 1
    self.turnover += 1
    self.pos = [self.pos[0]+random.uniform(-1, 1)*self.radius, self.pos[1]+random.uniform(-1, 1)*self.radius]
    print('new proliferating cell created with cell ID = %s' % str(new_cell.ID))
    return new_cell
```

Figure 5.12: Proliferating cell undergoing mitosis, creating 2 daughter cells

5.1.5 Agent Solve

A crucial aspect of agent based models is the application of rules (behaviours) each iteration. The agent solve class is used for just that. It is called each iteration from the main CellABM class and it takes the environment, containing the numbers of each type of agent, as its one parameter. The implementation of this class has been adapted from the original to decrease

types of environment to one and has been extended as per Figure 4.4 to include the logic for the new agents.
For each senescent agent, it only checks to see what stage the cell is at. If it has reached its max stage (3 years) the cell will be killed.
For each proliferative agent, it starts by testing whether the cell can turn senescent, if false it will test to see if the cell can turn quiescence, if false it will migrate the agent and then test to see if its smaller than the minimum allowed radius
For each quiescent agent, it starts by testing whether the cell can turn senescent, if false it will test to see if the cell turn proliferative again. It will then migrate the cell.
Each new agent created by agent solve is added to a list of new cells before the next iteration and all agents are checked to see if they're alive, removing them from the environment if not.

```
def agent_solve(env):
    """
    Goes through each agent applying their rules each iteration.

    :param env: Contains the current agents on the simulation
    :return: The updated states for each agent
    """
    new_senescent_cells = []  # List of new senescent cells created for this iteration
    new_proliferating_cells = []  # List of new proliferating cells created for this iteration
    new_quiescent_cells = []  # List of new quiescent cells created this iteration

    for agent in env.senescent_cells:
        if agent.stage == agent.max_stage:
            agent.kill_cell()
        if not agent.messages.dead:
            agent.growth()

    for agent in env.proliferating_cells:
        senescence = agent.senescence()
        if senescence is not None:
            new_senescent_cells.append(senescence)
            continue
        if agent.iscluster is True:
            quiescence = agent.quiescence()
            if quiescence is not None:
                new_quiescent_cells.append(quiescence)
                continue
        agent.migrate(env)
        agent.apoptosis()
        if not agent.messages.dead:
            new = agent.growth()
            if new is not None:
                new_proliferating_cells.append(new)

    for agent in env.quiescent_cells:
        senescence = agent.senescence()
        if senescence is not None:
            new_senescent_cells.append(senescence)
            continue
        if agent.iscluster is False:
            proliferating = agent.proliferating()
            if proliferating is not None:
                new_proliferating_cells.append(proliferating)
        agent.migrate(env)

    # Add new agents to list
    env.senescent_cells.extend(new_senescent_cells)
    env.proliferating_cells.extend(new_proliferating_cells)
    env.quiescent_cells.extend(new_quiescent_cells)

    update_messages(env)
```

Figure 5.13: Function which applies rules to each agent each iteration

5.1.6 Environment

5.1.6.1 Create Agents

This function has been adapted from the original to include cell stages and to incorporate the new agents. For the user defined number of starting senescent and proliferating cells, the function will create a new cell of that type with a stochastic radius, position and stage within ranges. Quiescent cells have not been implemented in this function as they are an emergent behaviour that occurs when a monolayer has formed.

```
def create_agents(self, nsc, npc):
    """
    Populates simulation with starting cells.

    Creates a user defined number of senescent and proliferating
    cells, giving them a random size (within a range), a random position,
    a random stage (age, within a range), a random direction, a set
    turnover (number of times divided and appends them to the list
    of cells.
    :param nsc: User defined starting number of senescent cells
    :param npc: User defined starting number of proliferating cells
    :return: Three lists containing the different starting cell agents
    """
    senescent_cells = []
    proliferating_cells = []
    quiescent_cells = []
    print(nsc, npc)

    for s in range(nsc):
        ID = s
        radius = random.randint(10,50)
        area = math.pi*(radius*radius)
        pos = [radius+(round(rand(), 3))*(self.size-(2*radius)),
               radius+(round(rand(), 3))*(self.size-(2*radius))]
        stage = np.ceil(rand()*4380)
        direc = rand()*2*np.pi
        turnover = 1
        senescent_cells.append(sc(ID, stage, pos, direc, turnover, radius, area))


    for p in range(npc):
        ID = p
        radius = random.randint(5,10)
        area = math.pi*(radius*radius)
        pos = [radius+(round(rand(), 3))*(self.size-(2*radius)),
               radius+(round(rand(), 3))*(self.size-(2*radius))]
        stage = np.ceil(rand()*8)  # 4)
        direc = rand()*2*np.pi
        turnover = 1
        proliferating_cells.append(pc(ID, stage, pos, direc, turnover, radius, area))

    #if list type is seperate (each agent type has its own list)
    self.senescent_cells = senescent_cells
    self.proliferating_cells = proliferating_cells
    self.quiescent_cells = quiescent_cells
```

Figure 5.14: Stochastic creation of initial agents

5.1.6.2 Wound Creation

This function is used when the first confluence is formed to create the simulated wound. The wound is created across the whole Y axis but only by a user defined length across the centre of the X axis. The size of the wound can be altered on the command line before the simulation is run. Any cells that are

within the x1 and x2 range (where x1 + x2 = wound size) are removed from the simulation using the .kill_cell() method.
Special consideration has been given to the creation of chained comparisons with Figure 5.16 being the desired implementation over Figure 5.15.

```
for n in range(len(self.senescent_cells)):
    if self.senescent_cells[n].pos[0] > x1 and self.senescent_cells[n].pos[0] < x2:
        self.senescent_cells[n].kill_cell()
```

Figure 5.15: Inefficient chained comparison

```
for n in range(len(self.senescent_cells)):
    if x1 < self.senescent_cells[n].pos[0] < x2:
        self.senescent_cells[n].kill_cell()
```

Figure 5.16 PEP8 standard for chained comparisons

5.1.7 Overlap Correction

This class has mainly remained unchanged from the original, with new logic to count the number of neighbours each cell has being implemented. It works by using a brute force approach to correct the overlap. First each cell from the environment is added to a list 'cells'. This list of cells is iterated through in turn with each cell i compared with each other cell j to see if their current position on the environment and the size of each cell causes them to overlap. A list of overlapping cells is then created and passed to a correct overlap function where the cells are assigned new positions to ensure they no longer overlap. However, moving the cells to a new position can cause them to overlap with another cell and so the process must be repeated until no cells are overlapping.

5.1.8 Confluence Detection

A novel approach has been taken for the detection of confluences. It works off the emergence behaviour of proliferating cells turning quiescence, which only occurs when the cell density is high enough to prevent proliferation and creation of new cells. When the total number of quiescent cells have passed a threshold, a confluence has formed. Here the threshold has been set to be $1/4$ the number of proliferative cells as $1/5$ caused the program to sometimes detect a confluence even when there were significant gaps, this was especially true in simulations with higher senescence.
$1/6$ wasn't as bad as $1/5$, however would take more iterations than $1/4$ in forming a confluence.
After the first confluence, the wound is simulated and it takes an iteration or two for the quiescent cells to notice the extra space and change back to proliferative cells hence the if time > 2 condition.
When the second confluence has occurred, the simulation is halted and the total time for the wound to heal is output.

```
if qc.num_qc >= (pc.num_pc/4):
    if counter == 0:
        env.wound(wsize)  # Remove a strip of cells
        timer = n_it
        counter += 1
    else:
        time = n_it - timer
        if time > 2:  # Quiescent cells take an iteration or two to differentiate back to proliferating cells
            print("CONFLUENCE DETECTED, time taken: %s itteration == %s hours." % (time, time*6))
            print('Senescent cells = %s | Endothelial cells = %s | Quiescent cells = %s' % (sc.num_sc,
                                                                                            pc.num_pc,
                                                                                            qc.num_qc))

            num_cells[0, n_it] = sc.num_sc
            num_cells[1, n_it] = pc.num_pc
            num_cells[2, n_it] = qc.num_qc
            plot_2d(env, directory, labels, n_it)
            growth_curve(num_cells, directory)
            sys.exit("End")
```

Figure 5.17: Confluence detection within CellABM class

5.1.9 Command Line Interface

The program does not utilise a graphical user interface and therefore all conditions for the simulation must be given at the start on the command line. The conditions that can be changed are: the size of the model (size), the starting number of senescent cells (nsc), the starting number of proliferating cells (npc), the number of iterations to model (steps), the size of the wound (wsize), and the name of the directory to save the output graphs (directory). These conditions are then passed through the program to where they are needed.

```
def CellABM(size, nsc, npc, steps, wsize, directory, freq=0, labels=False):
```

Figure 5.18: command line definition within CellABM class

```
In [15]: CellABM(200, 1, 10, 50, 100, "Simulation 1.2")
```

Figure 5.19: Example of how to call the program with a size of 200µm^2, 1 senescent cell, 10 proliferative cells, 50 iterations and a wound 100µm wide.

5.2 Overview of Parameters

Parameter Name	Value	Source
Time Step	6 hours	Estimate
Proliferating Cells		
Min Radius	5µm	[7]
Speed	1µm / min	Estimate
Direction	Random	Estimate
Growth Rate	Double during cycle	[3]
Max Proliferation	50	[10]
Cycle Time	24 hours	[3]
Num Stages	4	[2]
Senescent Cells		
Min Radius	5µm	[7]
Speed	0	Domain Expert
Direction	0	Estimate
Max Age	3 years	[8]
Growth Rate	10 times in 2 weeks	Domain Expert
Quiescent Cells		
Min Radius	5µm	[7]
Speed	0	Estimate
Direction	0	Estimate
Max Age	10 years	[6]

Table 5.1: Values associated with the parameters for the program.

5.3 Testing

The testing for this project has been divided into two sections, first being unit testing of the code containing the rules which affect the agents, and second being face validation of basic simulations to ensure the predicted behaviour acts like the observed behaviour.

5.3.1 Unit Testing

Unit test have been developed for the cell rules outlined in Chapter 4. This is to ensure that each agent changes state only under the correct conditions and new cells created start with the correct parameters. These tests have been created using the Python module unittest which allows for rapid development of automated tests, using inbuilt functions to check outputs.
In total 12 test have been created to ensure correct functionality of cell rules and are outlined below.

```
paaa:CellABM_student_ver harrycooper$ python -m unittest discover
............
----------------------------------------------------------------------
Ran 12 tests in 0.001s

OK
```

Figure 5.20: All unit tests passing.

Name	Expectation	Result
Senescent growth	If the cell is smaller than max size, increase radius by 0.8μm and increase cell age, otherwise just increase cell age.	Pass
Quiescent to Proliferating	When proliferation is possible, change back to a PC with current parameters.	Pass
Quiescent to Senescent	When cell is old enough, changes to a SC and starts to grow.	Pass
Proliferating to Quiescent	When proliferation is not required, change into a QC with current parameters.	Pass
Proliferating to Senescent	When cell is old enough, changes to a SC and starts to grow.	Pass
Proliferative Mitosis	Produce two identical daughter cells when parent cell is in M phase, ensuring daughter cells are half the size of the parent cell.	Pass
Proliferative Growth	Over the 4 stages of the cell cycle, increase the cells area to double its original size.	Pass

Table 5.2: Unit testing results.

5.3.2 Face Validation

Here the behaviours of the implemented rules are shown on a micro scale and compared against what is expected. The theory behind these tests is that if the rules work under basic conditions they will still work when scaled up to a full-size simulation. The simulations will involve a low number of cells, around 1 to 10, and will be simulated for the least amount of time required to observe the desired behaviour.

This simulation ensures proliferating cells undergo mitosis correctly. It is set up with one proliferating cell with a starting stage of 1 and is expected that on iteration 4 there will be two cells next to each other (mitotic division) each the same size as the cell in the first iteration.

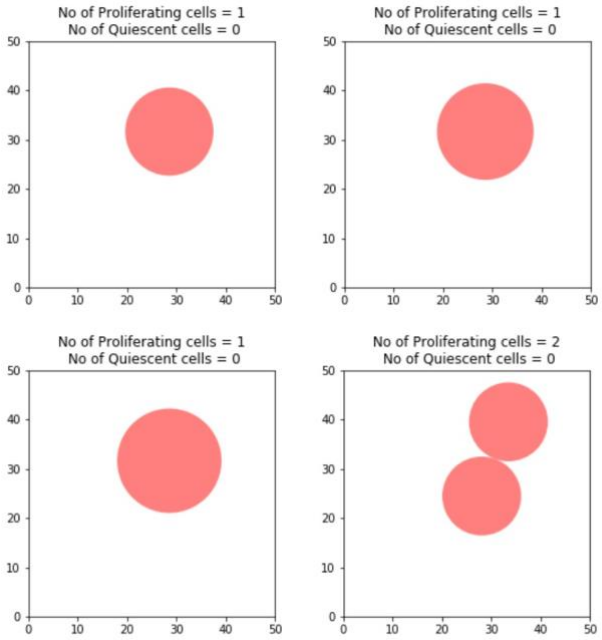


Figure 5.21: Proliferating cell undergoing mitosis.

The next simulation tests to ensure a proliferating cell will turn senescent when it has hit the proliferation limit. It has been run with one starting PC with a turnover of 49 (1 below the Hayflick limit [10]). It is expected that on iteration four the cell will undergo mitosis, dividing and increasing its turnover to 50, therefore turning into a senescent cell.

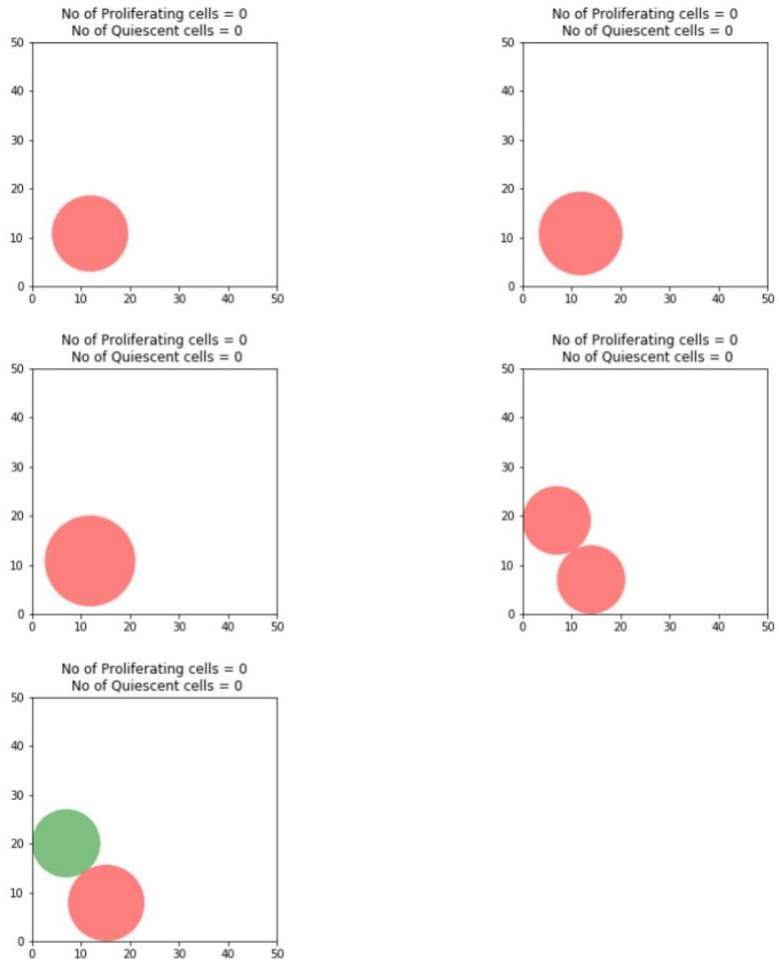


Figure 5.22: Proliferating cell turning senescent.

This simulation ensures a proliferating cell will turn quiescent when proliferation is not required. It is expected that one of the PCs will turn into a QC due to the confluence formation. As quiescence is an emergent behaviour that occurs when a cell is surrounded by a certain number of cells and is unable to move, it is difficult to test on the micro scale and in this case, was formed by overfitting the environment with cells.

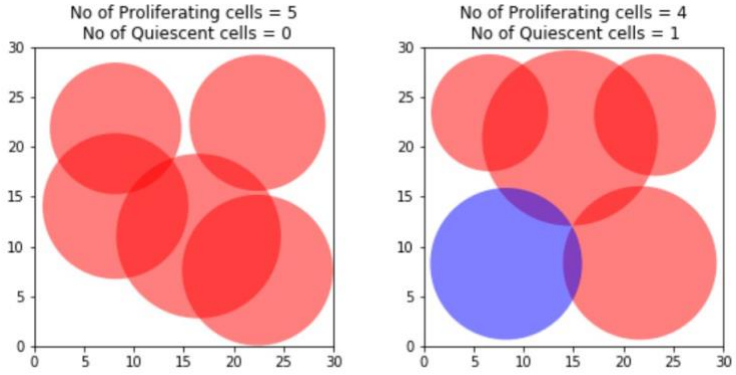


Figure 5.23: Proliferating cell turning quiescent.

This test ensures that senescent cells grow to the correct size. The simulation was started with one SC with a radius of 5μm and it is expected that by iteration 56 (2 weeks) it will have reached a radius of 50μm.

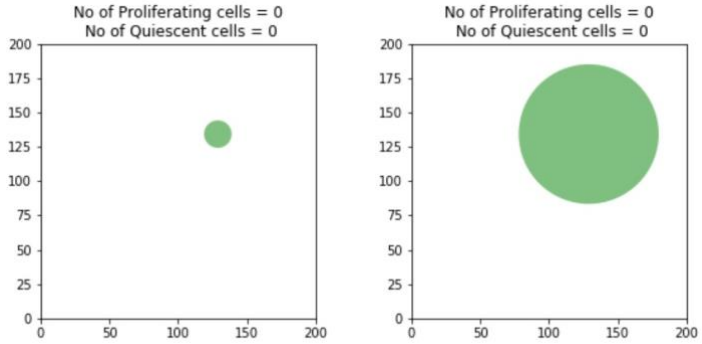


Figure 5.24: Senescent cell growth.

For the sake of testing, a single quiescent cell was simulated by adapting the environment class to allow for quiescent cells to be input from the command line. This test is to ensure that a quiescent cell will start to proliferate if there's space. It is expected that the cell will swap back to a proliferating cell the next iteration due to the lack of external pressures as explained in [4].

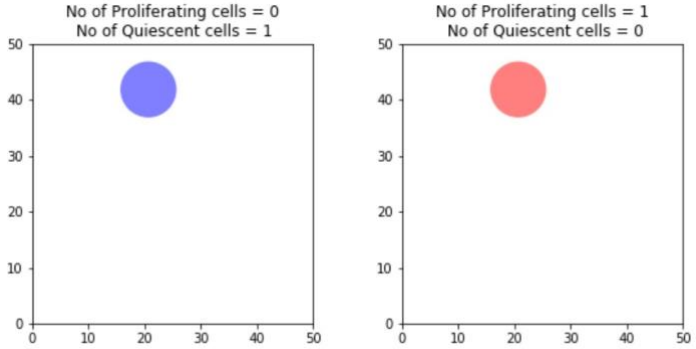


Figure 5.25: Quiescent cell starting to proliferate.

Following, it was tested to see if the QC would correctly differentiate into a SC (before turning into a PC) if it was at its maximum age. This initially brought up an error in the program where due to the lack of surrounding neighbours and being at max stage the cell passed the conditions required for both turning senescent and quiescent. The program first removed the QC and initialise a new SC but would then go onto remove the SC and initialise a new PC in the same iteration. This was due to a missing continue statements in the agent solve function and has now been fixed as shown below.

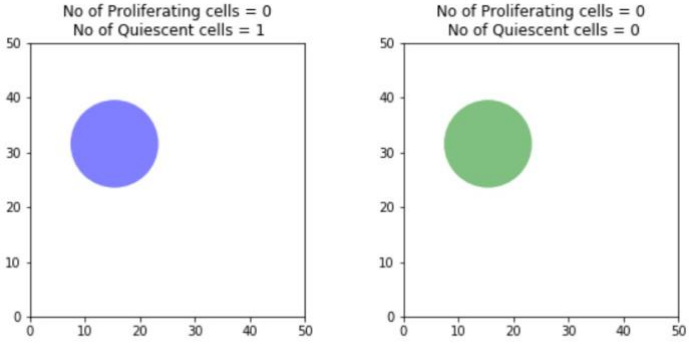


Figure 5.26: Quiescent cell turning senescent.

6 Results and Discussion

All simulations were run on a 2015 15" mac book pro with a 2.8GHz Intel core i7 processor and 16GB 1600MHz DDR3 memory and a MSI GT70 with a 2.4GHz Intel core i7 processor and 16GB DDR3 memory. The software developed in Chapter 5 and all simulations run can be found on GitHub at: https://github.com/HarrisonCooper/dissertation. In all simulations, green circles are senescent agents, blue circles are quiescent agents, and red circles are proliferating agents.

6.1 Wound Healing Rate Results

Following Chapter 4.5, several simulations with the same starting conditions were run to provide a statistically accurate representation of the emergent behaviours for the stochastic ABM produced. However, due to time complexity issues it was not possible to accurately copy the process in [27] where they used a $1mm^2$ area of endothelial cells and a wound 400μm wide. Instead most simulations were run at $500\mu m^2$ with a wound size of 200μm, generally producing around 800 agents, requiring between 90 and 150 minutes to complete. One simulation has been run with the same dimensions as [27], taking 1,800 minutes to run due to simulating 3,433 agents.
The time step of each iteration is 6 hours and each simulation was initialised with 50 proliferating cells and a varying number of senescent cells to achieve the desired percentage senescence at confluence. Results are in Appendix Tables A.1 to Table A.6.

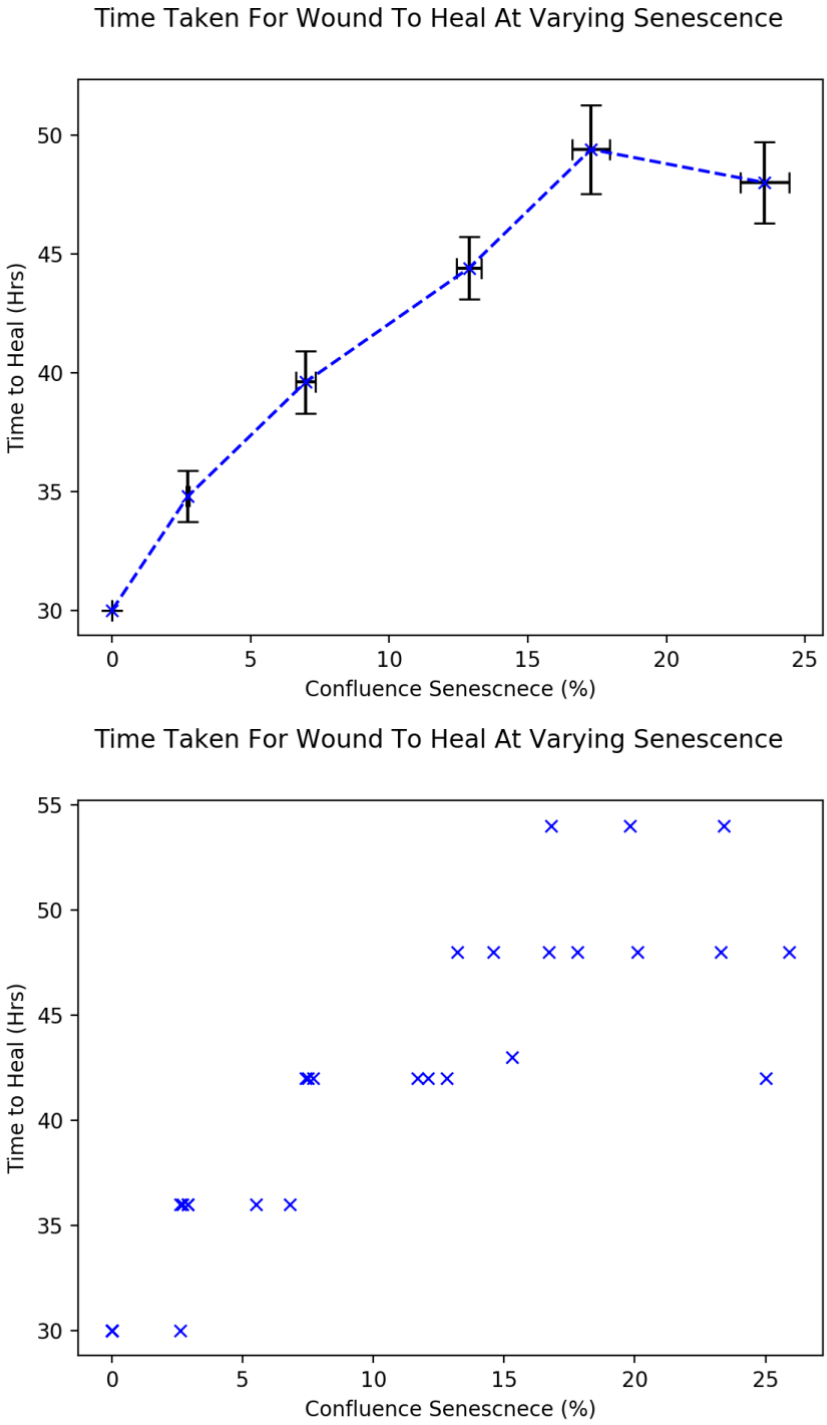


Figure 6.1: Time for 200µm wound to heal with varying levels of senescence. Top: Samples grouped into bins of 0%, 0-5%, 5-10%, 10-15%, 15-20%, and 20-25% senescent and averages plotted. Bottom: raw values from simulations plotted.

Figure 6.1 supports [14]'s suggestion that senescent cells impair wound healing, by showing us that as senescence is increased, time taken to heal is increased on average at a linear rate between 0 and 17% then levels out between 17 and 24%. This implies that regions above 17% senescent have no added detrimental effects to wound healing. Applying this to the primate paper [13] which examined baboons aged between 5 and 30, puts 17% senescence around 27 years old, which is towards the end of most baboons lives. We therefore have this linear relationship between age and wound healing but an exponential relationship between age and percentage senescence. Thus, the older you get the more detrimental wounds are to your health.

Data from Tables A.1 to A.6 can be used to calculate the average speed of migration for each category of senescence and is shown in Table 6.1. The speeds were calculated by taking the wound size of 200µm and dividing that by the average healing time for each category. This helps to further understand that as senescence is increased average migration speeds decrease. This is important as an increase in wound healing time can lead to localised inflammatory responses and thrombosis [14] and so older individuals will be at an increased risk of these heart associated problems. These results are generally similar to [27] where average cell migration speeds were observed to be 8.35µm per hour and [28] where migration speeds varied from 84 to 20µm per day (3.5 to 0.8µm per hour), slowing down as time progressed. These two in vitro observations place the programs migration speed predictions in-between the two, providing a high level of validity.

Percentage Senescent	Speed (µm/hr)
0	6.67
0-5	5.75
5-10	5.05
10-15	4.50
15-20	4.04
20-25	4.17

Table 6.1: Average migration speed for each test.

Figure 6.2 shows us the number of cells that have migrated into the wound every 6 hours. Higher senescent simulations show fewer cells entering the wound over the whole healing process. This is due to senescent cells having a larger area than proliferating and quiescent cells and so fewer total cells can be fit onto the environment. This is validated by each subsequent simulation with decreased levels of senescence showing a higher total number of cells than the simulation prior. An interesting feature of this figure is that the lower the average senescence the steeper the gradient of cell migration, showing increased migration speeds when senescence is lower.

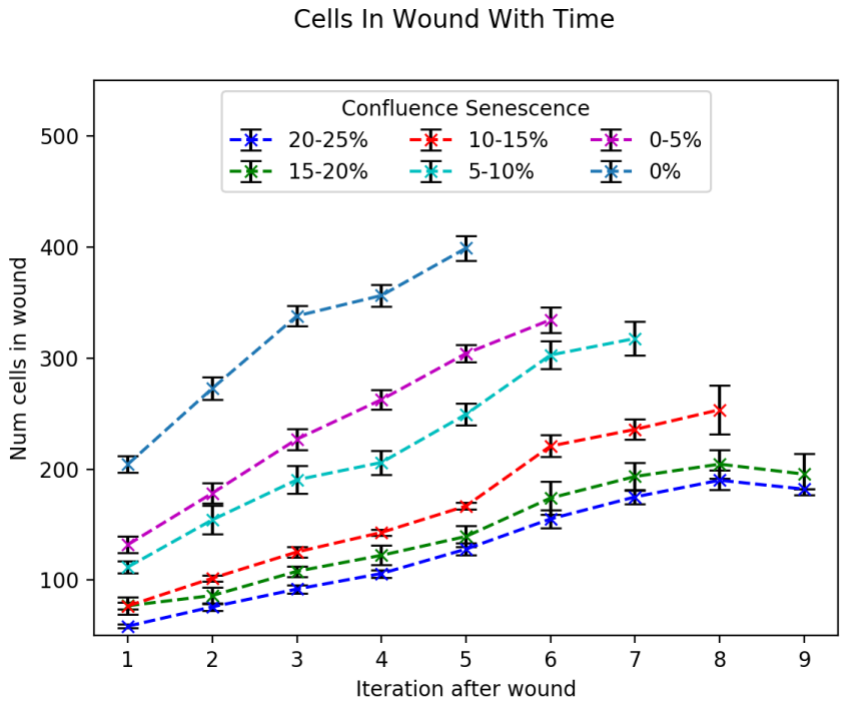


Figure 6.2: Number of cells to fill the wound each iteration.

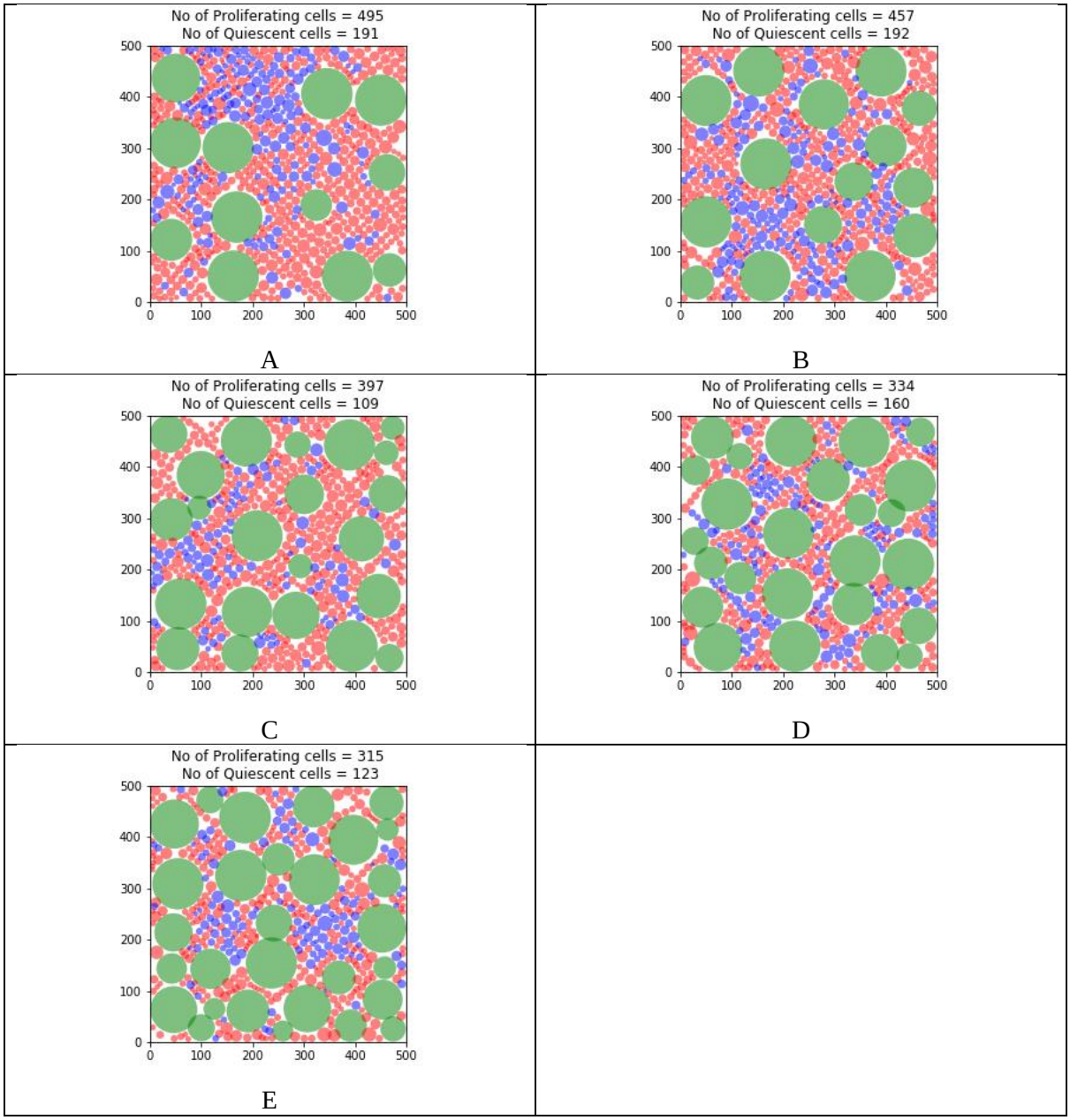


Figure 6.3: Figures A-E showing the final iteration from each sample. A: 0-5%, B: 5-10%, C: 10-15%, D: 15-20%, E: 20-25%

Figure 6.3 shows the final image of one of the simulations run for each category. You can see that as senescence is increased fewer proliferating and quiescent cells are present. And in each case a full confluence was formed showing the model works for high levels of senescence. It must be noted, however that these simulations produce some edge cases, such as Figure D, where due to the lack of simulated cells outside the environment there can be significant gaps (left side) in the monolayer that aren't present in reality; this can be seen as a limitation of the model.

Following the images in Figure 6.4, we can see that after the wound is formed, cells start to migrate into the wound as expected. Two iterations (12 hours) after the wound, all the quiescent cells have changed back into proliferating cells as there is once more space for proliferation. Over the successive iterations, the density of the wound starts to increase as more cells migrate in, and due to no cells being simulated outside the environment the density at the edges of the simulation decreases. As the senescent cells are so large, the proliferative cells behind them are unable to easily migrate into the wound space, slowing down the total migration.

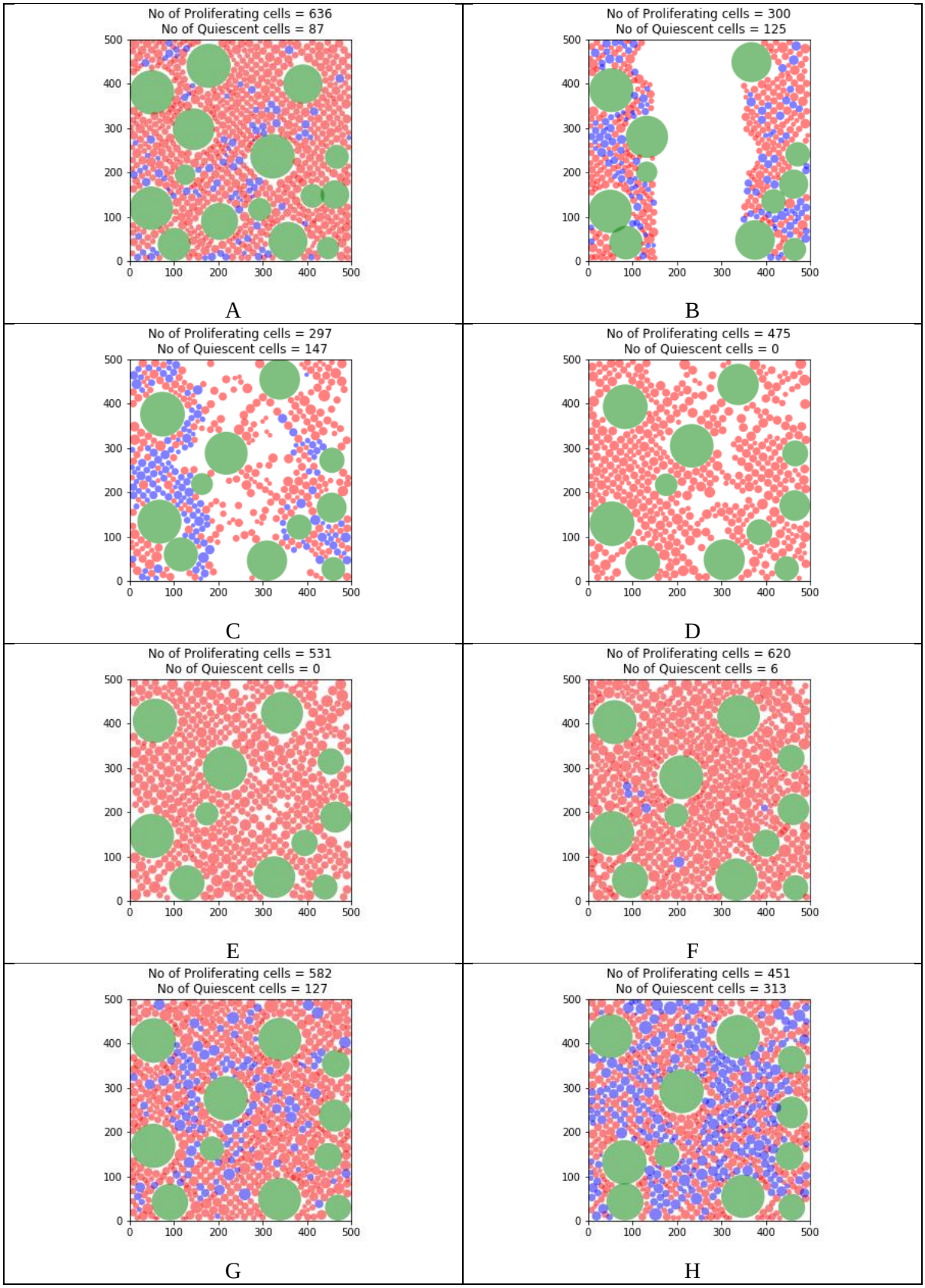


Figure 6.4: Figures A-H showing the iteration before wounding, the wound, and subsequent iterations after wounding until confluence formation.

6.2 Simulations with 1 hour time steps

Simulations were run with identical starting parameters to those in 6.1 but with a decreased time step. Due to the computation time, only two simulations were run for each category and so stochastic elements in the results may still be present due to the low sample size. It does, however, help to visualise the cell movements in forming the monolayer and migration of the cells into the wound. Results are in Appendix Table A.7 to Table A.12.

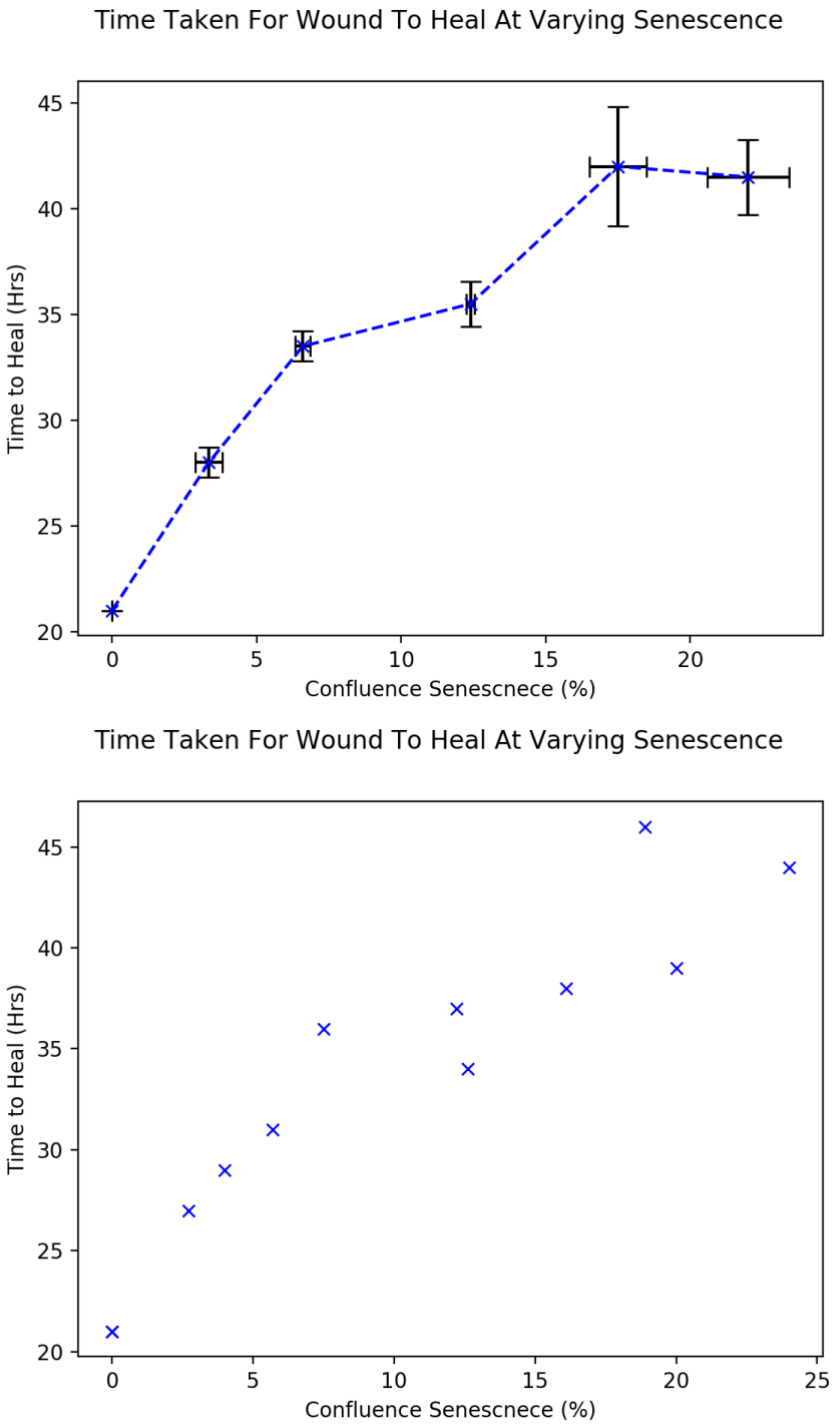


Figure 6.5: Time for 200µm wound to heal with varying levels of senescence with 1 hour time steps. Top: Averages for each category used. Bottom: raw values from simulations run.

Figure 6.5 supports the findings in figure 6.1, but also provides a higher level of insight into the rates of wound healing with time, even though only two simulations were run for each category.

Figure 6.6 follows the same trends as figure 6.2 but provides further insights into the rate of healing for each category with time. Even though only two samples were used for each category, many more data points were produced. It is interesting to note that towards the end of the healing, the rate of cell migration plateaus for each category, and the lower the average senescence the later this plateau occurs. Looking at the graph, the largest change occurs when senescence surpasses 5%, and applying the ages found in the primate paper [13] 5% senescence equates to an age around 16 years which is just over half the life expectancy of the average baboon. Therefore, taking this over to a human with an average life expectancy of 71.4 years [31] wound healing is significantly decreased beyond the age of 35.

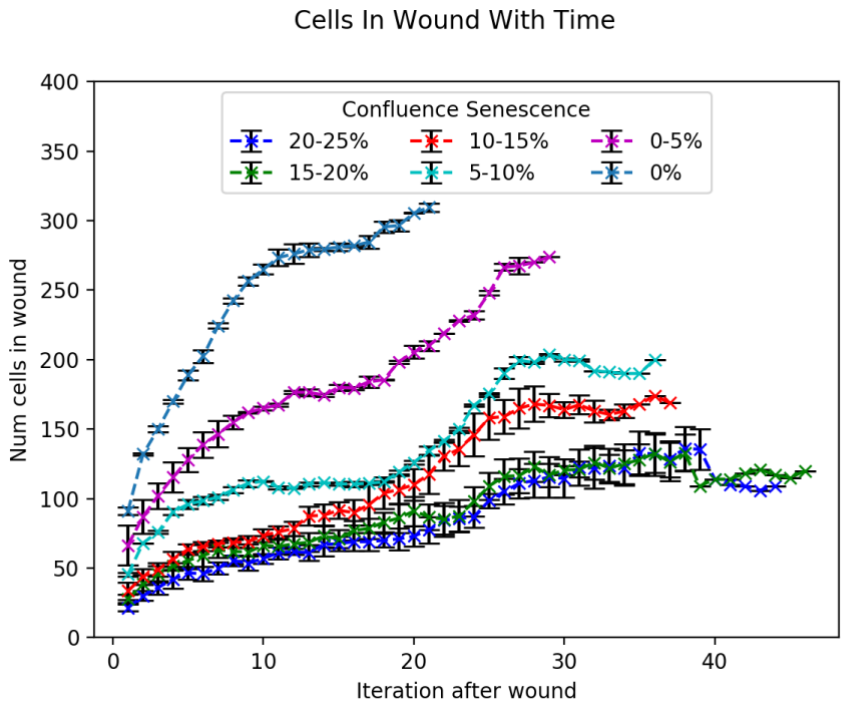


Figure 6.6: Number of cells in wound each hour

6.3 Sensitivity Analysis

Two local sensitivity analysis tests have been run. The first varies the migration rate of the proliferating cells by doubling and halving the max speed parameter of the proliferating class. The second varies the rate of mitosis by setting the cell cycle (max stage) for the proliferating cells to 2 then 8.

The predictions have been run using the same starting conditions as the simulations run in 6.1, with an area of 500μm^2, a wound size of 200μm, and initial population of 50 proliferating cells and varying number of starting senescent cells. Looking at the results in Appendix Table A.13 to Table A.18 changing the rate of mitosis has a far greater impact on the predictions than changing the rate of migration. However, changing the mitosis rate had negative impacts such as doubling the rate would significantly reduce the total percentage of senescent at wounding and halving the rate would significantly increase the total percentage at confluence.

The implications of these results are that the rate of mitosis is a far more sensitive parameter than migration rate and more experiments should be run with smaller changes to the mitosis rate. Since the time step of each iteration is an assumption and rate of mitosis is based on the time step it shows that the time step assumption is one of the more important estimates and therefore more experiments

varying the time step should be run.

6.4 Program Efficiency and Runtime Analysis

The runtime of CellABM is dominated by the overlap correction function. Here each cell is compared to each other cell to find the distance between the two and update their positions if they're overlapping. This algorithm is $O(N^2)$ and is not a problem for small simulations, such as 500μm^2 where there is maximum of around 800 cells, taking 2 hours to complete. However, larger simulations, such as 1mm^2 can have as many as 3,500 cells, taking 30 hours to complete the simulation.
Figure 6.7 was produced by recording the number of agents each iteration and how long it took the program to produce the output image. These data points were plotted and a curve of best fit produced and extrapolated out to 5,000 agents, showing the $O(N^2)$ relationship.
This overall high level of time complexity is a limitation of the model as it prevents multiple large scale simulations being produced in a reasonable time, however could be improved by either using a high-end computing cluster or rewriting the overlap function to only look at cells close to each other rather than comparing each cell to each other cell.

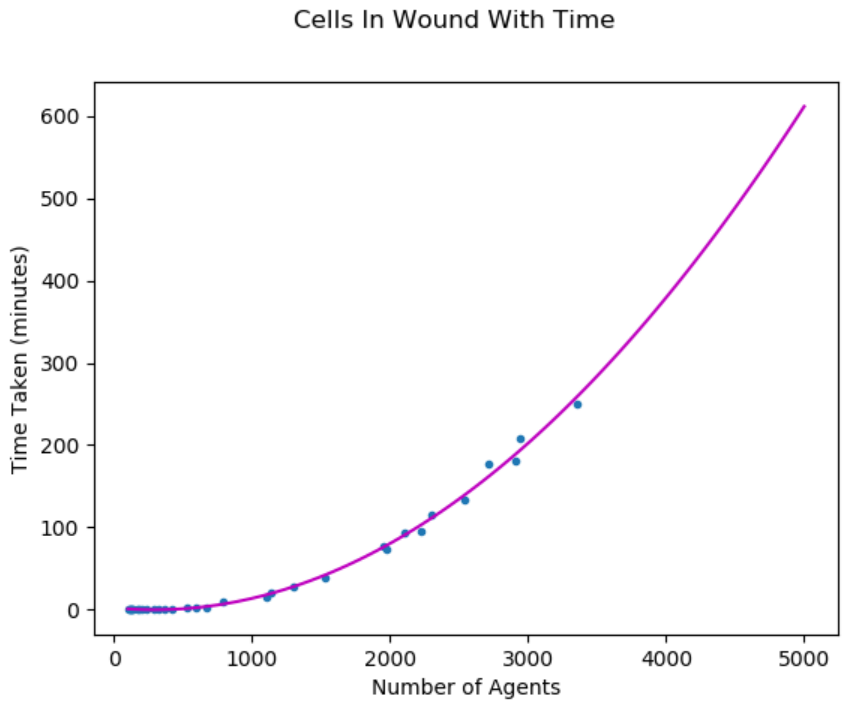


Figure 6.7: Time complexity of program.

6.5 Feedback From Domain Expert

After the initial development of the above results, I met with my domain expert Prof. Paul Evans for his input on the predictions the project produced and whether there was anything that could be adapted. He suggested several sensitivity simulations that could be run including changing the migration speed of the proliferating cells and the rate of mitosis. Another suggestion was to add a control group which would have 0% senescent cells and has now been included above.
The domain expert was particularly pleased with the results in Figure 6.1 and Figure 6.2 as these correlated to observations within his own research [14], however, more simulations between 0-1% senescence would be beneficial as that was the range used in the research.

Finally, the domain expert noted that the growth of the senescent agents is incorrect. Currently the model increases the radius of the cells each iteration, however, in reality senescent cells will suddenly grow then remain the same size before enlarging again some time later. This behaviour can be included into the model by adapting the code in chapter 5.1.2.2 to increase the cells size every 4 iterations (24 hours) rather than every iteration (6 hours).

6.6 Goals Achieved

The predictions of the model produced are like those found in vitro, and supports the theory the domain expert had that increased senescence with age decreases wound repair rate.
Going through the requirements in Tables 3.1 to 3.5, all 11 have been fully met. In regard to requirement 1) the two time scales used for the project were 6 and 1 hour steps. These are appropriate as 6 hours allows for explicit representation of each stage of the eukaryotic cell cycle, whereas the 1 hour step allows for a more granular visualisation of cellular migration. Requirement 2) and 5) are fulfilled by creating the wound after a quarter of total cells in the simulation have turned quiescent, ensuring a confluence has formed. Senescent cells have been represented as agents with a speed of 0 as so fulfils 3). Requirement 4) has been partially met as due to the stochastic nature of the proliferating cells it is difficult to predict the precise senescent percentage at confluence. Requirement 6) is fulfilled at the end of the simulation where the program returns the number of iterations between wound formation and healing. Requirement 7) has been completed as at the end of each iteration an image is created and saved showing the positions of each agent in the simulation. Successive images can be strung together into a video to show migration behaviours. Requirement 8) is fulfilled by stopping the simulation when a quarter of the cells have turned quiescent after the wounding. Requirement 9) has been fulfilled by giving senescent cells a stage which is incremented each iteration, when this has reached the max stage (3 years) the cells are removed from the simulation. Requirement 10) has been completed as the user only needs to provide 6 parameters to the program to start it. Requirement 11) has been fulfilled by writing java doc style comments for each class and function in the program, stating the logic of the function, the input arguments and what the function returns.

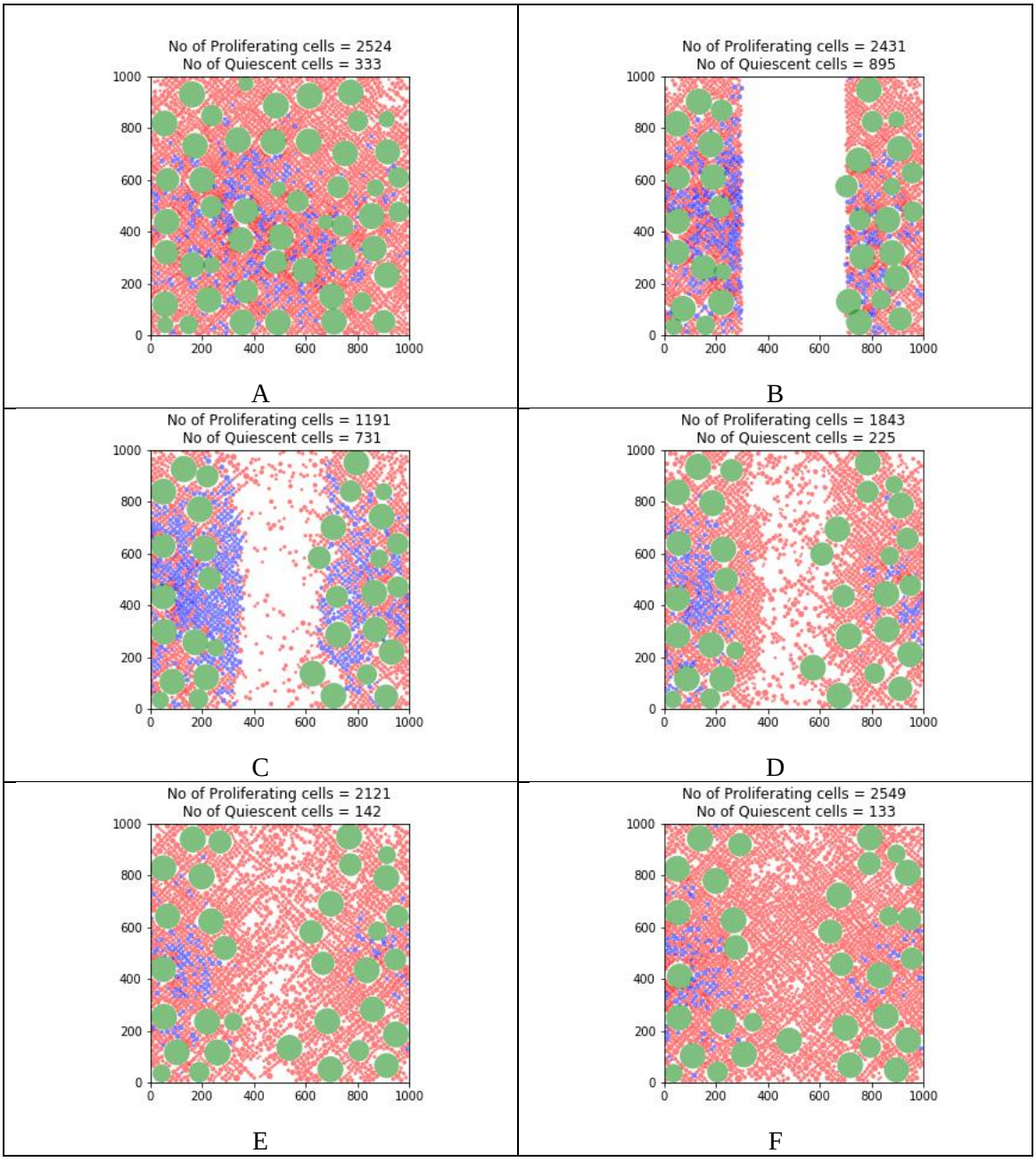
6.7 Further Work

The predictions the program makes in relation to the rate of wound healing with age is interesting as they follow what was predicted to happen, however due to a lack of in vitro data surrounding senescence migration rates it is not possible to validate the model to a rigorous level. Therefore, it would be useful to carry out in vitro experiments which match the simulations and compare the results.

Due to time constraints of this project, cell adhesion was not implemented but is an integral part of wound healing, therefore further work to demonstrate the effect (if any) cell adhesion would have on time taken for the wounds to heal would be of interest. This could be achieved by extending the current detection of neighbouring cells to make neighbours stick together.

One simulation has been run at the same size as the in vitro experiment in [27] mainly due to the time complexity meaning this simulation took 30 hours to compute. Images of the iteration before wounding, the wound itself and each subsequent iteration (6 hours) until fully healed is shown in Figure 6.8. It is interesting to note that similar emergent behaviours are observed here (on a larger scale) to the simulations run in 6.1, namely after the wound has occurred it takes a few hours for the quiescent cells to change back into a proliferating cell. Image C seems to show edge cases where the cells at the top and bottom migrate slower than the cells towards the centre. I believe this is down to there being a higher percentage of the cells towards the centre of the simulation and due to the overlap function, before the wound, the cells are bunched up and slightly overlapping and when there's space they quickly spring out to correct the overlap. As there's more cells in the middle, there's a higher force pushing, hence the faster movement.

It would be interesting to rerun the above simulations in 6.1 on this area of cells as this would more closely follow the in vitro experiments of scratch assays [27] and so can be used to validated the program further. Also, setting the time step to 1 hour instead of 6 hours would be beneficial as it will help to further understand the movement of cells into the wound over time, and [28] could then be used to validate if the cells do in fact migrate faster in the first few hours after wound creation, subsequently slowing down as healing progresses. This could be achieved by decreasing the time complexity of the overlap class or by running the simulations of a high-end computing cluster.



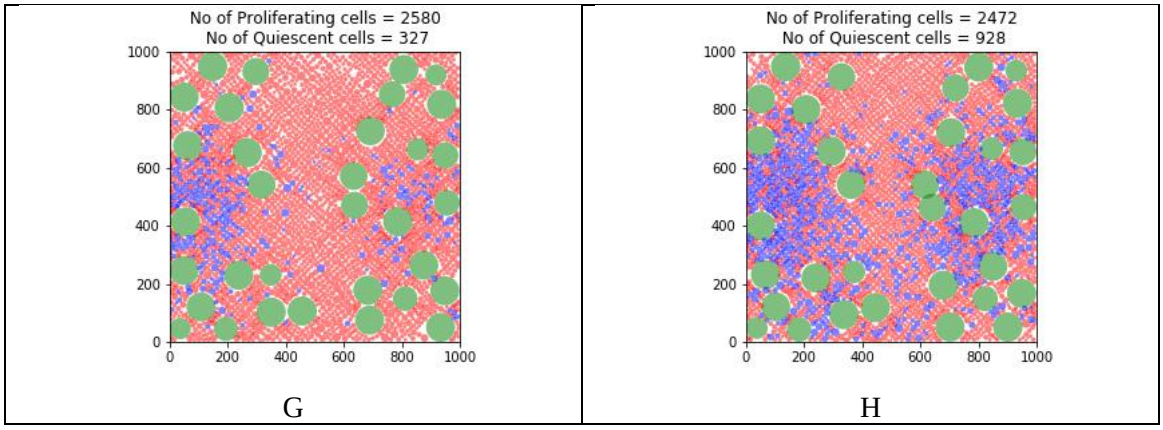


Figure 6.8: A: The iteration before wounding, B: The iteration of the wound, C-H: images of the healing every 6 hours.

7 Conclusion

This project began by looking at the biological processes involved with the aging of endothelial cells then looked at several methodologies for modelling these processes computationally to provide insight into the effect aging has on wound healing. Research into the usefulness and limitations of cellular automata, equation based models, and agent based models was carried out, and it was determined that an agent based model would be most appropriate for this project due to the individual interactions between cells, the stochastic nature of wound healing, and the ability to produce a visual output of the migration of cells over time.

The program CellABM, developed by Marzieh Tehrani, was adapted and extended to implement the three types of agents required in the model and multiple rules have been developed to mimic the natural behaviours of these cells observed in vitro. Parameters associated with these rules were found in the literature and those that weren't were heuristically found over several simulations to find the most appropriate.

The program was run with varying starting number of senescent cells to produces differing senescent percentages when a confluence had been formed, with most simulations being run at $500\mu m^2$ with a 200μm wound and time steps of 6 hours. These simulations quantitatively showed that as senescence is increase, the time taken for the wound to heal increases with it. And that time to heal was most sensitive at lower percentages of senescence. Increasing senescence from between 0 and 10% had the largest effect on wound healing, and increasing senescence above 10% had little effect on the wound healing.
Simulations with a time step of 1 hour correlate with the 6-hour time step results but provide further insight into the rate of healing with time, showing that as the wound density increases the migration rate slows and starts to plateau, and increasing the percentage of senescent cells causes this plateau to occur later.
It can therefore be concluded that we are most effective at healing wounds when we are young, and as we age the total number of senescent cells increases and so our ability to heal wounds decreases.

The implemented model has several limitations, most notably the $O(n^2)$ time complexity of the overlap class preventing large scale simulations from being run in a reasonable time. There are also edge cases associated with the simulations as no cells are simulated outside of the environment so as migration occurs, gaps form at the sides of the simulation where in reality there would be cells further out migrating to fill these gaps.

Further work can be conducted to decrease the time complexity and therefore increase the size of the simulation. Also in vitro experiments of varying senescence to wound healing would provide validation to this model's predictions and modelling cell adhesion would provide more accurate results.

All functional and non-functional requirements for the project have been fully met.

References

[1] Pearson, J. (2000). Normal endothelial cell function. *Lupus*, 9(3), pp.183-188.

[2] Ncbi.nlm.nih.gov. (2017). *Figure 14.1, Phases of the cell cycle - The Cell - NCBI Bookshelf*. [online] Available at: https://www.ncbi.nlm.nih.gov/books/NBK9876/figure/A2435/?report=objectonly [Accessed 3 Dec. 2017].

[3] Cooper, G. (2000). *The cell*. Washington, D.C.: ASM Press.

[4] Cell Proliferation. (1991). *Index medicus*, [online] 24(1). Available at: http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1365-2184 [Accessed 3 Dec. 2017].

[5] En.wikipedia.org. (2017). *G0 phase*. [online] Available at: https://en.wikipedia.org/wiki/G0_phase [Accessed 3 Dec. 2017].

[6] Su, T. and O'Farrell, P. (1998). Size control: Cell proliferation does not equal growth. *Current Biology*, 8(19), pp.R687-R689.

[7] Lab.anhb.uwa.edu.au. (1998). *Blue Histology - more about Endothelial Cells*. [online] Available at: http://www.lab.anhb.uwa.edu.au/mb140/moreabout/endothel.htm [Accessed 27 Nov. 2017].

[8] P.Brandes, R. (2005). Endothelial Aging. *Cardiovascular Research*, [online] 66(2), pp.286–294. Available at: https://doi.org/10.1016/j.cardiores.2004.12.027 [Accessed 3 Dec. 2017].

[9] Foreman, K. and Tang, J. (2003). Molecular mechanisms of replicative senescence in endothelial cells. *Experimental Gerontology*, 38(11-12), pp.1251-1257.

[10] Senescence.info. (2017). *Cellular Senescence: The Hayflick Limit and Senescent and Aging Cells*. [online] Available at: http://www.senescence.info/cell_aging.html [Accessed 3 Dec. 2017].

[11] Dimri, G., Lee, X., Basile, G., Acosta, M., Scott, G., Roskelley, C., Medrano, E., Linskens, M., Rubelj, I. and Pereira-Smith, O. (1995). A biomarker that identifies senescent human cells in culture and in aging skin in vivo. *Proceedings of the National Academy of Sciences*, 92(20), pp.9363-9367.

[12] Wang, C., Jurk, D., Maddick, M., Nelson, G., Martin-Ruiz, C. and Von Zglinicki, T. (2009). DNA damage response and cellular senescence in tissues of aging mice. *Aging Cell*, 8(3), pp.311-323.

[13] Herbig, U. (2006). Cellular Senescence in Aging Primates. *Science*, 311(5765), pp.1257-1257.

[14] Warboys, C., de Luca, A., Amini, N., Luong, L., Duckles, H., Hsiao, S., White, A., Biswas, S., Khamis, R., Chong, C., Cheung, W., Sherwin, S., Bennett, M., Gil, J., Mason, J., Haskard, D. and Evans, P. (2014). Disturbed Flow Promotes Endothelial Senescence via a p53-Dependent Pathway. *Arteriosclerosis, Thrombosis, and Vascular Biology*, [online] 34(5), pp.985-995. Available at: http://atvb.ahajournals.org/content/suppl/2014/03/20/ATVBAHA.114.303415.DC1.html [Accessed 26 Nov. 2017].

[15] Chaudhury, H., Zakkar, M., Boyle, J., Cuhlmann, S., van der Heiden, K., Luong, L., Davis, J., Platt, A., Mason, J., Krams, R., Haskard, D., Clark, A. and Evans, P. (2010). c-Jun N-Terminal

Kinase Primes Endothelial Cells at Atheroprone Sites for Apoptosis. *Arteriosclerosis, Thrombosis, and Vascular Biology*, [online] 30(3), pp.546-553. Available at: http://atvb.ahajournals.org/cgi/content/full/30/3/546 [Accessed 20 Nov. 2017].

[16] Gerrity, R., Richardson, M., Somer, J., Bell, F. and Schwartz, C. (1977). Endothelial cell morphology in areas of in vivo Evans blue uptake in aorta of young pigs. *Am J Path*, (89), pp.313-335.

[17] Hansson, G., Chao, S., Schwartz, S. and Reidy, M. (1985). Aortic endothelial cell death and replication in normal and lipopolysaccharide-treated rats. *Am J Pathol*, (121), pp.123-127.

[18] Hu, Y., Foteinos, G., Xiao, Q. and Xu, Q. (2008). RAPID ENDOTHELIAL TURNOVER IN ATHEROSCLEROSIS-PRONE AREAS COINCIDES WITH STEM CELL REPAIR IN APOE-DEFICIENT MICE. *Atherosclerosis*, 199(2), p.467.

[19] Pavelka, J., Tel, G. and Bartosek, M. (2000). *SOFSEM'99 - Theory and Practice of Informatics*. New York: Springer.

[20] Walker, D., Hill, G., Wood, S., Smallwood, R. and Southgate, J. (2004). Agent-Based Computational Modeling of Wounded Epithelial Cell Monolayers. *IEEE Transactions on Nanobioscience*, 3(3), pp.153-163.

[21] Docs.python.org. (2017). *1. Extending Python with C or C++ — Python 3.6.3 documentation*. [online] Available at: https://docs.python.org/3/extending/extending.html [Accessed 3 Dec. 2017].

[22] Walker, D., Southgate, J., Hill, G., Holcombe, M., Hose, D., Wood, S., Mac Neil, S. and Smallwood, R. (2004). The epitheliome: agent-based modelling of the social behaviour of cells. *Biosystems*, 76(1-3), pp.89-100.

[23] Michaelis, U. (2014). Mechanisms of endothelial cell migration. *Cellular and Molecular Life Sciences*, 71(21), pp.4131-4148.

[24] Gail, M. and Boone, C. (1970). The Locomotion of Mouse Fibroblasts in Tissue Culture. *Biophysical Journal*, 10(10), pp.980-993.

[25] Seluanov, A., Hine, C., Azpurua, J., Feigenson, M., Bozzella, M., Mao, Z., Catania, K. and Gorbunova, V. (2009). Hypersensitivity to contact inhibition provides a clue to cancer resistance of naked mole-rat. *Proceedings of the National Academy of Sciences*, 106(46), pp.19352-19357.

[26] Salk, D., Bryant, E., Au, K., Hoehn, H. and Martin, G. (1981). Systematic growth studies, cocultivation, and cell hybridization studies of Werner syndrome cultured skin fibroblasts. *Human Genetics*, 58(3), pp.310-316.

[27] Jonkman, J., Cathcart, J., Xu, F., Bartolini, M., Amon, J., Stevens, K. and Colarusso, P. (2014). An introduction to the wound healing assay using live-cell microscopy. *Cell Adhesion & Migration*, 8(5), pp.440-451.

[28] Matsuda, M., Sawa, M., Edelhauser, H., Bartels, S., Neufeld, A. and Kenyon, K. (1985). Cellular migration and morphology in corneal endothelial wound repair. *Invest. Ophthalmol. Vis. Sci.*, 26(4), pp.443-449.

[29] Python.org. (n.d.). *Comparing Python to Other Languages*. [online] Available at: https://www.python.org/doc/essays/comparisons/ [Accessed 2 Apr. 2018].

[30] Python.org. (2018). *PEP 8 -- Style Guide for Python Code*. [online] Available at: https://www.python.org/dev/peps/pep-0008/ [Accessed 2 Apr. 2018].

[31] World Health Organization. (2018). *Life expectancy*. [online] Available at: http://www.who.int/gho/mortality_burden_disease/life_tables/situation_trends/en/ [Accessed 2 May 2018].

Appendix

Main Simulation Results

		Run					Average	Standard Deviation
		1	2	3	4	5		
% Senescent		0	0	0	0	0	0	0
Time to Heal (Hrs)		30	30	30	30	30	30	0
Cells in Gap	IT 1	221	184	215	217	185	204.4	7.32
	IT 2	305	252	254	294	259	272.8	9.93
	IT 3	376	333	332	342	312	339	9.36
	IT 4	394	354	342	363	329	356.4	9.84
	IT 5	427	385	395	427	361	399	11.36
	IT 6	-	-	-	-	-	-	-
	IT 7	-	-	-	-	-	-	-
	IT 8	-	-	-	-	-	-	-
	IT 9	-	-	-	-	-	-	-

Table A.1: 0% (control) Senescent Results

		Run					Average	Standard Deviation
		1	2	3	4	5		
% Senescent		2.9	2.9	2.6	2.7	2.6	2.74	0.06
Time to Heal (Hrs)		36	36	36	36	30	34.8	1.07
Cells in Gap	IT 1	140	112	119	130	159	132	7.39
	IT 2	209	149	175	168	191	178.4	9.12
	IT 3	250	194	225	216	250	227	9.53
	IT 4	260	252	279	234	288	262.6	8.61
	IT 5	323	291	316	276	315	304.2	7.95
	IT 6	378	310	342	308	-	334.5	11.40
	IT 7	-	-	-	-	-	-	-
	IT 8	-	-	-	-	-	-	-
	IT 9	-	-	-	-	-	-	-

Table A.2: 0-5% Senescent Results

		Run					Average	Standard Deviation
		1	2	3	4	5		
% Senescent		7.4	5.5	7.7	6.8	7.5	6.98	0.36
Time to Heal (Hrs)		42	36	42	36	42	39.6	1.31
Cells in Gap	IT 1	103	130	95	110	121	111.8	5.58
	IT 2	146	198	174	113	141	154.4	13.04
	IT 3	184	239	201	161	167	190.4	12.53
	IT 4	229	235	210	182	174	206	10.93
	IT 5	283	265	244	230	225	249.4	9.74
	IT 6	335	333	301	278	267	302.8	12.41
	IT 7	355	-	302	-	296	317.67	15.31
	IT 8	-	-	-	-	-	-	-
	IT 9	-	-	-	-	-	-	-

Table A.3: 5-10% Senescent Results

		Run					Average	Standard Deviation
		1	2	3	4	5		
% Senescent		14.6	12.8	12.1	11.7	13.2	12.88	0.45
Time to Heal (Hrs)		48	42	42	42	48	44.4	1.31
Cells in Gap	IT 1	65	79	74	83	83	76.8	3.03
	IT 2	99	104	109	105	92	101.8	2.61
	IT 3	126	109	141	122	129	125.4	4.64
	IT 4	147	135	143	139	151	143	2.53
	IT 5	171	157	166	164	176	166.8	2.88
	IT 6	261	215	211	222	195	220.8	9.82
	IT 7	265	222	231	252	209	235.8	9.05
	IT 8	285	-	-	-	222	253.5	22.27
	IT 9	-	-	-	-	-	-	-

Table A.4: 10-15% Senescent Results

		Run					Average	Standard Deviation
		1	2	3	4	5		
% Senescent		16.7	19.8	15.3	17.8	16.8	17.28	0.67
Time to Heal (Hrs)		48	54	43	48	54	49.4	1.87
Cells in Gap	IT 1	81	61	108	73	62	77	7.68
	IT 2	86	70	114	91	70	86.2	7.27
	IT 3	108	102	128	106	96	108	4.83
	IT 4	124	96	157	116	119	122.4	8.83
	IT 5	133	107	170	152	135	139.4	9.39
	IT 6	178	128	228	185	151	174	15.10
	IT 7	204	146	233	195	189	193.4	12.57
	IT 8	220	161	-	226	211	204.5	12.84
	IT 9	-	169	-	-	222	195.5	18.74

Table A.5: 15-20% Senescent Results

		Run					Average	Standard Deviation
		1	2	3	4	5		
% Senescent		25	25.9	23.3	23.4	20.1	23.54	0.89
Time to Heal (Hrs)		42	48	48	54	48	48	1.70
Cells in Gap	IT 1	56	54	65	58	60	58.6	1.79
	IT 2	66	74	73	87	80	76	3.16
	IT 3	96	83	100	100	81	92	3.72
	IT 4	94	114	105	103	114	106	3.36
	IT 5	114	127	143	141	115	128	5.51
	IT 6	151	150	190	145	140	155.2	7.98
	IT 7	164	166	200	165	180	175	6.17
	IT 8	-	180	219	173	188	190	8.78
	IT 9	-	-	-	182	-	182	0

Table A.6: 20-25% Senescent Results

Simulations Results with 1 Hour Time Step

		Run		Average	Standard Deviation
		1	2		
% Senescent		0	0	0	0
Time to Heal (Hrs)		21	21	21	0
Cells in Gap	**IT 1**	87	95	91	2.82
	IT 2	131	133	132	0.71
	IT 3	153	147	150	2.12
	IT 4	172	168	170	1.41
	IT 5	194	184	189	3.54
	IT 6	209	196	202.5	4.60
	IT 7	222	227	224.5	1.77
	IT 8	240	245	242.5	1.77
	IT 9	261	252	256.5	3.18
	IT 10	260	270	265	3.54
	IT 11	282	265	273.5	6.01
	IT 12	286	266	276	7.07
	IT 13	286	272	279	4.95
	IT 14	281	278	279.5	1.06
	IT 15	285	277	281	2.83
	IT 16	281	283	282	0.71
	IT 17	278	291	284.5	4.60
	IT 18	290	301	295.5	3.89
	IT 19	291	302	296.5	3.89
	IT 20	306	305	305.5	0.35
	IT 21	305	314	305.5	3.18

Table A.7: 0% senescence results with each iteration = 1 hour

		Run		Average	Standard Deviation
		1	2		
% Senescent		4	2.7	3.35	0.46
Time to Heal (Hrs)		29	27	28	0.71
Cells in Gap	**IT 1**	87	46	66.5	14.50
	IT 2	104	70	87	12.02
	IT 3	115	89	102	9.19
	IT 4	131	100	115.5	10.96
	IT 5	140	116	128	8.49
	IT 6	152	126	139	9.19
	IT 7	160	133	146.5	9.55
	IT 8	162	148	155	4.95
	IT 9	163	161	162	0.71
	IT 10	167	163	165	1.41
	IT 11	169	166	167.5	1.06
	IT 12	178	175	176.5	1.06
	IT 13	180	173	176.5	2.47
	IT 14	176	173	174.5	1.06
	IT 15	183	177	180	2.12
	IT 16	178	181	179.5	1.06
	IT 17	178	190	184	4.24

	IT 18	186	185	185.5	0.35
	IT 19	200	197	198.5	1.06
	IT 20	212	199	205.5	4.60
	IT 21	215	205	210	3.54
	IT 22	219	219	219	0.00
	IT 23	229	227	228	0.71
	IT 24	236	228	232	2.83
	IT 25	246	250	248	1.41
	IT 26	270	263	266.5	2.47
	IT 27	276	259	267.5	6.01
	IT 28	270	-	270	0.00
	IT 29	274	-	274	0.00

Table A.8: 0-5% senescence results with each iteration = 1 hour

		Run		**Average**	**Standard Deviation**
		1	**2**		
% Senescent		5.7	7.5	6.6	0.25
Time to Heal (Hrs)		31	36	33.5	0.71
Cells in Gap	**IT 1**	41	50	45.5	1.27
	IT 2	68	68	68	0.00
	IT 3	79	73	76	0.85
	IT 4	98	83	90.5	2.12
	IT 5	107	85	96	3.11
	IT 6	108	89	98.5	2.69
	IT 7	108	94	101	1.98
	IT 8	114	99	106.5	2.12
	IT 9	120	102	111	2.55
	IT 10	113	112	112.5	0.14
	IT 11	112	104	108	1.13
	IT 12	107	108	107.5	0.14
	IT 13	106	115	110.5	1.27
	IT 14	110	113	111.5	0.42
	IT 15	107	114	110.5	0.99
	IT 16	104	117	110.5	1.84
	IT 17	109	113	111	0.57
	IT 18	110	114	112	0.57
	IT 19	124	114	119	1.41
	IT 20	131	121	126	1.41
	IT 21	145	124	134.5	2.97
	IT 22	146	137	141.5	1.27
	IT 23	154	146	150	1.13
	IT 24	170	164	167	0.85
	IT 25	178	173	175.5	0.71
	IT 26	177	203	190	3.68
	IT 27	191	208	199.5	2.40
	IT 28	195	201	198	0.85
	IT 29	207	200	203.5	0.99
	IT 30	204	196	200	1.13
	IT 31	202	197	199.5	0.71
	IT 32	-	192	192	0.00
	IT 33	-	191	191	0.00
	IT 34	-	190	190	0.00

	IT 35	-	190	190	0.00
	IT 36	-	200	200	0.00

Table A.9: 5-10% senescence results with each iteration = 1 hour

		Run		**Average**	**Standard Deviation**
		1	**2**		
% Senescent		12.2	12.6	12.4	0.14
Time to Heal (Hrs)		37	34	35.5	1.06
Cells in Gap	**IT 1**	42	25	33.5	6.01
	IT 2	52	36	44	5.66
	IT 3	56	42	49	4.95
	IT 4	64	49	56.5	5.30
	IT 5	69	58	63.5	3.89
	IT 6	70	61	65.5	3.18
	IT 7	71	64	67.5	2.47
	IT 8	73	64	68.5	3.18
	IT 9	75	63	69	4.24
	IT 10	80	67	73.5	4.60
	IT 11	82	70	76	4.24
	IT 12	86	71	78.5	5.30
	IT 13	97	78	87.5	6.72
	IT 14	99	77	88	7.78
	IT 15	100	83	91.5	6.01
	IT 16	103	77	90	9.19
	IT 17	116	75	95.5	14.50
	IT 18	120	88	104	11.31
	IT 19	121	92	106.5	10.25
	IT 20	126	94	110	11.31
	IT 21	137	98	117.5	13.79
	IT 22	145	116	130.5	10.25
	IT 23	152	119	135.5	11.67
	IT 24	168	124	146	15.56
	IT 25	180	136	158	15.56
	IT 26	176	142	159	12.02
	IT 27	185	145	165	14.14
	IT 28	186	150	168	12.73
	IT 29	179	155	167	8.49
	IT 30	172	156	164	5.66
	IT 31	177	158	167.5	6.72
	IT 32	174	152	163	7.78
	IT 33	166	155	160.5	3.89
	IT 34	170	156	163	4.95
	IT 35	168	-	168	0.00
	IT 36	174	-	174	0.00
	IT 37	169	-	169	0.00

Table A.10: 10-15% senescence results with each iteration = 1 hour

		Run		Average	Standard Deviation
		1	2		
% Senescent		16.1	18.9	17.5	0.99
Time to Heal (Hrs)		38	46	42	2.83
Cells in Gap	**IT 1**	32	24	28	2.83
	IT 2	48	29	38.5	6.72
	IT 3	54	37	45.5	6.01
	IT 4	55	48	51.5	2.47
	IT 5	64	47	55.5	6.01
	IT 6	70	48	59	7.78
	IT 7	71	55	63	5.66
	IT 8	74	50	62	8.49
	IT 9	70	53	61.5	6.01
	IT 10	76	57	66.5	6.72
	IT 11	73	58	65.5	5.30
	IT 12	76	59	67.5	6.01
	IT 13	78	59	68.5	6.72
	IT 14	84	62	73	7.78
	IT 15	80	62	71	6.36
	IT 16	87	67	77	7.07
	IT 17	89	68	78.5	7.42
	IT 18	93	73	83	7.07
	IT 19	101	71	86	10.61
	IT 20	106	76	91	10.61
	IT 21	100	74	87	9.19
	IT 22	102	68	85	12.02
	IT 23	104	71	87.5	11.67
	IT 24	113	83	98	10.61
	IT 25	123	95	109	9.90
	IT 26	134	98	116	12.73
	IT 27	134	99	116.5	12.37
	IT 28	138	108	123	10.61
	IT 29	129	105	117	8.49
	IT 30	125	116	120.5	3.18
	IT 31	129	113	121	5.66
	IT 32	143	109	126	12.02
	IT 33	136	108	122	9.90
	IT 34	145	105	125	14.14
	IT 35	144	112	128	11.31
	IT 36	152	113	132.5	13.79
	IT 37	146	107	126.5	13.79
	IT 38	152	114	133	13.44
	IT 39	-	109	109	0.00
	IT 40	-	114	114	0.00
	IT 41	-	114	114	0.00
	IT 42	-	118	118	0.00
	IT 43	-	121	121	0.00
	IT 44	-	117	117	0.00
	IT 45	-	115	115	0.00
	IT 46	-	120	120	0.00

Table A.11: 15-20% senescence results with each iteration = 1 hour

		Run		Average	Standard Deviation
		1	2		
% Senescent		20	24	22	1.41
Time to Heal (Hrs)		39	44	41.5	1.77
Cells in Gap	**IT 1**	25	18	21.5	2.47
	IT 2	35	25	30	3.54
	IT 3	43	29	36	4.95
	IT 4	51	33	42	6.36
	IT 5	53	40	46.5	4.60
	IT 6	53	39	46	4.95
	IT 7	56	44	50	4.24
	IT 8	61	50	55.5	3.89
	IT 9	61	46	53.5	5.30
	IT 10	62	52	57	3.54
	IT 11	67	55	61	4.24
	IT 12	65	59	62	2.12
	IT 13	69	53	61	5.66
	IT 14	79	55	67	8.49
	IT 15	75	60	67.5	5.30
	IT 16	78	60	69	6.36
	IT 17	78	60	69	6.36
	IT 18	77	63	70	4.95
	IT 19	82	60	71	7.78
	IT 20	83	63	73	7.07
	IT 21	91	64	77.5	9.55
	IT 22	99	70	84.5	10.25
	IT 23	98	73	85.5	8.84
	IT 24	98	76	87	7.78
	IT 25	106	91	98.5	5.30
	IT 26	118	92	105	9.19
	IT 27	126	96	111	10.61
	IT 28	130	96	113	12.02
	IT 29	136	95	115.5	14.50
	IT 30	134	95	114.5	13.79
	IT 31	141	104	122.5	13.08
	IT 32	143	101	122	14.85
	IT 33	141	106	123.5	12.37
	IT 34	140	104	122	12.73
	IT 35	155	111	133	15.56
	IT 36	154	110	132	15.56
	IT 37	147	110	128.5	13.08
	IT 38	158	114	136	15.56
	IT 39	156	115	135.5	14.50
	IT 40	-	114	114	0.00
	IT 41	-	110	110	0.00
	IT 42	-	109	109	0.00
	IT 43	-	106	106	0.00
	IT 44	-	109	109	0.00

Table A.12: 20-25% senescence results with each iteration = 1 hour

Sensitivity Analysis Results

		Migration * 2	Migration / 2	Mitosis * 2	Mitosis /2
% Senescence		0	0	0	0
Time to Heal (Hrs)		30	30	24	48
Cells in gap	**IT 1**	153	162	144	199
	IT 2	226	236	261	276
	IT 3	290	315	414	329
	IT 4	317	324	475	341
	IT 5	338	366	-	362
	IT 6	-	-	-	374
	IT 7	-	-	-	402
	IT 8	-	-	-	399
	IT 9	-	-	-	-

Table A.13: 0% senescence sensitivity analysis

		Migration * 2	Migration / 2	Mitosis * 2	Mitosis /2
% Senescence		2.6	3	3	3.4
Time to Heal (Hrs)		36	36	24	72
Cells in gap	**IT 1**	127	138	141	91
	IT 2	186	197	223	133
	IT 3	234	235	321	149
	IT 4	281	258	376	163
	IT 5	325	298	-	189
	IT 6	387	332	-	218
	IT 7	-	-	-	206
	IT 8	-	-	-	235
	IT 9	-	-	-	250
	IT 10	-	-	-	273
	IT 11	-	-	-	293
	IT 12	-	-	-	302

Table A.14: 0-5% senescence sensitivity analysis

		Migration * 2	Migration / 2	Mitosis * 2	Mitosis /2
% Senescence		5.8	6.7	6.1	7.8
Time to Heal (Hrs)		36	42	24	84
Cells in gap	**IT 1**	107	87	126	62
	IT 2	146	131	205	82
	IT 3	198	167	277	114
	IT 4	212	188	348	115
	IT 5	234	225	-	123
	IT 6	282	283	-	125
	IT 7	-	300	-	140

	IT 8	-	-	-	139
	IT 9	-	-	-	165
	IT 10	-	-	-	199
	IT 11	-	-	-	190
	IT 12	-	-	-	182
	IT 13	-	-	-	205
	IT 14	-	-	-	229

Table A.15: 5-10% senescence sensitivity analysis

		Migration * 2	**Migration / 2**	**Mitosis * 2**	**Mitosis /2**
% Senescence		11.3	13.8	8.5	21.6
Time to Heal (Hrs)		48	42	24	102
Cells in gap	**IT 1**	72	49	122	51
	IT 2	99	76	172	69
	IT 3	128	121	239	60
	IT 4	143	153	314	71
	IT 5	148	201	-	82
	IT 6	195	222	-	87
	IT 7	241	218	-	91
	IT 8	255	-	-	95
	IT 9	-	-	-	116
	IT 10	-	-	-	121
	IT 11	-	-	-	120
	IT 12	-	-	-	133
	IT 13	-	-	-	148
	IT 14	-	-	-	147
	IT 15	-	-	-	143
	IT 16	-	-	-	140
	IT 17	-	-	-	155

Table A.16: 10-15% senescence sensitivity analysis

		Migration * 2	**Migration / 2**	**Mitosis * 2**	**Mitosis /2**
% Senescence		19	25.9	12.2	27
Time to Heal (Hrs)		54	84	30	96
Cells in gap	**IT 1**	55	56	110	38
	IT 2	81	66	147	51
	IT 3	118	65	222	58
	IT 4	137	74	275	70
	IT 5	144	77	322	70
	IT 6	181	77	-	75
	IT 7	199	82	-	87
	IT 8	227	90	-	79
	IT 9	254	107	-	88
	IT 10	-	121	-	108
	IT 11	-	115	-	117
	IT 12	-	122	-	117
	IT 13	-	129	-	123
	IT 14	-	138	-	120
	IT 15	-	-	-	135
	IT 16	-	-	-	124

Table A.17: 15-20% senescence sensitivity analysis

		Migration * 2	Migration / 2	Mitosis * 2	Mitosis /2
% Senescence		23.8	21.2	17.9	35.7
Time to Heal (Hrs)		48	42	30	90
Cells in gap	**IT 1**	59	52	90	44
	IT 2	83	86	136	55
	IT 3	102	111	167	61
	IT 4	122	127	243	54
	IT 5	150	137	282	61
	IT 6	185	163	-	64
	IT 7	182	186	-	70
	IT 8	193	-	-	75
	IT 9	-	-	-	80
	IT 10	-	-	-	81
	IT 11	-	-	-	86
	IT 12	-	-	-	95
	IT 13	-	-	-	101
	IT 14	-	-	-	102
	IT 15	-	-	-	111

Table A.18: 20-25% senescence sensitivity analysis